Competency Based Learning Material (CBLM)

Web Design

Level-3

Module 2: Working with HTML

Code: CBLM-ICT-WD-02-L3-EN-V1

National Skills Development Authority
Prime Minister's Office
Government of the People's Republic of Bangladesh

Copyright

National Skills Development Authority
Prime Minister's Office
Level: 10-11, Biniyog Bhaban,
E-6 / B, Agargaon, Sher-E-Bangla Nagar Dhaka-1207, Bangladesh.
Email: ec@nsda.gov.bd
Website: www.nsda.gov.bd.
National Skills Portal: http:\\skillsportal.gov.bd

The CBLM on "Work with HTML" is developed based on NSDA approved Competency Standards and Competency Based Curriculum under Web Design, Level-3 Occupation. It contains the information required to implement the Web Design Level-3 standard.

This document has been prepared by NSDA with the help of relevant experts, trainers/professionals.

All Government-Private-NGO training institutes in the country accredited by NSDA can use this CBLM to implement skill-based training of Web Design level-3 course.

This document has been developed by NSDA in association with industry representatives, academia, related specialist, trainer and related employee.

Public and private institutions may use the information contained in this CBLM for activities benefitting Bangladesh.

Approved by

---th Executive Committee (EC) Meeting of NSDA

Held on ----------

How to use this Competency Based Learning Materials (CBLMs)

The module, Working with HTML contains training materials and activities for you to complete. These activities may be completed as part of structured classroom activities or you may be required you to work at your own pace. These activities will ask you to complete associated learning and practice activities in order to gain knowledge and skills you need to achieve the learning outcomes.

1. Review the **Learning Activity** page to understand the sequence of learning activities you will undergo. This page will serve as your road map towards the achievement of competence.

2. Read the **Information Sheets.** This will give you an understanding of the jobs or tasks you are going to learn how to do. Once you have finished reading the **Information Sheets** complete the questions in the **Self-Check.**

3. **Self-**Checks are found after each **Information Sheet**. **Self-Checks** are designed to help you know how you are progressing. If you are unable to answer the questions in the **Self-Check** you will need to re-read the relevant **Information Sheet**. Once you have completed all the questions check your answers by reading the relevant **Answer Keys** found at the end of this module.

4. Next move on to the **Job Sheets. Job Sheets** provide detailed information about *how to do the job* you are being trained in. Some **Job Sheets** will also have a series of **Activity Sheets**. These sheets have been designed to introduce you to the job step by step. This is where you will apply the new knowledge you gained by reading the Information Sheets. This is your opportunity to practice the job. You may need to practice the job or activity several times before you become competent.

5. Specification **sheets**, specifying the details of the job to be performed will be provided where appropriate.

6. A review of competency is provided on the last page to help remind if all the required assessment criteria have been met. This record is for your own information and guidance and is not an official record of competency

When working though this Module always be aware of your safety and the safety of others in the training room. Should you require assistance or clarification please consult your trainer or facilitator.

When you have satisfactorily completed all the Jobs and/or Activities outlined in this module, an assessment event will be scheduled to assess if you have achieved competency in the specified learning outcomes. You will then be ready to move onto the next Unit of Competency or Module

Table of Contents

Module Content

Unit Title: Work with HTML
Unit Code: OU- ICT-WD-02-L3-V1
Module Title: Working with HTML

Module Descriptor: This module encompasses the necessary knowledge, skills, and attitudes (KAS) for establishing work with HTML. It encompasses proficiencies such as introducing HTML, applying fundamentals of typography, creating HTML pages, and using HTML form.

Nominal Duration: 35 Hours

Learning Outcomes:

Upon completion of this module the trainees must be able to:

1. Demonstrate an understanding of HTML.
2. Utilize the fundamentals of typography.
3. Create an HTML page.
4. Construct and implement HTML forms.

Assessment Criteria:

1. The structure of HTML (Hypertext Mark-up Language) is interpreted.
2. DHTML tags are introduced.
3. Entities & attributes of HTML are interpreted.
4. Typography is interpreted.
5. Guidelines for web typography are applied.
6. Guidelines for print typography are applied.
7. Most common HTML tags are used.
8. Most common entities & attributes are used.
9. HTML multicolumn layout is implemented.
10. HTML Graphics are used.
11. HTML Media is used.
12. HTML page is saved.
13. HTML form elements are used.
14. HTML input attributes are used.
15. HTML form validation is used.
16. A webpage is created using form attributes.

Learning Outcome 1: Demonstrate an understanding of HTML

Assessment Criteria:	1. Structure of HTML (Hypertext Mark-up Language) is interpreted. 2. DHTML tags are introduced 3. Entities & attributes of HTML are interpreted.
Conditions and Resources	1. Applicable tools, utensils, and equipment as prescribed by competency standard. 2. Supply materials 3. Relevant ingredients 4. CBLM related to the learning outcome. 5. Instructions, job sheets, activity sheets, and standard operating procedures 6. Personal protective equipment 7. Module/reference
Contents	1. Structure of HTML (Hypertext Markup Language) 2. DHTML Tags 3. Entities and Attributes in HTML
Training Methods	1. Discussion 2. Presentation 3. Demonstration 4. Guided Practice 5. Individual Practice 6. Project Work 7. Problem Solving 8. Brainstorming
Learning Materials	1. CBLM 2. Handouts 3. Books, Manuals 4. Module/ Reference 5. Paper 6. Pen
Assessment Methods	1. Written Test 2. Demonstration 3. Oral Questioning

Learning Experience: Demonstrate an understanding of HTML

You must perform the learning steps below to achieve the objectives stated in this learning guide. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Steps	Resources specific instructions
1. Student will ask the instructor about work with HTML	1. The instructor will provide the learning materials for` demonstrate an understanding of HTML.
2. Read the **Information sheet/s**	2. Information Sheet No: 1. Demonstrate an understanding of HTML.
3. Complete the **Self-Checks & Answer key sheets.**	3. Self-Check No: 2.1 Demonstrate an understanding of HTML. Answer key No. 2.1 Demonstrate an understanding of HTML.
4. Read the **Job/ Task sheet** and **Specification Sheet**	4. Job/ task sheet and specification sheet Individual Activity: ▪ Task Sheet No: 01 Building a Basic HTML Web Page

Information Sheet 1: Demonstrate an understanding of HTML

Learning Objective:

After completion of this information sheet, the learners will be able to explain, define and interpret the following contents:

1.1 The Structure of HTML (Hypertext Markup Language)
1.2 DHTML Tags
1.3 Entities and Attributes in HTML

1.1 The Structure of HTML (Hypertext Markup Language)

HTML (Hypertext Markup Language) is the standard markup language used for creating web pages. It defines the structure and content of a web page by using a set of tags and elements. Here is an overview of the structure of HTML:

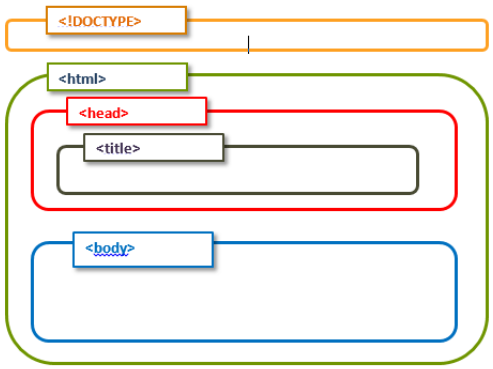

Figure 1: The Structure of HTML

1.1.1 Document Type Declaration (DTD):

HTML documents start with a Document Type Declaration, which specifies the HTML version and document type. For example, in HTML5, the declaration is:

<!DOCTYPE html>

1.1.2 HTML Element:

The HTML element is the root element of an HTML document. It wraps the entire content of the page and is represented by the **<html>** tag. All other elements are descendants of the HTML element.

```
<html>
<!-- Content goes here -->
</html>
```

1.1.3 Structural Elements:

HTML provides a set of structural elements that define the layout and organization of the content within the **<body>** section. These elements include headings, paragraphs, lists, divisions, and more. Here are a few examples:

```
<h1>Heading 1</h1>
```

```
<p>This is a paragraph.</p>
<ul>
<li>List item 1</li>
<li>List item 2</li>
</ul>
<div>This is a division.</div>
```

1.1.4 Semantic Elements:

HTML5 introduced semantic elements that provide meaning and structure to the content. These elements describe the type of content they contain, making it easier for search engines and assistive technologies to understand the page. Examples of semantic elements include **<header>, <nav>, <article>, <section>, <footer>,** and more.

```
<header>
<!-- Header content -->
</header>
<nav>
<!-- Navigation menu -->
</nav>
<article>
<!-- Article content -->
</article>
<section>
<!-- Section content -->
</section>
<footer>
<!-- Footer content -->
</footer>
```

1.1.5 Tags and Attributes:

HTML tags are used to define different elements, and they are enclosed in angle brackets (**<>**). Tags can have attributes, which provide additional information or modify the behavior of the element. Attributes are specified within the opening tag of an element. Here's an example:

```
<a href="https://example.com">Link</a>
```

In this example, <a> is the anchor tag used for creating links. The href attribute specifies the URL that the link points to.

This is a basic overview of the structure of HTML. HTML is a versatile language, and there are many more elements, attributes, and features available for creating rich and interactive web pages.

1.2 DHTML Tags

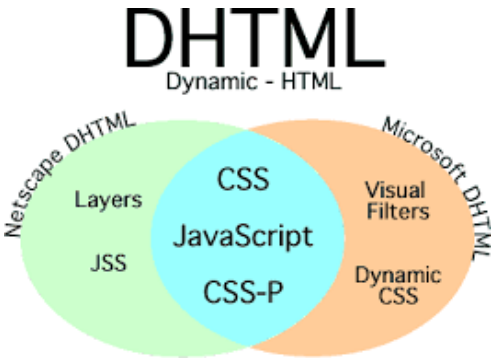

Figure 2: DHTML Tags

DHTML (Dynamic HTML) is a combination of technologies used to create dynamic and interactive web pages. It combines HTML, CSS (Cascading Style Sheets), and JavaScript to allow for more dynamic and responsive web content. DHTML enables the manipulation of HTML elements, styles, and behavior on the client-side, providing a more interactive user experience.

1.2.1 DHTML Tags:

DHTML doesn't introduce new tags but rather utilizes existing HTML tags along with JavaScript and CSS to create dynamic effects. Here are some commonly used tags and techniques in DHTML:

- ** ** and **:**
  These tags are frequently used in DHTML to create containers for grouping and styling content. They can be manipulated using JavaScript to change their properties, visibility, and position on the page.

- **<script>**:
  The <script> tag is used to embed JavaScript code within an HTML document. JavaScript provides the interactivity and dynamic behavior in DHTML, allowing for things like animations, event handling, and DOM manipulation.

- **Event Handlers**:
  DHTML heavily relies on event handlers to respond to user actions or specific events. Event handlers can be added to HTML elements using attributes such as **onclick, onmouseover, onmouseout,** etc. These event handlers call JavaScript functions that define the behavior in response to the events.

- **CSS:**
  Cascading Style Sheets (CSS) are used in DHTML to control the presentation and styling of HTML elements. CSS can be dynamically modified through JavaScript to change the appearance of elements or create animations and transitions.

- **Dynamic Content Loading:**

DHTML enables the dynamic loading of content into a web page without requiring a full page refresh. This is often achieved using techniques like AJAX (Asynchronous JavaScript and XML) or modern approaches like Fetch API and XMLHttp Request.

- **Animation and Effects:**

DHTML allows for the creation of animations and effects using JavaScript and CSS. This can include transitions, fade-ins, slide-outs, and other visual effects that enhance the user experience.
These are just a few examples of how DHTML combines HTML, CSS, and JavaScript to create dynamic and interactive web pages. The specific techniques and tags used in DHTML can vary depending on the desired functionality and the tools and frameworks employed.

1.3 Entities and Attributes in HTML

In HTML, entities and attributes are two different concepts related to the structure and content of an HTML document.

1.3.1 Entities:

Entities in HTML are special character codes used to represent characters that have special meaning or cannot be easily typed or displayed in HTML directly. Entities start with an ampersand (&) and end with a semicolon (;). They are primarily used to display reserved characters or to avoid conflicts with HTML syntax. For example:

- **<** represents the less-than symbol “<”.
- **>** represents the greater-than symbol >.
- **&** represents the ampersand itself &.
- **"** represents the double quotation mark ".

Entities are often used when you want to display special characters or symbols, such as copyright **(©)**, registered trademark **(®),** or non-breaking space **( )**

1.3.2 Attributes:

Attributes in HTML provide additional information or modify the behavior of HTML elements. They are specified within the opening tag of an element. Attributes consist of a name and a value, separated by an equal sign (=) and enclosed in quotation marks. Attributes can be classified into two types:

- **Global attributes:** These attributes can be used with any HTML element. Some common global attributes include class, id, style, title, and data-* attributes. For example:

Html code

```
<div class="container" id="myDiv" style="color: red;" data-value="123">Content</div>
```

- **Element-specific attributes:** These attributes are specific to certain HTML elements and provide additional functionality or behavior for those elements. For example, the <a> tag has attributes like href for specifying the link URL and target for determining how the link opens. Example:

Html code

```
<a href="https://example.com" target="_blank">Click here</a>
```

Attributes can be used to define the appearance, behavior, or functionality of HTML elements, such as setting the size of an image (**width** and **height** attributes), specifying the source of an iframe (**src** attribute), or defining alternative text for an image (**alt** attribute).

It's important to note that not all attributes are applicable to all HTML elements. Each element has its own set of valid attributes, and using inappropriate attributes may result in invalid HTML markup.

Entities and attributes are both integral to HTML, helping to represent special characters correctly and providing additional information to elements, respectively.

Self-Check Sheet – 1: Demonstrate an understanding of HTML

Questionnaire:

1. What does HTML stand for?

   Answer:

2. What is the purpose of HTML?

   Answer:

3. What is the root element of an HTML document?

   Answer:

4. What section of an HTML document contains meta information?

   Answer:

5. How do you create a hyperlink in HTML?

   Answer:

6. What is the purpose of the tag?

   Answer:

7. How do you add an image in HTML?

   Answer:

8. What is the purpose of the <table> tag in HTML?

   Answer:

9. How do you create a numbered list in HTML?

   Answer:

10. What does the <form> tag in HTML represent?

    Answer:

Answer Key – 1: Demonstrate an understanding of HTML

1. What does HTML stand for?
   Answer: HTML stands for Hypertext Markup Language.

2. What is the purpose of HTML?
   Answer: HTML is used to structure and present content on the web

3. What is the root element of an HTML document?
   Answer: The root element is <html>.

4. What section of an HTML document contains meta information?
   Answer: The <head> section contains meta-information

5. How do you create a hyperlink in HTML?
   Answer: Hyperlinks are created using the <a> tag with the href attribute

6. What is the purpose of the tag?
   Answer: The tag is used as a container to group and style content

7. How do you add an image in HTML?
   Answer: Images are inserted using the <img> tag with the src attribute

8. What is the purpose of the <table> tag in HTML?
   Answer: The <table> tag is used to create tabular data

9. How do you create a numbered list in HTML?
   Answer: Numbered lists are created using the <ol> tag with <li> tags for each list Item

10. What does the <form> tag in HTML represent?
    Answer: The <form> tag is used to create interactive forms for user input

Task Sheet 1: Build a Basic HTML Web Page

Objective:

- Understand the basic structure and elements of an HTML document.
- Create a simple web page using HTML tags.
- Add text content, headings, and paragraphs to the web page.
- Insert images into the web page.
- Create hyperlinks to other web pages.
- Apply basic styling using CSS.

Required Equipment:

- A Computer with an Internet connection
- Computer with a text editor or integrated development environment (IDE).
- Web browser for previewing the web page.

Procedure:

1. Open a text editor or IDE on your computer.
2. Create a new HTML file and save it with a .html extension.
3. Start building the HTML structure:
    - Add the HTML doctype declaration at the top of the file: <!DOCTYPE html>.
    - Create the opening and closing HTML tags: <html> and </html>.
    - Within the HTML tags, create the opening and closing head tags: <head> and </head>.
    - Inside the head tags, include the title of your web page using the <title> tag.
    - After the head section, create the opening and closing body tags: <body> and </body>.
4. Within the body tags, start adding content to your web page:
    - Add a heading to the web page using the <h1> or <h2> tags.
    - Insert paragraphs using the <p> tag to provide text content.
    - Use the <img> tag to add an image to your web page, specifying the image source (src attribute), alt text (alt attribute), and any desired width or height attributes.
    - Create hyperlinks to other web pages using the <a> tag, providing the destination URL (href attribute) and the anchor text.
5. Apply basic CSS styling:
    - Create a new CSS file and save it with a .css extension.
    - Link the CSS file to your HTML document by adding the <link> tag with the appropriate href and rel attributes within the head section.
    - Inside the CSS file, target HTML elements or classes and apply desired styles using CSS properties like color, font size, background color, etc.
6. Save your HTML and CSS files.
7. Open your web page in a web browser to preview and test its functionality.
8. Make any necessary adjustments or modifications to the HTML and CSS code.
9. Repeat steps 6-8 until you are satisfied with the final result.

Note: This job sheet provides a general guideline for building a basic HTML web page. You can customize the content, styling, and structure based on your specific requirements and creativity.

Learning Outcome 2: Utilize the fundamentals of typography

Assessment Criteria:	1. Typography is interpreted. 2. Guidelines for web typography are applied. 3. Guidelines for print typography are applied.
Conditions and Resources	1. Applicable tools, utensils, and equipment as prescribed by competency standards. 2. Supply materials 3. Relevant ingredients 4. CBLM related to the learning outcome. 5. Instructions, job sheets, activity sheets, and standard operating procedures 6. Personal protective equipment 7. Module/reference
Contents	1. Typography: An Introduction 2. Typography Guidelines for Web Design 3. Typography Guidelines for Print Design
Training Methods	1. Discussion 2. Presentation 3. Demonstration 4. Guided Practice 5. Individual Practice 6. Project Work 7. Problem Solving 8. Brainstorming
Learning Materials	1. CBLM 2. Handouts 3. Books, Manuals 4. Module/ Reference 5. Paper 6. Pen
Assessment Methods	1. Written Test 2. Demonstration 3. Oral Questioning

Learning Experience: Utilize the fundamentals of typography

You must perform the learning steps below to achieve the objectives stated in this learning guide. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Steps	Resources specific instructions
1. Students will ask the instructor about work with HTML	1. The instructor will provide the learning materials for utilize the fundamentals of typography
1. Read the **Information sheet/s**	2. Information Sheet No: 2. Utilize the fundamentals of typography
1. Complete the **Self-Checks & Answer key sheets.**	3. Self-Check No: 2 Utilize the fundamentals of typography Answer key No. 2. Utilize the fundamentals of typography
1. Read the **Job/ Task sheet** and **Specification Sheet**	4. Job/ task sheet and specification sheet ▪ Job Sheet No: 02 Create an HTML Form

Information Sheet 2: Utilize the fundamentals of typography

Learning Objective: After completion of this information sheet, the learners will be able to interpret the following contents:

1.1 Typography
1.2 Typography Guidelines for Web Design
1.3 Typography Guidelines for Print Design

4.1 Typography:

Typography in web design refers to the art and technique of arranging and presenting text on a website. It involves selecting and styling fonts, adjusting their size, spacing, and color, and organizing them in a visually appealing and readable manner.

Typography plays a crucial role in web design because it directly impacts the user experience and the overall aesthetics of a website. Well-chosen typography can enhance the readability of content, establish a visual hierarchy, convey emotions or brand identity, and improve overall user engagement.

Here are some key aspects of typography in web design:

- Font selection: Choosing appropriate fonts that align with the website's purpose, content, and target audience. There are various categories of fonts, such as serif, sans-serif, script, and display fonts, each with its own characteristics and connotations.

- Font styling: Modifying the appearance of fonts by adjusting attributes like size, weight (boldness), style (italic, oblique), and letter-spacing. These choices should be consistent and harmonious throughout the website.

- Hierarchy and formatting: Creating a visual hierarchy by varying font sizes, weights, and styles to distinguish different levels of headings, subheadings, and body text. Formatting techniques like using bullet points, block quotes, and italics can also enhance readability and comprehension.

- Line length and spacing: Determining the optimal line length (number of characters per line) to ensure comfortable reading. Additionally, adjusting line spacing (leading) and letter spacing (tracking/kerning) can enhance legibility and visual appeal.

- Responsive typography: Adapting typography for different screen sizes and devices, ensuring that text remains readable and visually pleasing across various resolutions and orientations.

- Accessibility: Considering the accessibility needs of users by ensuring sufficient color contrast between text and background, using scalable fonts, and providing alternative text for images.

Designers often combine multiple typefaces (font families) to create contrast and visual interest, but it's important to strike a balance and maintain consistency to avoid overwhelming the reader. Typography should align with the overall design concept and brand identity, reinforcing the website's message and enhancing the user's experience.

4.2 Typography Guidelines for Web Design

Typography guidelines for web design are a set of principles and best practices that help designers create effective and visually appealing typographic layouts for websites. These guidelines ensure that the typography is readable, consistent, and aligned with the overall design concept.

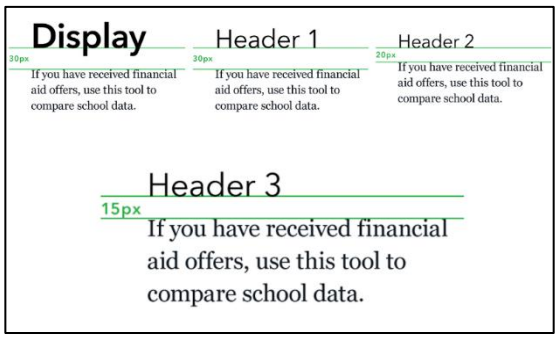

Figure 3: Typography Guidelines for Web Design

4.2.1 Here are some key typography guidelines for web design:

- Font selection: Choose legible and appropriate fonts for the website's purpose and target audience. Use a combination of fonts for headings, subheadings, and body text to create visual contrast and hierarchy. Limit the number of fonts used to maintain consistency and readability.

- Readability: Prioritize readability by selecting fonts with clear letterforms and appropriate spacing. Avoid decorative or overly stylized fonts that can be difficult to read, especially in longer passages of text. Ensure there is sufficient contrast between the text and the background to make it easily readable.

- Font size: Use an appropriate font size for different elements of the website. Headings should be larger and more prominent than body text to create a visual hierarchy. Body text should be large enough to be comfortably read on various devices and screen sizes.

- Line length and spacing: Limit the line length to a readable width by adjusting the width of the text container. Longer lines can be harder to read, so it's recommended to keep the line length between 45-75 characters. Use appropriate line spacing (leading) to ensure sufficient space between lines for readability.

- Font weight and style: Utilize different font weights (such as regular, bold, and italic) to create emphasis and hierarchy. Reserve bold or italic styles for specific purposes, such as highlighting key points or distinguishing quotes.

- Consistency: Maintain consistency in typography throughout the website. Use the same font families, weights, and styles across different pages and sections to establish a cohesive visual identity. Consistency also applies to text formatting, such as indentation, bullet points, and paragraph spacing.

- Alignment and whitespace: Ensure proper alignment of text elements to create a visually pleasing layout. Use whitespace effectively to separate different sections, paragraphs, and elements. Sufficient whitespace around text improves readability and reduces visual clutter.

- Responsive typography: Optimize typography for different screen sizes and devices. Consider using responsive font sizes that adjust dynamically based on the screen size. Ensure that text remains legible and well-proportioned across various resolutions and orientations.

- Accessibility: Follow accessibility guidelines to make typography inclusive for all users. Provide sufficient color contrast between text and background for readability. Ensure that font sizes are scalable and can be increased without loss of readability. Use alternative text for images and provide captions for videos.

4.2.2 Website Typography Guidelines

- Limit the number of typefaces per website.
- Use a sans-serif font for body text.
- Stick to standard fonts at first.
- Size your text appropriately.
- Don’t use all caps.
- Use colors carefully and intentionally.
- Stay between around 40 and 80 characters per line.
- Provide sufficient spacing between lines.
- Eliminate text animations.

You probably haven’t had to think much about typography on the web. That’s because popular websites have typography down to a well-styled T. Many years of trial, error, and experimentation have left us with 9 best practices — follow them to ensure your site’s typography meets expectations and keeps users reading (or at least skimming).

4.2.3 Limit the number of typefaces per website

To maintain visual cohesion throughout your content, use no more than two different typefaces across your website. Many websites do just fine with one typeface, especially if they apply different fonts within the typeface for different purposes (like headings, body, button text, etc.). For instance:

If you opt for two different typefaces, select options that are visually compatible but distinguishable from each other.

4.2.4 Use a sans serif font for body text

While serif fonts are common in printed text, typography experts generally agree that sans serif fonts are actually more readable in digital contexts. Our eyes follow web text better without the decorations.
This doesn't mean you can't use serif fonts on your website at all — a serif text in a title, heading, pull quote, or decorative section can draw attention and provide nice contrast. However, for blocks of text that require more effort to read and understand, your text is better "sans."

4.2.5 Stick to standard fonts at first

By "standard," I don't mean "plain" or "boring," but "compatible." Choosing a web-safe font ensures your text is easily readable for everyone through any digital means.

There are a few advantages to standard fonts. First, all web-safe fonts will render on every web browser and device, be it desktop or mobile. If a font isn't recognized, the system will default to a font that might look worse.
Second, readers are accustomed to seeing standard fonts online. They won't be distracted by the appearance of the text, and will be able to scan it more quickly. Ultimately, your typography should help the reader, not distract them from the content they want.

Third, web-safe fonts tend to lack some inconvenient design flaws seen in other fonts. A standard font, for example, won't contain any strange kerning that make two letters look stuck together. They also minimize instances in which two different characters are hard to distinguish, like "**I"** form "**L**" or "**r"** from "**n**".

If you'd prefer a non-standard typeface or font, there's always a chance some browsers won't recognize the style and instead display something like plain Times New Roman. There's nothing wrong with good old Times, but again, sans serif fonts are better for body text.

To solve this, consider implementing a font stack, a list of backup fonts in your CSS file that the browser will render if your first font choice fails to do so. Put a couple of standard fonts in your font stack to ensure visitors will always see the most suitable style of text.

4.2.6 Size your text appropriately

Web designers specify font size with pixels (px) rather than points (pt). This is because a pixel is a standardized unit online, whereas a font point is not — two

people reading 12pt text on the same website might see different things depending on their devices or web browsers.

A common practice is to set all website text to a minimum size of 16px. This is roughly the size of body text in printed media, and is the smallest font that most people can read without needing to zoom in. Of course, you can and should increase and vary the size of your text to further assist readers and establish hierarchy, but don't go overboard with massive fonts either.

On the subject of hierarchy, headings should always be larger than the body text and decrease in size by **H1, H2, H3**, etc. This helps readers scan your pages for the target content. You might also elect to add varying weights to your headings for a greater contrast with the body text.

4.2.7 Don't use all caps

This is more of a straightforward rule —" all caps" is unnecessary in nearly all contexts outside of the decorative text, branding, and the occasional set of headings.

If you want to emphasize the body or heading text, bold it. This lends the same effect while being more readable and visually pleasing.

4.2.8 Use colors carefully and intentionally

A common pain point for web users is a bad pairing of text color and background color, in which the two do not contrast enough to maintain legibility. Refrain from layering text over a background with similar colors, and be very cautious when placing text atop images.

More precisely, the Web Content Accessibility Guidelines (WCAG) recommend a contrast ratio of at least 4.5:1 for most text, and 3:1 for large, bolded text. Use a free tool like this one to check the contrast between your front and background color. Alternatively, you can't go wrong with black or dark text on a white background, at least in terms of legibility.

Aside from contrast, be mindful of your choice of text color as well. It's best to keep your body text one uniform color with the exception of hyperlinks, which should contrast with the rest of the text. Avoid using blue as the default color in your text, since it implies a hyperlink.

Also, refrain from using red and green as a visual cue in your text, as this won't be apparent to individuals with red-green color blindness. In fact, color alone should not be used to distinguish one piece of text from the rest. Combine color with other styling (like bolding, italics, or underlining) to emphasize a text snippet.

4.2.9 Stay between around 40 and 80 characters per line

Humans are picky readers — we favor lines of text that fall between 40 and 80 characters. Anything less forces our eyes to move to the next line too frequently, which distracts us. On the other end, any line length greater will bore readers, cause discomfort, and required more effort to find the start of a new line as the eye travels back to the left side of the text block.

These parameters provide some wiggle room for different page layouts and mobile-responsive designs. But, if you can, aim for the sweet spot of **60-70** characters per line. Your eyes will thank you.

4.2.10 Provide sufficient spacing between lines

Proper whitespace ensures that readers can easily follow single lines of text and return to the next line after a line break. Accessibility frameworks tend to allot vertical space based on the font size of the respective text.

For body text, start with a spacing of **1.5**, which means that the leading is **50%** the height of the text line. For headings, this distance should be slightly greater. For between paragraphs, start with a spacing of **2.5** and adjust up or down from there.

4.2.11 Eliminate text animations

Yes, animations do grab the reader's attention, but few things are worse for readability than flashing or moving text. If you've ever tried to read a note that someone held up to you, you'll understand why — it takes work to stabilize it in our brains.

What's worse, many visitors will think of this text as an unnecessary inconvenience, a gimmick, and/or an ad. Flashing image may also trigger photosensitive seizures.

The sole exception to this final rule is entrance or exit effects. These can be a fun way to build an experience for a visitor as they scroll. But, once the text appears, it should stay static.

4.2.12 Testing Your Text

Armed with these guidelines and typography know-how, you're ready to start experimenting with typefaces, fonts, and styles in search of the perfect reading experience.

However, there's one last catch: You could follow all of the guidelines above, and still neglect some aspects of typography that an average visitor will notice instantly. That's why the final phase of any text design iteration is thorough user testing.

By following these typography guidelines, web designers can create visually appealing and readable typographic layouts that enhance the user experience and effectively communicate the website's content.

4.3 Typography Guidelines for Print Design

Typography guidelines for print design are a set of principles and best practices that help designers create effective and visually appealing typographic layouts for printed materials. These guidelines ensure that the typography is readable, aesthetically pleasing, and aligned with the overall design concept. Here are some key typography guidelines for print design:

- **Font selection**: Choose fonts that are appropriate for the purpose and tone of the printed piece. Consider the target audience and the content being communicated. Select fonts that are legible and have clear letterforms. Serif and sans-serif fonts are commonly used in print design, but other font categories can be considered based on the design concept.

- **Readability:** Prioritize readability by selecting fonts with appropriate spacing and letterforms that are easy to distinguish. Avoid overly decorative or elaborate fonts that may hinder readability, especially for longer passages of text. Ensure sufficient contrast between the text and the background for legibility.

- **Font size:** Use an appropriate font size that ensures readability. Headlines and titles should be larger and more prominent, while body text should be comfortable to read at a typical reading distance. Consider the size of the printed piece and the viewing conditions when determining font sizes.

- **Hierarchy and formatting**: Establish a clear typographic hierarchy by using different font sizes, weights, and styles to differentiate headings, subheadings, and body text. Use formatting techniques such as bold, italics, underline, or all-caps to add emphasis and guide the reader's attention.

- **Alignment and spacing:** Align text elements properly to create a visually balanced layout. Use appropriate line spacing (leading) to ensure readability and avoid overcrowding or tightly spaced text. Adjust paragraph spacing and indentation to provide visual separation between different sections of text.

- **Consistency**: Maintain consistency in typography throughout the printed materials. Use the same font families, weights, and styles across different pages and sections to establish

a cohesive visual identity. Consistency also applies to text formatting, such as bullet points, numbering, and indentation.

- **Color and contrast:** Consider the use of color in typography to enhance the visual appeal of the printed piece. Ensure sufficient contrast between the text and the background color to maintain readability. Select colors that align with the overall design concept and brand identity.

- **Kerning and tracking:** Adjust letter spacing (kerning) and word spacing (tracking) to achieve optimal visual spacing between characters and words. Fine-tune the spacing to create balanced and visually pleasing typography. Avoid excessive spacing that can hinder readability.

- **Typography and imagery:** Integrate typography with imagery in a harmonious way. Ensure that text does not compete with or obscure important visual elements. Pay attention to the placement and interaction between text and images to create a cohesive and visually pleasing design.

- **Proofreading:** Always proofread the text before finalizing the print design. Check for spelling and grammar errors, consistency in typography, and proper alignment. Ensure that all text is accurate and conveys the intended message.

Following these typography guidelines for print design helps designers create visually appealing and readable typographic layouts that effectively communicate the message and enhance the overall aesthetic of printed materials.

Self-Check Sheet 2: Utilize the fundamentals of typography

Questionnaire:

1. What is typography in web design?
   Answer:

2. Why is typography important in web design?
   Answer:

3. What are some considerations for font selection in web design?
   Answer:

4. What does font styling involve in typography?
   Answer:

5. What is the recommended line length for optimal readability in web typography?
   Answer:

6. What is the role of hierarchy in typography?
   Answer:

7. How does responsive typography adapt to different devices?
   Answer:

8. What is the significance of whitespace in typography?
   Answer:

9. What are some accessibility considerations in web typography?
   Answer:

10. Why is consistency important in typography for web design?
    Answer:

Answer Key 2: Utilize the fundamentals of typography

1. What is typography in web design?
   Answer: Typography in web design refers to the art and technique of arranging and presenting text on a website.

2. Why is typography important in web design?
   Answer: Typography is important in web design as it enhances the overall aesthetics, readability, and user experience of a website.

3. What are some considerations for font selection in web design?
   Answer: Legibility, readability, target audience, website purpose, brand identity, and style are some considerations for font selection in web design

4. What does font styling involve in typography?
   Answer: Font styling involves adjusting font size, weight (boldness), style (italic, etc.), and spacing for effective visual presentation

5. What is the recommended line length for optimal readability in web typography?
   Answer: The recommended line length is typically between 60-75 characters for optimal readability in web typography.

6. What is the role of hierarchy in typography?
   Answer: The recommended line length is typically between 60-75 characters for optimal readability in web typography.

7. How does responsive typography adapt to different devices?
   Answer: The recommended line length is typically between 60-75 characters for optimal readability in web typography.

8. What is the significance of whitespace in typography?
   Answer: Whitespace, or empty space, around text elements helps to create visual separation, improve readability, and enhance overall design aesthetics.

9. What are some accessibility considerations in web typography?
   Answer: Accessibility considerations include providing sufficient color contrast, scalable font sizes, and alternative text for images to ensure inclusive typography.

10. Why is consistency important in typography for web design?
    Answer: Consistency in typography helps to establish a cohesive visual identity, reinforce brand recognition, and provide a seamless user experience across the website.

Task Sheet 2: Create an HTML Form

Objectives: The objective of this task is to create a basic HTML form that allows users to input information and submit it. By completing this task, you will gain a better understanding of HTML form elements and their attributes, as well as how to handle form submission.

Working Procedure:

1. Set up the project:
    - Create a new folder on your computer for this project.
    - Open a code editor of your choice (e.g., Visual Studio Code, Sublime Text, etc.).
    - Create a new HTML file and save it as "index.html" within the project folder.
2. Define the HTML structure:
    - Start by creating the HTML boilerplate by typing **<!DOCTYPE html>** and **<html>** tags.
    - Inside the **<html>** tags, add the **<head>** and **<body>** sections.
3. Create the form:
    - Within the **<body>** section, add the **<form>** element to create the form.
    - Set the "action" attribute of the form to "#" for now. This will prevent the form from submitting to a server during testing.
    - Set the "method" attribute to "post" or "get," depending on your preferences. For this task, either will work fine, but "post" is more secure.
4. Add form elements: a. Inside the **<form>** tags, add various form elements using appropriate tags:
    - **<label>:** To add labels for each input field.
    - **<input>:** For text input fields, checkboxes, radio buttons, etc.
    - **<textarea>:** For multi-line text input.
    - **<select>** and **<option>:** For dropdown menus.
5. Set input field attributes:
    - For each **<input>** element, set the "type," "name," and "id" attributes.
    - Assign a unique "id" for each form element to associate labels with their respective input fields.
    - Use the "required" attribute for mandatory fields if needed.

6. Add a submit button:
    - Include a submit button within the form using the **<input>** element with "type" set to "submit."
    - Customize the button text as needed.
7. Test the form:
    - Save your HTML file.
    - Open the file in your web browser.
    - Fill out the form fields and submit the form to see the default behavior.
8. Form validation (optional):
    - Add validation to ensure the correct format of user inputs (e.g., valid email address, minimum password length)
    - Use JavaScript to handle form validation. For example, you can use the **onsubmit** attribute on the **<form>** element to call a validation function.
9. Enhance the styling (optional):
    - Add CSS to style the form elements and improve the overall appearance.
    - Make the form responsive to different screen sizes.
10. Deploy the form (optional):
    - If you have access to a web server, you can upload the "index.html" file along with the associated CSS and JavaScript (if any) to make the form accessible online.

Learning Outcome 3: Create an HTML page

Assessment Criteria:	1. Most common HTML tags are used 2. Most common entities & attributes are used 3. HTML multicolumn layout is implemented. 4. HTML Graphics are used. 5. HTML Media is used. 6. HTML page is saved.
Conditions and Resources	1. Applicable tools, utensils, and equipment as prescribed by competency standard. 2. Supply materials 3. Relevant ingredients 4. CBLM related to the learning outcome. 5. Instructions, job sheets, activity sheets, and standard operating procedures 6. Personal protective equipment 7. Module/reference
Contents	1. Most Common HTML Tags 2. Entities and Attributes in HTML 3. HTML Multicolumn Layouts 4. Webpages with HTML Graphics 5. Media in HTML 6. HTML Pages
Training Methods	1. Discussion 2. Presentation 3. Demonstration 4. Guided Practice 5. Individual Practice 6. Project Work 7. Problem Solving 8. Brainstorming
Learning Materials	1. CBLM 2. Handouts 3. Books, Manuals 4. Module/ Reference 5. Paper 6. Pen
Assessment Methods	1. Written Test 2. Demonstration 3. Oral Questioning

Learning Experience 3: Create an HTML page

You must perform the learning steps below to achieve the objectives stated in this learning guide. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Steps	Resources specific instructions
1. Students will ask the instructor about work with HTML	1. The instructor will provide the learning materials for` create an HTML page.
2. Read the **Information sheet/s**	2. Information Sheet No: 3. Create an HTML page.
3. Complete the **Self-Checks & Answer key sheets.**	3. Self-Check No: 3 Create an HTML page Answer key No. 3 Create an HTML page
4. Read the **Job/ Task sheet** and **Specification Sheet**	4. Job/ task sheet and specification sheet: ▪ Task Sheet No: 3 Create a Simple HTML Page

Information Sheet 3: Create an HTML page

Learning Objective: After completion of this information sheet, the learners will be able to interpret the following contents:

3.1 Mastering the Most Common HTML Tags
3.2 Exploring Entities and Attributes in HTML
3.3 Implementing HTML Multicolumn Layouts
3.4 Enhancing Webpages with HTML Graphics
3.5 Incorporating Media in HTML
3.6 Saving and Publishing HTML Pages

3.1 Most Common HTML Tags

HTML Tags: HTML tags are the building blocks of an HTML document. They are used to define the structure and content of a web page. HTML tags are enclosed in angle brackets (< >) and usually come in pairs, with an opening tag and a closing tag. The opening tag contains the name of the tag, while the closing tag is similar but with a forward slash (/) before the tag name.

Here are some of the most common HTML tags:

- **<html>** : Represents the root of an HTML document.
- **<head>** : Contains metadata about the HTML document.
- **<title>** : Defines the title of the document, displayed in the browser's title bar or tab.
- <**body>** : Encloses the main content of the HTML document.
- **<h1>** to **<h6>** : Represents headings, with <h1> being the highest and <h6> the lowest.
- **<p>** : Defines a paragraph.
- **<a>** : Creates a hyperlink to another webpage or resource.
- **<img>** : Inserts an image into the HTML document.
- **<ul>** : Defines an unordered (bulleted) list.
- **<ol>** : Defines an ordered (numbered) list.
- **<li>** : Represents a list item within <ul> or <ol>.
- ** ** : Defines a division or a section in an HTML document.
- **** : Defines an inline section within a document.
- **<table>** : Represents a table.
- **<tr>** : Defines a table row.

- **<td>** : Defines a table cell.
- **<form>** : Creates a form for user input.
- **<input>** : Defines an input field within a form.
- **<button>** : Creates a clickable button.
- **
** : Inserts a line break.
- **<header>** : Defines the header section of a document or a section.
- **<footer>** : Defines the footer section of a document or a section.
- **<nav>** : Represents a navigation menu.
- **<section>** : Defines a section in a document.
- **<article>** : Represents an independent piece of content within a document.
- **<aside>** : Defines content that is tangentially related to the main content.
- **<blockquote>** : Indicates a section of quoted content.
- **<cite>** : Defines the title of a work being cited.
- **<code>** : Represents a piece of computer code.
- **<pre>** : Preserves whitespace formatting within its content.
- **** : Indicates emphasis on text.
- **** : Indicates strong importance or seriousness.
- **<mark>** : Highlights or marks a specific section of text.
- **<del>** : Represents deleted or struck-out text.
- **<ins>** : Represents inserted or underlined text.
- **<sub>** : Specifies subscripted text.
- **<sup>** : Specifies superscripted text.
- **<iframe>** : Embeds another HTML document within the current document.
- **<video>** : Embeds a video into the document.
- **<audio>** : Embeds audio content into the document.
- **<label>** : Represents a label for an <input> element.
- **<select>** : Creates a drop-down list.
- **<option>** : Represents an option within a <select> element.
- **<textarea>** : Creates a multi-line text input field.
- **<button>** : Defines a clickable button.
- **<fieldset>** : Groups related form elements together.
- **<legend>** : Provides a caption for a <fieldset> element.
- **<datalist>** : Specifies a list of pre-defined options for an <input> element.
- **<progress>** : Represents the progress of a task or process.

- **<meter>** **:** Defines a scalar measurement within a known range.
- **<hr>** **:** Inserts a horizontal rule (a thematic break) in the document.
- **<abbr>** **:** Represents an abbreviation or acronym.
- **<address>** **:** Represents contact information for the author or owner of a document.
- **<time>** **:** Specifies a date or time value.
- **<iframe>** **:** Embeds another web page or external content within the document.
- **<canvas>** **:** Creates a drawing canvas for graphics and animations.
- **<svg>** **:** Embeds scalable vector graphics in an HTML document.
- **<script>** **:** Inserts or references an external JavaScript code or defines inline script.
- **<style>** **:** Contains CSS styles that can be applied to the document.
- **<meta>** **:** Provides metadata about the HTML document.

3.2 Entities and Attributes in HTML

In HTML, entities and attributes are essential components used to enhance the structure, presentation, and functionality of web pages.

Entities: Entities are special codes used to represent reserved characters or symbols in HTML. They help ensure proper rendering and interpretation of these characters by web browsers. For example:

- **<** represents the less-than symbol (<). Example: 4 < 5 will display as "4 < 5" in the browser.
- **>** represents the greater-than symbol (>). Example: 7 > 3 will display as "7 > 3" in the browser.
- **&** represents the ampersand symbol (&). Example: AT&T will display as "AT&T" in the browser.
- **"** represents the double quotation mark ("). Example: "Hello World" will display as ""Hello World"" in the browser.
- **©** represents the copyright symbol (©). Example: © 2023 MyCompany will display the copyright symbol followed by "2023 MyCompany" in the browser.
- **€:** Represents the Euro symbol (€). Example: Price: €10 will display as "Price: €10" in the browser.

- **&pound:** Represents the Pound sterling symbol (£). Example: Price: £20 will display as "Price: £20" in the browser.
- **&yen:** Represents the Yen symbol (¥). Example: Price: ¥1000 will display as "Price: ¥1000" in the browser.
- **&reg:** Represents the registered trademark symbol (®). Example: Product: MyBrand® will display as "Product: MyBrand®" in the browser.
- **&trade:** Represents the trademark symbol (™). Example: Product: MyProduct™ will display as "Product: MyProduct™" in the browser.
- **&ndash:** Represents an en dash (–). Example: Pages: 100–150 will display as "Pages: 100–150" in the browser.
- **&mdash:** Represents an em dash (—). Example: Author: John Smith—New York Times will display as "Author: John Smith—New York Times" in the browser.
- **&hellip:** Represents an ellipsis (...). Example: Read more… will display as "Read more..." in the browser.

Attributes: Attributes provide additional information or instructions to HTML elements, modifying their behavior, appearance, or functionality. They are specified within the opening tag of an element as name-value pairs. Here are a few examples:

- **class:** Specifies a class name to an element for styling or JavaScript manipulation. Example: ... .
- **id:** Defines a unique identifier for an element, often used for targeting with CSS or JavaScript. Example: <h1 id="main-heading">Hello World</h1>.
- **src:** Specifies the source URL of an external resource, such as an image or script. Example: <img src="image.jpg" alt="Image">.
- **href:** Defines the URL or destination of a hyperlink. Example: Visit Example.
- **alt:** Provides alternative text for images, displayed if the image cannot be loaded. Example: <img src="image.jpg" alt="Description of the image">.
- **disabled:** Disables an input or button element, preventing user interaction. Example: <input type="text" disabled>.
- **required:** Specifies that an input field must be filled out before submitting a form. Example: <input type="text" required>.
- **accesskey:** Specifies a keyboard shortcut for an element.

- **align:** Specifies the alignment of an element (e.g., <img align="left">).
- **autocomplete:** Controls whether an input field should have autocompleted suggestions.
- **autofocus:** Specifies that an input field should automatically have focus when the page loads.
- **charset:** Defines the character encoding for the document.
- **checked:** Sets a checkbox or radio button as selected by default.
- **class:** Specifies one or more class names for an element.
- **contenteditable:** Specifies whether the content of an element is editable by the user.
- **disabled:** Disables an input or button element, preventing user interaction.
- **href:** Defines the URL or destination of a hyperlink.
- **id:** Defines a unique identifier for an element.
- **placeholder:** Specifies a short hint or example text within an input field.
- **readonly:** Makes an input field read-only, preventing user input.
- **required:** Specifies that an input field must be filled out before submitting a form.
- **src:** Specifies the source URL of an external resource, such as an image or script.
- **style:** Applies inline CSS styles to an element.
- **target:** Specifies where to open the linked document.
- **title:** Specifies extra information about an element (often displayed as a tooltip).
- **type:** Specifies the type of an input element (e.g., type="text").
- **value:** Specifies the initial value of an input element.

These examples cover both entities and attributes in HTML. Entities ensure proper rendering of special characters, while attributes modify the behavior, appearance, or functionality of HTML elements. Feel free to explore and use these entities and attributes based on your specific requirements in web development.

3.3 HTML Multicolumn Layouts

HTML multicolumn layouts allow you to divide the content of a web page into multiple columns, similar to a newspaper or magazine layout. This feature is particularly useful for displaying large amounts of text or creating a visually appealing presentation of content. Here's how you can implement multicolumn layouts in HTML:

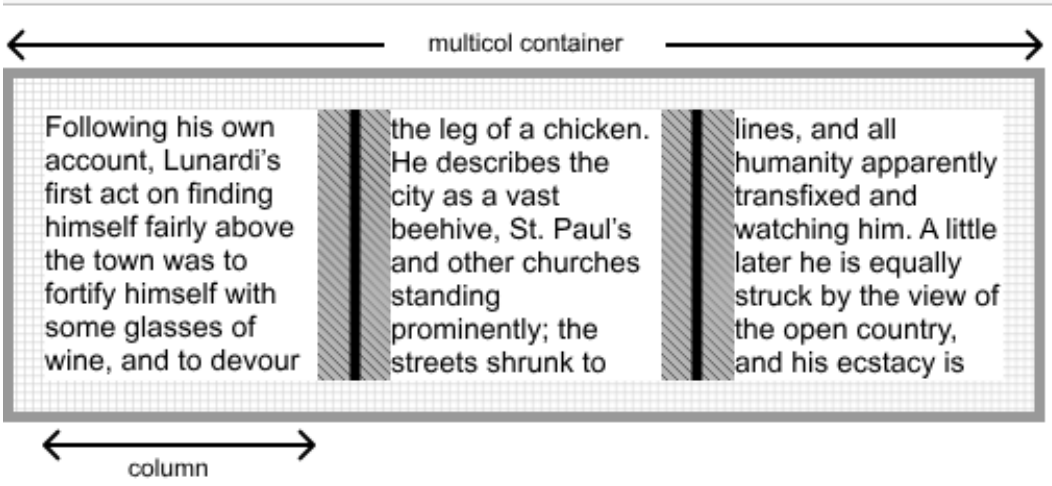

Figure 4: HTML Multicolumn Layouts

3.3.1 Using the CSS column-count property:

- The column-count property specifies the number of columns you want to divide the content into.
- You can apply this property to a container element, such as a , to create columns within that element.

Example:

```
<style>
.multicolumn {
column-count: 3;
}
</style>
<div class="multicolumn">
<p>Column 1 content</p>
<p>Column 2 content</p>
<p>Column 3 content</p>
</div>
```

- This will create a multicolumn layout with three columns.

3.3.2 Using the CSS column-count property:

- The column-count property specifies the number of columns you want to divide the content into.
- You can apply this property to a container element, such as a , to create columns within that element.

Example:

```
<style>
.multicolumn {
column-count: 3;
}
</style>
<div class="multicolumn">
<p>Column 1 content</p>
<p>Column 2 content</p>
<p>Column 3 content</p>
</div>
```

- This will create a multicolumn layout with three columns.

3.3.3 Using the CSS column-width property:

- The column-width property specifies the width of each column in the multicolumn layout.

Example:

```
<style>
.multicolumn {
column-width: 200px;
}
</style>
<div class="multicolumn">
<p>Column 1 content</p>
<p>Column 2 content</p>
<p>Column 3 content</p>
</div>
```

- This will create a multicolumn layout with columns of 200 pixels width each.

3.3.4 Using the CSS column-gap property:

- The column-gap property specifies the gap or spacing between columns in the multicolumn layout.

Example:

```
<style>
.multicolumn {
column-count: 2;
column-gap: 20px;
}
</style>
<div class="multicolumn">
<p>Column 1 content</p>
<p>Column 2 content</p>
<p>Column 3 content</p>
<p>Column 4 content</p>
</div>
```

- This will create a multicolumn layout with two columns and a gap of 20 pixels between them.

These are just a few examples of creating multicolumn layouts using CSS properties. You can further customize the layout by combining different properties and using responsive techniques to adapt the layout for different screen sizes.

3.3.5 CSS column-count property:

- The column-count property sets the number of columns in the multicolumn layout.
- It accepts an integer value to specify the number of columns.
- Example: column-count: 3; will create a multicolumn layout with three columns.
- The content is distributed evenly across the columns, and when the content exceeds the available space, it flows into the next column.

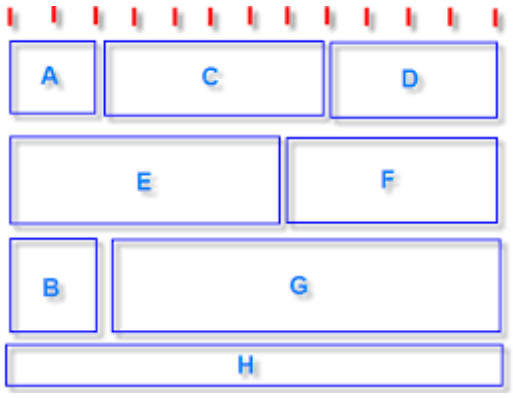

Figure 5: CSS column-count property

3.3.6 CSS column-width property:

- The column-width property sets the width of each column in the multicolumn layout.
- It accepts a length value, such as pixels (px) or percentages (%), to specify the width.
- Example: column-width: 200px; will create columns with a width of 200 pixels each.
- The number of columns may vary based on the available width, and the column width adjusts accordingly.

3.3.7 CSS column-gap property:

- The column-gap property sets the gap or spacing between columns in the multicolumn layout.
- It accepts a length value to specify the gap, such as pixels (px) or percentages (%).
- Example: column-gap: 20px; will create a 20-pixel gap between columns.
- The gap is applied evenly between columns, providing visual separation.

3.3.8 CSS column-rule property:

- The column-rule property allows you to add a vertical rule or line between columns in the multicolumn layout.
- It accepts values for the width, style, and color of the rule.
- Example: column-rule: 1px solid black; will add a 1-pixel wide solid black line between columns.
- The rule is applied vertically between columns, providing a visual divider.

3.3.9 CSS break-inside property:

- The break-inside property specifies whether or not an element can be broken between columns in a multicolumn layout.
- By default, elements can be broken between columns (break-inside: auto), but you can set it to avoid to prevent column breaks.
- Example: break-inside: avoid; will prevent an element from being split between columns.
- This property is useful for maintaining the integrity of elements like images or block-level content.

3.3.10 Responsive multicolumn layouts:

- You can make multicolumn layouts responsive by using CSS media queries to adapt the layout based on screen sizes.
- For example, you can change the number of columns or adjust the column width at different breakpoints.

Examples:

```
<style>
.multicolumn {
column-count: 3;
}
@media screen and (max-width: 768px) {
.multicolumn {
column-count: 2;
}
}
@media screen and (max-width: 480px) {
.multicolumn {
column-count: 1;
}
}
</style>
<div class="multicolumn">
<!-- Content goes here -->
</div>
```

- In this example, the multicolumn layout adapts to three columns for larger screens, two columns for medium screens, and one column for smaller screens.

These features provide flexibility in creating multicolumn layouts in HTML. You can experiment with different combinations of properties, adjust the number of columns, widths, and gaps, and use responsive techniques to achieve the desired layout for your web page.

3.4 HTML Graphics

HTML graphics refer to the use of various elements and techniques within HTML to display visual content on a web page. Graphics in HTML allow you to incorporate images, icons, shapes, and other visual elements to enhance the visual appeal and interactivity of your web pages. Here are some common HTML graphics techniques:

- **Images:**

- The <img> element is used to display images on a web page.
- You specify the image source using the src attribute, and you can provide alternate text with the alt attribute.

Example: `<img src="image.jpg" alt="Description of the image">`

- **Icons:**

- Icons are small, symbolic images commonly used to represent actions, categories, or features.
- Icon libraries like Font Awesome or Material Icons provide a wide range of icons that can be easily integrated into HTML using <i> or elements with appropriate classes.

Example: `<i class="fas fa-heart"></i>` for a heart icon using Font Awesome.

- **SVG:**

- Scalable Vector Graphics (SVG) is a markup language for creating vector-based graphics in HTML.
- SVG graphics are defined using XML tags and provide crisp and scalable images that can be styled and animated.
- You can embed SVG directly into HTML using the <svg> element or reference an external SVG file using the <img> element.

Example

```
<svg xmlns="http://www.w3.org/2000/svg" width="100" height="100">
<circle cx="50" cy="50" r="40" fill="red" />
</svg>
```

Canvas:

- The <canvas> element provides a drawing surface in HTML for dynamically generated graphics and animations.
- JavaScript is used to interact with the canvas and draw shapes, images, and animations.

```
<canvas id="myCanvas" width="400" height="200"></canvas>
<script>
const canvas = document.getElementById('myCanvas');
const context = canvas.getContext('2d');
context.fillStyle = 'blue';
context.fillRect(50, 50, 100, 100);
</script>
```

CSS for Graphics:

- CSS can be used to style and enhance graphics within HTML.
- You can apply CSS properties like colors, gradients, shadows, and transformations to elements, including images, icons, and SVG graphics.
- CSS transitions and animations can also be used to create visually appealing effects.

```
<style>
.my-icon {
color: red;
font-size: 24px;
text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
transform: rotate(45deg);
}
</style>
<i class="my-icon">+</i>
```

These are just some of the techniques you can use to incorporate graphics into your HTML pages. By leveraging images, icons, SVG, canvas, and CSS, you can create visually engaging and interactive experiences for your website visitors.

3.5 Media in HTML

Media in HTML refers to the inclusion and presentation of various types of media content within a web page. HTML provides specific elements and attributes to embed and control media elements such as images, audio, video, and interactive media. Here are some key elements and attributes used for media in HTML:

- **Images:**

- The <img> element is used to embed images in HTML.
- The src attribute specifies the source URL of the image.

Example: *<img src="image.jpg" alt="Description of the image">*

- **Audio:**
- The <audio> element allows you to embed audio content in HTML.
- The src attribute specifies the source URL of the audio file.

Example: *<audio src="audio.mp3" controls></audio>*

- **Video:**

- The <video> element is used to embed video content in HTML.
- The src attribute specifies the source URL of the video file.
- Example: <video src="video.mp4" controls></video>
- The controls attribute adds playback controls (play, pause, volume, etc.) to the video player.

- **Embedded Content:**

- The <iframe> element allows you to embed external content such as maps, documents, or web pages within an HTML document.
- The src attribute specifies the source URL of the embedded content.

Example: *<iframe src="https://www.example.com" width="500" height="300"></iframe>*

- **Canvas:**

- The <canvas> element provides a drawing surface where you can use JavaScript to create and manipulate graphics, animations, and interactive content.
- JavaScript is used to interact with the canvas and draw shapes, images, and animations.

Example:

<canvas id="myCanvas" width="400" height="200"></canvas>

<script>

const canvas = document.getElementById('myCanvas');

```
const context = canvas.getContext('2d');
context.fillStyle = 'blue';
context.fillRect(50, 50, 100, 100);
</script>
```

- **Multimedia Controls:**

- Various attributes and options can be used to control the behavior and appearance of media elements.
- For example, the controls attribute can be added to the <audio> and <video> elements to display playback controls.
- Other attributes like autoplay, loop, preload, and poster can be used to define autoplay, looping, preloading, and poster image options.
- Media Events and APIs:
- HTML provides JavaScript APIs and events to interact with and control media elements programmatically.
- You can use these APIs to play, pause, seek, control volume, and handle other media-related interactions.
- For example, the play(), pause(), currentTime, and volume properties can be used to control media playback programmatically.

- **HTML Video**

To embed a video in HTML, you can use the <video> element. Here's an example:

```
<video width="640" height="360" controls>
<source src="video.mp4" type="video/mp4">
<source src="video.webm" type="video/webm">
Your browser does not support the video tag.
</video>
```

In the above example:

- The width and height attributes specify the dimensions of the video player.
- The controls attribute adds playback controls (play, pause, volume, etc.) to the video player.
- The <source> element is used to specify multiple video sources. Different browsers support different video formats, so providing multiple sources increases compatibility. In the example, we have provided two sources: one in MP4 format and another in WebM format.
- The text "Your browser does not support the video tag." is displayed if the browser does not support the <video> element.

You need to provide the actual source file paths for the src attribute of each <source> element. In the example, replace "video.mp4" and "video.webm" with the paths to your video files.

Additionally, you can include more <source> elements with different video formats if needed, such as for formats like Ogg or MOV.

Remember to adjust the width, height, and source file paths based on your specific requirements.

- **HTML Audio**

  To embed audio in HTML, you can use the <audio> element. Here's an example:

```
<audio controls>
<source src="audio.mp3" type="audio/mpeg">
<source src="audio.ogg" type="audio/ogg">
Your browser does not support the audio element.
</audio>
```

In the above example:

- The controls attribute adds playback controls (play, pause, volume, etc.) to the audio player.
- The <source> element is used to specify multiple audio sources. Different browsers support different audio formats, so providing multiple sources increases compatibility. In the example, we have provided two sources: one in MP3 format and another in Ogg format.
- The text "Your browser does not support the audio element." is displayed if the browser does not support the <audio> element.

You need to provide the actual source file paths for the src attribute of each <source> element. In the example, replace "audio.mp3" and "audio.ogg" with the paths to your audio files.

You can include additional <source> elements with different audio formats if needed, such as for formats like WAV or AAC.

Remember to adjust the source file paths based on your specific audio files.

- **HTML Plug-ins**

  In HTML, plugins are additional software components that can be embedded within a web page to extend its functionality or provide multimedia capabilities. Historically, plugins were commonly used to add interactive content, multimedia playback, or support for specific technologies that were not natively supported by web browsers. However, with the advancements in web standards and the adoption of HTML5, the need for plugins has decreased significantly.

Here are a few examples of popular plugins that were widely used in the past:

1. **Adobe Flash Player:**

- Flash Player was a popular plugin used for playing multimedia content, such as videos, animations, and interactive applications.
- However, support for Flash has been discontinued, and most web browsers no longer support it due to security and performance concerns.

2. **Java Applets:**

- Java Applets were small Java programs that could be embedded in web pages to provide interactive features and functionality.
- Similar to Flash, the usage of Java Applets has significantly declined, and support for them has been removed from most web browsers.

3. **Silverlight:**

- Silverlight was a plugin developed by Microsoft that provided rich media and interactive applications.
- However, support for Silverlight has also been discontinued, and it is no longer widely used.

It's important to note that the use of plugins has become less prevalent due to advancements in web technologies. Modern web standards, such as HTML5, CSS3, and JavaScript, provide native capabilities for handling multimedia, animations, interactivity, and other features. This has allowed for more seamless and standardized web experiences without the need for additional plugins.

Instead of relying on plugins, it is recommended to use HTML5 elements (such as <video> and <audio>) and utilize JavaScript frameworks or libraries to enhance interactivity and functionality.

Overall, the use of plugins in modern web development has significantly decreased, and it's best to leverage the native capabilities of web technologies to build robust and compatible web experiences.

HTML YouTube

To embed a YouTube video in HTML, you can use the <iframe> element with the source URL of the video. Here's an example:

```
<iframe width="560" height="315" src="https://www.youtube.com/embed/{VIDEO_ID}"
frameborder="0" allowfullscreen></iframe>
```

Replace {VIDEO_ID} with the actual ID of the YouTube video you want to embed. The video ID is typically a combination of letters and numbers found in the YouTube video URL.

For example, if the YouTube video URL is "https://www.youtube.com/watch?v=dQw4w9WgXcQ", the video ID would be "dQw4w9WgXcQ". You would replace {VIDEO_ID} in the above code with "dQw4w9WgXcQ" to embed the video.

Here's the updated code:

```
<iframe width="560" height="315" src="https://www.youtube.com/embed/dQw4w9WgXcQ" frameborder="0" allowfullscreen></iframe>
```

You can adjust the width and height values to customize the size of the video player according to your needs. The allow full-screen attribute allows the video to be played in full-screen mode.

Remember to replace {VIDEO_ID} with the actual video ID from the YouTube URL to embed the desired YouTube video in your HTML page.

3.6 Saving and Publishing HTML Pages

Saving and publishing HTML pages involves the process of storing the HTML code and making it accessible to others on the internet. Here's a step-by-step guide on how to save and publish HTML pages:

1. **Saving the HTML Page:**
   - Create or edit your HTML page using a text editor or an integrated development environment (IDE).
   - Save the file with an .html extension. For example, you can save it as index.html.
   - Choose a relevant file name that reflects the content of your webpage.

2. **Local Preview:**
   - Open the saved HTML file in a web browser to preview and verify that the page appears as intended.
   - Double-click on the HTML file, and it will open in the default web browser.

3. **Publishing the HTML Page:**
   - To make your HTML page accessible to others on the internet, you need to publish it to a web server or hosting service.
   - Select a web hosting service or set up your own web server to host your HTML files. Some popular hosting providers include GitHub Pages, Netlify, or traditional web hosting services.

- Follow the instructions provided by the hosting service to upload your HTML file to the server. This may involve using FTP (File Transfer Protocol) or a web-based file manager provided by the hosting service.

4. **Accessing the Published HTML Page:**

- Once you have published your HTML page, it can be accessed by others using its URL.
- The URL typically consists of the domain name or hosting provider's address followed by the path to your HTML file. For example, if your domain is "example.com" and your HTML file is named "index.html", the URL would be "http://example.com/index.html".
- Share the URL with others to allow them to access your published HTML page.

Note: If you don't have a web hosting service or want to share your HTML page locally, you can compress the entire folder containing the HTML file and any associated assets (such as CSS stylesheets or images) into a ZIP file. Then, others can extract the ZIP file and open the HTML file locally on their machines.

Remember to regularly update and republish your HTML pages as needed to keep them up to date and ensure any changes or improvements are reflected on the published version.

Self-Check Sheet 3: Create an HTML page

1. What is an HTML tag?

   Answer:

2. What is the basic structure of an HTML document?

   Answer:

3. How do you create a hyperlink in HTML?

   Answer:

4. What is the purpose of the <img> tag?

   Answer:

5. What is the purpose of the <table> tag?

   Answer:

6. How do you add comments in HTML?

   Answer:

7. What is the purpose of the <form> tag?

   Answer:

8. How do you create a numbered list in HTML?

   Answer:

9. How do you add styling to HTML elements?

   Answer:

10. What is the purpose of the tag?

Answer Key 3: Create an HTML page

1. What is an HTML tag?

   Answer: The <img> tag is used to display images in HTML

2. What is the basic structure of an HTML document?

   Answer: The <table> tag is used to create tables in HTML

3. How do you create a hyperlink in HTML?

   Answer: Comments in HTML can be added using the <!-- --> syntax.

4. What is the purpose of the <img> tag?

   Answer: The <form> tag is used to create interactive forms in HTML

5. What is the purpose of the <table> tag?

   Answer: By using the <ol> tag with nested <li> tags, you can create a numbered list in HTML.

6. How do you add comments in HTML?

   Answer: Styling in HTML can be added using CSS (Cascading Style Sheets) either internally or externally.

7. What is the purpose of the <form> tag?

   Answer: The tag is a container element used to group and style other HTML elements

8. How do you create a numbered list in HTML?

   Answer: By using the
 tag, you can create a line break in HTML

9. How do you add styling to HTML elements?

   Answer: The <head> section contains metadata and defines the title of the HTML document

10. What is the purpose of the tag?

    Answer: By using the <video> tag with appropriate source elements, you can display videos in HTML.

Task Sheet 3- Create a Simple HTML Page

Objectives: The objective of this task is to create a simple HTML page that serves as a foundation for building web content. By completing this task, you will gain a better understanding of basic HTML structure, tags, and how to include content in an HTML page.

Working Procedure:

1. **Set up the project:**
   - Create a new folder on your computer for this project.
   - Open a code editor of your choice (e.g., Visual Studio Code, Sublime Text, etc.).
   - Create a new HTML file and save it as "index.html" within the project folder.
2. **Define the HTML structure:**
   - Start by creating the HTML boilerplate by typing <!DOCTYPE html> and <html> tags.
   - Inside the <html> tags, add the <head> and <body> sections.
3. **Set the document title:**
   - Within the <head> section, add the <title> element to specify the title of your web page.
   - Choose a descriptive title that reflects the content of your page.
4. **Add metadata (optional):**
   - Within the <head> section, you can include additional metadata like character encoding, author, description, etc., using <meta> tags.
5. **Create the page header:**
   - Within the <body> section, add a <header> element to represent the header section of your page.
   - Include a heading (e.g., <h1>, <h2>) within the <header> to display the title of your page or a logo.
6. **Add the main content:**
   - After the <header>, add a <main> element to represent the main content section of your page.
   - Add various HTML elements to create the desired content, such as headings, paragraphs, images, lists, etc.
7. **Create a navigation menu (optional):**
   - If your page requires navigation, you can add a <nav> element within the <header>.
   - Use unordered or ordered lists <ul> or <ol> with list items <li> to create the navigation links.
8. **Add additional sections (optional):**
   - Depending on the complexity of your page, you may add more sections like <section>, <article>, <footer>, etc., to structure the content effectively.
9. **Enhance with hyperlinks:**
   - Use the <a> (anchor) element to create hyperlinks that direct users to other web pages or external resources.
   - Set the "href" attribute to the appropriate URL.

10. **Insert multimedia (optional):**
    - To include images, use the <img> element with the "src" attribute pointing to the image file.
    - For multimedia content like videos or audio, you can use the <video> and <audio> elements.
11. **Apply basic styling (optional):**
    - To add styling, create an external CSS file or use the <style> element within the <head> section.
    - Use CSS to modify fonts, colors, spacing, and other visual aspects of your page.
12. **Test the HTML page:**
    - Save your HTML file.
    - Open the file in your web browser to view the page and verify that everything looks as expected.
13. **Debug and revise (if necessary):**
    - Review your HTML code and fix any errors or issues.
    - Test your page on different browsers to ensure compatibility.
14. **Deploy the HTML page (optional):**
    - If you have access to a web server, you can upload the "index.html" file along with any associated CSS or multimedia to make the page accessible online.

Learning Outcome 4: Construct and implement HTML forms

Assessment Criteria:	1. HTML form elements are used. 2. HTML input attributes are used. 3. HTML form validation is used. 4. Webpage is created using form attributes.
Conditions and Resources	1. Applicable tools, utensils, and equipment as prescribed by competency standards. 2. Supply materials 3. Relevant ingredients 4. CBLM related to the learning outcome. 5. Instructions, job sheets, activity sheets, and standard operating procedures 6. Personal protective equipment 7. Module/reference
Contents	1. HTML Form Elements 2. HTML Input Attributes 3. HTML Form Validation 4. Webpages with Form Attributes
Training Methods	1. Discussion 2. Presentation 3. Demonstration 4. Guided Practice 5. Individual Practice 6. Project Work 7. Problem Solving 8. Brainstorming
Learning Materials	1. CBLM 2. Handouts 3. Books, Manuals 4. Module/ Reference 5. Paper 6. Pen
Assessment Methods	1. Written Test 2. Demonstration 3. Oral Questioning

Learning Experience 4: Construct and implement HTML forms

You must perform the learning steps below to achieve the objectives stated in this learning guide. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Steps	Resources specific instructions
1. Students will ask the instructor about work with HTML	1. The instructor will provide the learning materials for construct and implement HTML forms
2. Read the **Information sheet/s**	2. Information Sheet No: 4. Construct and implement HTML forms
3. Complete the **Self-Checks & Answer key sheets**	3. Self-Check No: 4 - Construct and implement HTML forms Answer key No. 4 - Construct and implement HTML forms
4. Read the **Job/ Task sheet** and **Specification Sheet**	4. Job/ task sheet and specification sheet Individual Activity: ▪ Task Sheet No: Develop a responsive website

Information Sheet 4: Construct and implement HTML forms

Learning Objective:

After completion of this information sheet, the learners will be able to explain, define and interpret the following contents:

4.1 HTML Form Elements
4.2 HTML Input Attributes
4.3 HTML Form Validation
4.4 Webpages with Form Attributes

4.1 HTML Form Elements

HTML form elements are essential components used to create interactive web forms on a website. These elements allow users to input data and submit it to the server for processing. Here are some commonly used HTML form elements:

Form: The <form> element in HTML is used to create a container for various form elements. It defines an area on a webpage where users can input data and submit it to the server for processing. Here's an example of how to use the <form> element:

```
<form action="/submit" method="POST">
<!-- Form elements go here -->
</form>
```

In the example above:

- The action attribute specifies the URL or server-side script that will handle the form data when it is submitted. In this case, it is set to "/submit".
- The method attribute determines the HTTP method used to send the form data to the server. Common methods are "GET" and "POST", where "GET" appends the form data to the URL, and "POST" sends it as a separate payload. In this case, it is set to "POST".

Within the <form> element, you can add various form elements like <input>, <textarea>, <select>, and more. These elements allow users to enter and select data.

After adding the necessary form elements, you typically include a submit button to allow users to send.

Figure 6: Form

the form data to the server. Here's an example:

```
<form action="/submit" method="POST">
<!-- Form elements go here -->
<input type="submit" value="Submit">
</form>
```

In this example, an <input> element with type="submit" is added to create a submit button. The value attribute specifies the text displayed on the button. When the user clicks the submit button, the form data will be sent to the URL specified in the action attribute using the HTTP method specified in the method attribute.

It's important to note that the form data is not processed by HTML alone. It requires server-side scripts or programming languages like PHP, Python, or JavaScript to handle and process the submitted data.

Input:

The <input> element in HTML is used to create various types of input fields within a form. It allows users to enter different types of data, such as text, numbers, dates, checkboxes, radio buttons, and more. The specific type of input field is determined by the type attribute of the <input> element. Here are some commonly used types of input fields:

Text input:

```
<input type="text" name="username" placeholder="Enter your username">
```

This creates a single-line text input field where users can enter textual data. The name attribute specifies the name of the input field, which is used to identify the input data on the server. The placeholder attribute provides an optional hint or example value that is displayed inside the input field.

Password Input:

```
<input type="password" name="password" placeholder="Enter your password">
```

This creates a text input field where the entered text is masked (usually with asterisks) to hide the actual characters. It is commonly used for password entry to provide security.

Email input:

```
<input type="email" name="email" placeholder="Enter your email">
```

This creates a text input field specifically designed for entering email addresses. It can validate the input to ensure it is in a valid email format.

Number input:

```
<input type="number" name="age" min="18" max="99">
```

This creates a text input field that only allows numeric input. The min and max attributes can be used to specify the minimum and maximum values allowed.

Checkbox:

```
<input type="checkbox" name="interests" value="sports"> Sports
```

This creates a checkbox that allows users to select multiple options. The name attribute specifies the name of the input field, and the value attribute represents the value associated with the checkbox when it is submitted.

Radio buttons:

```
<input type="radio" name="gender" value="male"> Male
<input type="radio" name="gender" value="female"> Female
```

This creates a group of mutually exclusive radio buttons where users can select only one option. The name attribute should be the same for all radio buttons in a group, while the value attribute represents the value associated with each radio button when it is submitted.

Lebel:

The <**label>** element in HTML is used to associate a text label with a form element. It provides a descriptive label for an input field, making it easier for users to understand what information should be entered in the field. The <**label**> element helps improve accessibility and usability by enhancing the form's overall structure. Here's how to use the <**label**> element:

```
<label for="username">Username:</label>
<input type="text" id="username" name="username">
```

In the example above:

- The <**label**> element is placed before the corresponding form element.
- The **for** attribute of the <**label**> element is set to the id of the associated input field. It establishes a connection between the label and the input field.
- The <**label**> element contains the text that describes the input field. In this case, it says "Username:".

When users click on the label, it focuses the associated input field, improving the user experience. The **for** attribute ensures that the label is associated with the correct input

field. The id attribute of the input field should match the value of the **for** attribute in the <**label**> element.

Alternatively, you can also wrap the input field within the <**label**> element. This eliminates the need for the **for** attribute, as the association is implied:

```
<label>
Username:
<input type="text" name="username">
</label>
```

In this case, the label text "Username:" is placed before the <**input**> element, and they are enclosed within the <**label**> element. The association between the label and the input field is established by their proximity and common parent <**label**> element.

Using <**label**> elements in your forms helps improve accessibility by providing a clear connection between labels and input fields, making it easier for users to navigate and interact with the form.

Text:

If you are referring to a plain text element in HTML, you can use the <p> (paragraph) element or the element to display text on a webpage. Here's how you can use them:

<p**>** element:
The <p> element is used to define a paragraph of text. It creates a block-level element that automatically adds some vertical spacing before and after the text. Here's an example:

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

In this example, the text "This is a paragraph of text." will be displayed as a separate paragraph on the webpage.

 element:

The element is an inline element that is used to group and style inline elements or small sections of text. It does not add any line breaks or additional spacing. Here's an example:

```
<p>This is a <span>small section</span> of text.</p>
```

In this example, the text "This is a" and "of text." will be displayed as normal text, while the text "small section" will be wrapped in a <**span**> element. The <**span**> element can be styled using CSS to apply specific formatting or styles.

These are just two examples of displaying text in HTML. Depending on the specific use case and desired styling, there are other HTML elements and techniques available to display and structure text on a webpage.

Select:

The <select> element in HTML is used to create a dropdown list or a list box, allowing users to select one or multiple options from a predefined set of choices. Here's how to use the <select> element:

```
<select name="country">
  <option value="usa">United States</option>
  <option value="canada">Canada</option>
  <option value="uk">United Kingdom</option>
</select>
```

In the example above:

- The <select> element is used to create the dropdown list.
- The name attribute specifies the name of the input field, which is used to identify the selected option on the server.
- Inside the <select> element, multiple <option> elements are added. Each <option> represents an available choice in the dropdown list.
- The value attribute of each <option> specifies the value associated with the selected option when the form is submitted. It is typically used on the server-side to process the selected option.
- The text content within the <option> elements is the visible label or name of each option that appears in the dropdown list.

Users can select one option from the list by clicking on it. The selected option will be displayed in the dropdown list, and its associated value will be sent to the server when the form is submitted.

To allow users to select multiple options, you can add the multiple attribute to the <select> element:

```
<select name="colors" multiple>
<option value="red">Red</option>
<option value="green">Green</option>
<option value="blue">Blue</option>
</select>
```

In this case, users can select multiple options by holding down the Ctrl (Windows) or Command (Mac) key while clicking on the desired options.

The <select> element provides a way to present a list of choices to users in a compact and organized manner, making it easier for them to select the appropriate option(s) from the available set.

Option:

The <**option**> element in HTML is used to define an option within a <**select**> element. It represents an individual item or choice that users can select from a dropdown list or list box. Here's how to use the <**option**> element:

```
<select name="country">
<option value="usa">United States</option>
<option value="canada">Canada</option>
<option value="uk">United Kingdom</option>
</select>
```

In the example above:

- The <**select**> element creates the dropdown list.
- Each <**option**> element represents an individual choice within the dropdown list.
- The value attribute of each <**option**> specifies the value associated with the selected option. This value is sent to the server when the form is submitted or can be accessed via JavaScript.
- The text content within the <option> tags is the visible label or name of each option that appears in the dropdown list.

Users can select one option from the list by clicking on it. The selected option will be displayed in the dropdown list, and its associated value will be sent to the server when the form is submitted.

Additionally, you can set a default selected option by adding the selected attribute to the desired <**option**> element:

```
<select name="country">
<option value="usa" selected>United States</option>
<option value="canada">Canada</option>
<option value="uk">United Kingdom</option>
</select>
```

In this case, "**United States**" will be pre-selected when the page loads.

The <**option**> element allows you to define multiple options within a <**select**> element, providing users with a list of choices to select from.

Textarea

The <textarea> element in HTML is used to create a multiline text input field where users can enter and edit larger blocks of text. It allows for the entry of multiple lines of text and is often used when longer, more detailed input is required. Here's how to use the <textarea> element:

```
<textarea name="message" rows="4" cols="40">Enter your message here</textarea>
```

In the example above:

- The <textarea> element creates the text input field.
- The name attribute specifies the name of the input field, which is used to identify the input data on the server.
- The rows attribute determines the number of visible lines in the textarea. In this case, it is set to 4, but you can adjust it according to your needs.
- The cols attribute determines the number of visible characters in each line of the textarea. In this case, it is set to 40, but you can modify it as per your requirements.
- The text content between the opening and closing <textarea> tags represents the initial value or placeholder text displayed in the textarea. Users can modify this text when they interact with the input field.

When users enter or edit text within the <textarea>, they can see the text wrapping to fit within the specified number of rows and columns. The entered text can be accessed on the server-side using the specified name attribute.

You can style the <textarea> element using CSS to adjust its appearance, such as its width, height, font, and more. Additionally, you can utilize JavaScript to manipulate the textarea's behavior or handle events related to its input.

The <textarea> element is useful when you need to collect longer and more descriptive text input from users, such as comments, messages, or any other form of multiline textual data.

Radio

The <input> element with type="radio" in HTML is used to create a radio button input field. Radio buttons are used when users need to select a single option from a predefined set of choices. Here's how to use the <input> element with type="radio":

```
<input type="radio" name="gender" value="male"> Male
<input type="radio" name="gender" value="female"> Female
```

In the example above:

- Two radio buttons are created, each represented by an <input> element with type="radio".
- The name attribute is used to group the radio buttons together. In this case, both radio buttons have the same name attribute, which is "gender". This ensures that only one option can be selected within the group.
- The value attribute specifies the value associated with each radio button. When the form is submitted, the selected value of the chosen radio button will be sent to the server.
- The label text "Male" and "Female" is placed after each radio button. The label text provides a description or name for each option and improves the accessibility and usability of the radio buttons.

Users can select only one option at a time within a group of radio buttons. Selecting a new option automatically deselects the previously selected option within the same group. The selected value of the chosen radio button can be accessed on the server-side or via JavaScript for further processing.

It's important to note that each radio button within a group should have a unique value attribute, as it is used to differentiate between the selected options.

Radio buttons are commonly used in forms when users need to make a single choice from a set of mutually exclusive options, such as selecting a gender, preference, or category.

Checkbox

The <input> element with type="checkbox" in HTML is used to create a checkbox input field. Checkboxes allow users to select one or more options from a predefined set of choices. Here's how to use the <input> element with type="checkbox":

```
<input type="checkbox" name="interests" value="sports"> Sports
<input type="checkbox" name="interests" value="music"> Music
```

In the example above:

- Two checkboxes are created, each represented by an <input> element with type="checkbox".
- The name attribute is used to group the checkboxes together. In this case, both checkboxes have the same name attribute, which is "interests". This allows users to select multiple options within the same group.
- The value attribute specifies the value associated with each checkbox. When the form is submitted, the selected values of the chosen checkboxes will be sent to the server.
- The label text "Sports" and "Music" is placed after each checkbox. The label text provides a description or name for each option and improves the accessibility and usability of the checkboxes.

Users can select one or more checkboxes at a time within a group. Each selected checkbox represents a chosen option, and the associated value will be sent to the server when the form is submitted.

To pre-select a checkbox, you can add the checked attribute to the desired <input> element:

```
<input type="checkbox" name="interests" value="sports" checked> Sports
<input type="checkbox" name="interests" value="music"> Music
```

In this case, the "Sports" checkbox will be pre-selected when the page loads.

Checkboxes are commonly used in forms when users need to select multiple options or toggle certain settings on or off.
It's important to note that each checkbox within a group should have a unique value attribute, as it is used to differentiate between the selected options.

Uploader

To create an uploader in HTML, you can use the <input> element with type="file". This allows users to select and upload files from their local device to a server. Here's how to create a basic file uploader:

```
<form action="upload.php" method="post" enctype="multipart/form-data">
    <input type="file" name="fileUpload">
    <input type="submit" value="Upload">
</form>
```

In the example above:

- The <form> element is used to wrap the file uploader.
- The action attribute specifies the URL or server-side script that will handle the file upload. In this case, it is set to "upload.php". You need to replace it with the appropriate server-side script or endpoint.
- The method attribute is set to "post" to send the file to the server as part of an HTTP POST request.
- The enctype attribute is set to "multipart/form-data" to indicate that the form data will include file content.
- The <input> element with type="file" creates the file input field. Users can click on it to open a file selection dialog and choose a file from their device.
- The name attribute of the file input field (in this case, "fileUpload") is used to identify the uploaded file on the server.
- The <input> element with type="submit" creates a submit button for the form. Users can click on it to initiate the file upload process.

When the form is submitted, the selected file(s) will be sent to the server-side script specified in the action attribute. You need to handle the file upload on the server using server-side programming languages like PHP, Python, or Node.js.

Note that the actual file handling and processing will need to be implemented on the server-side using appropriate server-side technologies and security measures. The example above provides the basic structure for creating a file uploader in HTML, but it requires server-side logic to handle and process the uploaded files.

Button

The <button> element in HTML is used to create a clickable button on a webpage. Buttons are interactive elements that users can click to perform an action or trigger a function. Here's how to use the <button> element:

```
<button type="button">Click me</button>
```

In the example above:

- The <button> element creates a button.
- The type attribute is set to "button" to indicate that the button does not have any specific behavior associated with it by default. This type is commonly used for buttons that trigger JavaScript functions.
- The text "Click me" between the opening and closing <button> tags represent the label or text displayed on the button.

You can also use the <button> element within a form to create a submit button:

```
<button type="submit">Submit</button>
```

In this case, the **type** attribute is set to "submit" to indicate that clicking the button will submit the form.

You can further customize the appearance and behavior of buttons using CSS and JavaScript. CSS can be used to style the button, such as setting its background color, font, padding, and more. JavaScript can be used to add interactivity to the button, such as handling click events and performing specific actions when the button is clicked.

Buttons are versatile elements used in various contexts, such as form submission, navigation, triggering functions, and more.

Legend

The <legend> element in HTML is used to provide a caption or a title for a group of form elements within a <fieldset> element. It helps in organizing and structuring form controls by visually grouping them together and providing a descriptive label. Here's how to use the <legend> element:

```
<fieldset>
<legend>User Information</legend>
<label for="name">Name:</label>
<input type="text" id="name" name="name">

<label for="email">Email:</label>
```

```
<input type="email" id="email" name="email">
</fieldset>
```

In the example above:

- The <fieldset> element creates a grouping container for form elements.
- The <legend> element is placed inside the <fieldset> element and provides a caption or title for the group of form elements.
- The text "User Information" between the opening and closing <legend> tags represents the caption for the form elements within the fieldset.
- Inside the fieldset, we have two form elements - <label> and <input> elements for collecting the user's name and email.
- The <label> elements provide textual descriptions for the corresponding form controls using the for attribute and the id attribute. This helps in associating the label with its respective input element.

Using the <legend> element with <fieldset> provides a visual and semantic grouping of form elements, making it easier for users to understand the context and purpose of the input fields. It also aids in accessibility by providing screen readers and other assistive technologies with additional information about the grouped form controls.

You can style the <fieldset> and <legend> elements using CSS to modify their appearance, such as changing the border, background, font, and alignment.

The <legend> element is particularly useful when you have a set of related form controls that need to be presented together with a clear and descriptive heading or title.

4.2 HTML Input Attributes

HTML input elements have various attributes that can be used to modify their behavior, appearance, and functionality. Here are some commonly used attributes for HTML input elements:

- **type:** Specifies the type of input field, such as text, password, checkbox, radio, etc.
- **name:** Provides a name for the input field, which is used to identify the input data on the server-side when the form is submitted.
- **value:** Sets the initial value of the input field.
- **placeholder:** Displays a short hint or example text within the input field, providing a suggestion to the user about the expected input.
- **required:** Specifies that the input field must be filled out before the form can be submitted.
- **disabled:** Disables the input field, preventing the user from interacting with or modifying its value.
- **readonly:** Makes the input field read-only, allowing the user to view the value but not modify it.
- **maxlength:** Sets the maximum number of characters allowed in the input field.

- **min and max:** Defines the minimum and maximum values or range allowed for number or date inputs.
- **pattern:** Specifies a regular expression pattern that the input value must match to be considered valid.
- **autocomplete:** Enables or disables the browser's autocomplete feature for the input field.
- **autofocus:** Specifies that the input field should automatically receive focus when the page loads.
- **multiple:** Indicates that multiple values can be selected in a file input or select input.
- **checked**: Pre-selects a checkbox or radio button when the page loads.
- **step**: Specifies the increment or decrement value for number inputs.

These are just a few examples of the attributes that can be used with HTML input elements. The specific attributes available depend on the type of input element being used. By utilizing these attributes, you can customize the behavior and appearance of your input fields to suit your needs.

Type: The **type** attribute is used to specify the type of input that is expected in an HTML input element. It determines the behavior and appearance of the input field. Here are some commonly used values for the **type** attribute:

- **text**: Creates a text input field where users can enter single-line text.
- **password**: Creates a password input field where entered text is masked (e.g., shown as asterisks) for security purposes.
- **email**: Creates an email input field that validates the entered value as an email address.
- **number**: Creates a numeric input field where users can enter numbers.
- **checkbox**: Creates a checkbox input field where users can select one or more options from a set of choices.
- **radio**: Creates a radio button input field where users can select a single option from a set of choices.
- **file**: Creates a file upload input field that allows users to select and upload files from their device.
- **date**: Creates a date input field for selecting a specific date.
- **time**: Creates a time input field for selecting a specific time.
- **color**: Creates a color picker input field for selecting a specific color.

These are just a few examples of the type attribute values available for HTML input elements. Each type value has its own specific behavior and may have additional attributes and validation rules associated with it.

It's important to choose the appropriate type value based on the desired input and to consider user experience and validation requirements. The type attribute helps to ensure that the input field accepts the correct type of data and provides appropriate user interface features to enhance usability.

Name: The **name** attribute is a fundamental attribute used in HTML to give a name or identifier to various elements, including form inputs. It provides a way to reference and identify elements in client-side scripting and when submitting form data to the server. Here are a few key points about the **name** attribute:

- Form Inputs: The name attribute is commonly used with form input elements like <**input**>, <**select**>, and <**textarea**>. It associates the user's input with a specific name that can be used to retrieve the input value on the server-side or manipulate it with JavaScript.

```
<input type="text" name="username">
```

- Unique Identifier: Each **name** attribute within a form should have a unique value to differentiate one input from another. It allows the server-side script or client-side code to identify and handle individual form fields accurately.

```
<input type="radio" name="gender" value="male"> Male
<input type="radio" name="gender" value="female"> Female
```

- Form Data Submission: When a form is submitted, the values of input fields with a **name** attribute are included in the request payload sent to the server. The name attribute acts as a key, and the user's input serves as the corresponding value.

- Accessibility and Form Labels: The **name** attribute also plays a role in accessibility. It should be used in conjunction with appropriate form labels (using the <label> element) to provide a clear and descriptive association between the input field and its purpose.

```
<label for="username">Username:</label>
<input type="text" id="username" name="username">
```

By utilizing the name attribute, you can effectively organize and process form data, distinguish between different form inputs, and improve accessibility by associating labels with input fields.

Value

The **value** attribute is used in HTML to set the initial or default value of an input element. It specifies the value that will be displayed or submitted when the form is rendered or submitted. Here are a few key points about the **value** attribute:

- Input Elements: The **value** attribute is commonly used with input elements like <**input**> and <**button**>. For input fields, it sets the initial value that will be displayed when the page is loaded.

```
<input type="text" value="John Doe">
```

- Default Selection: For radio buttons and checkboxes, the **value** attribute determines the value that will be submitted if the input is selected by default.

```
<input type="radio" name="gender" value="male" checked> Male
<input type="radio" name="gender" value="female"> Female
```

- User Input: The **value** attribute can also be used to retrieve or set the current **value** of an input field through JavaScript. You can access or modify the value using the DOM API or JavaScript frameworks.

```
<input type="text" id="myInput" value="Initial Value">

<script>
 var inputValue = document.getElementById("myInput").value;
 console.log(inputValue);  // Output: "Initial Value"

 document.getElementById("myInput").value = "New Value";
</script>
```

- Form Submission: When a form is submitted, the **value** attribute determines the value that will be sent to the server as part of the form data. It represents the user's input or the default value of the input field.

It's important to note that changing the value attribute dynamically using JavaScript does not affect the user's input or the actual value submitted unless the user interacts with the input field.

By utilizing the **value** attribute, you can set the initial value of an input field, define default selections, and access or modify the value programmatically using JavaScript.

Placeholder

The placeholder attribute is used in HTML to provide a short hint or example text within an input element. It is typically used to give users a suggestion or an indication of the type of input expected in the field. The placeholder text is displayed in a lighter font or style and is automatically cleared when the user starts typing in the input field. Here are a few key points about the placeholder attribute:

- Input Elements: The placeholder attribute is commonly used with input elements like <input> and <textarea>. It is particularly useful for text input fields where the purpose or format of the input may not be immediately apparent to the user.

```
<input type="text" placeholder="Enter your name">
```

- Example Usage: The placeholder text can be used to provide an example or a descriptive hint to guide users on how to fill out the input field.

```
<input type="email" placeholder="example@example.com">
```

- Styling and Accessibility: The appearance of the placeholder text is determined by the browser and can be customized using CSS. However, it's important to note that the placeholder text is not a replacement for a proper form label. To ensure accessibility, always use a <label> element associated with the input field.

```
<label for="email">Email:</label>
<input type="email" id="email" placeholder="example@example.com">
```

- JavaScript Interaction: The placeholder text is not submitted with the form data. If you need to access or validate the user's input, use the value attribute instead.

```
<input type="text" id="myInput" placeholder="Enter your name">

<script>
  var inputValue = document.getElementById("myInput").value;
  console.log(inputValue);  // Output: user's input value, not the placeholder text
</script>
```

The placeholder attribute provides a convenient way to offer users guidance or examples for filling out input fields. It improves the usability and clarity of your forms, helping users understand the expected input.

Self-Check Sheet 4: Construct and implement HTML forms

1 What is the purpose of the <form> element?
2 How do you create a text input field in HTML?
3 What does the <label> element do?
4 How do you create a dropdown list in HTML?
5 What does the required attribute do?
6 How can you disable an input field?
7 How can you set a default value for an input field?
8 What is the purpose of the <textarea> element?
9 How do you create radio buttons in HTML?
10 How do you create checkboxes in HTML?
11 How can you upload files using HTML?
12 What is the purpose of the <button> element?
13 How can you specify a range of allowed values for a number input field?
14 How do you create a submit button in HTML?
15 What is the purpose of the <legend> element?
16 How can you set a default selected option in a dropdown list?
17 How can you limit the number of characters entered in a text input field?
18 What is the purpose of the pattern attribute?
19 How can you create a time input field in HTML?
20 How can you disable the autocomplete feature for an input field?

Answer Key 4: Construct and implement HTML forms

1 The <form> element is used to create an HTML form that collects user input.

2 You can create a text input field using the <input> element with the type attribute set to "text".

3 The <label> element provides a label or description for an associated form control.

4 You can create a dropdown list using the <select> element, along with nested <option> elements.

5 The required attribute specifies that an input field must be filled out before the form can be submitted.

6 You can disable an input field by adding the disabled attribute to the element.

7 You can set a default value for an input field using the value attribute.

8 The <textarea> element is used to create a multi-line text input field.

9 You can create radio buttons using the <input> element with the type attribute set to "radio".

10 You can create checkboxes using the <input> element with the type attribute set to "checkbox".

11 You can create a file upload field using the <input> element with the type attribute set to "file".

12 The <button> element is used to create a clickable button on a webpage.

13 You can use the min and max attributes to define the minimum and maximum values for a number input field.

14 You can create a submit button using the <input> element with the type attribute set to "submit" or by using a <button> element with the type attribute set to "submit".

15 The <legend> element provides a caption or title for a group of form elements within a <fieldset> element.

16 You can use the selected attribute on the <option> element to specify the default selected option.

17 You can use the maxlength attribute to set the maximum number of characters allowed in a text input field.

18 The pattern attribute is used to specify a regular expression pattern that the input value must match to be considered valid.

19 You can create a time input field using the <input> element with the type attribute set to "time".

20 You can disable autocomplete by adding the autocomplete attribute to the <input> element and setting it to "off".

Job Sheet 4.1 Develop a Responsive Website

Objectives:

- Create a responsive website that adapts to different screen sizes and devices.
- Ensure optimal user experience by designing and implementing a fluid layout that adjusts based on screen dimensions.
- Apply responsive design techniques to handle varying viewports, such as mobile devices, tablets, and desktop screens.
- Optimize website performance by using responsive images and media queries to load appropriate assets based on device capabilities.
- Test and validate the website's responsiveness across multiple devices and web browsers.

Required Equipment:

- Computer with a text editor or integrated development environment (IDE) for web development.
- Web browser for testing and previewing the website.
- Device emulator or responsive design testing tools for cross-device testing.

Procedure:

1. Gather the project requirements and identify the target audience and devices the website needs to support.
2. Plan the website's layout and structure, considering the content hierarchy and the optimal user experience on different screen sizes.
3. Create a wireframe or mockup of the website's design, focusing on its responsive behavior and the visual appearance at various breakpoints.
4. Set up the project directory and create the necessary HTML, CSS, and JavaScript files.
5. Start with the mobile-first approach by designing and coding the website's layout for small screens first.
6. Use CSS media queries to define breakpoints at which the layout and design should adapt to different screen sizes.
7. Apply responsive design techniques, such as flexible grids, fluid images, and relative units (e.g., percentages, em, rem), to create a fluid layout that adjusts proportionally across devices.
8. Utilize CSS frameworks like Bootstrap or Foundation to expedite the responsive design implementation, if applicable.
9. Optimize images and media by using appropriate formats, compression techniques, and responsive image tags (e.g., <picture>, srcset) to serve different image sizes based on device capabilities.
10. Test the website's responsiveness by previewing it in various web browsers and using device emulators or responsive design testing tools.
11. Make necessary adjustments to the design, layout, and media queries to ensure a smooth and consistent user experience across different devices.

12 Validate the website's accessibility and usability, ensuring that all interactive elements are easily accessible and usable on various devices.
13 Optimize the website's performance by minifying CSS and JavaScript files, compressing images, and implementing caching techniques.
14 Deploy the responsive website to a web server or hosting platform for public access.
15 Continuously monitor and test the website's responsiveness on new devices and web browsers, making updates or refinements as necessary.
16 Document the responsive design implementation, including a summary of the responsive breakpoints used, media query logic, and any specific design considerations.
17 Provide documentation on how to maintain and update the responsive website, including guidelines for adding or modifying content while maintaining responsiveness.

Remember to prioritize user experience and accessibility while designing and developing a responsive website. Regularly test and validate the responsiveness to ensure a seamless experience across different devices and screen sizes.

Note: This job sheet assumes a basic understanding of HTML, CSS, and web development concepts related to responsive design.

Review of Competency

Below is your assessment rating for module **Work with HTML**

Assessment of Performance Criteria	Yes	No
The structure of HTML (Hypertext Mark-up Language) is interpreted.		
DHTML tags are introduced.		
Entities & attributes of HTML are interpreted.		
Typography is interpreted.		
Guidelines for web typography are applied.		
Guidelines for print typography are applied.		
Most common HTML tags are used.		
Most common entities & attributes are used.		
HTML multicolumn layout is implemented.		
HTML Graphics are used.		
HTML Media is used.		
HTML page is saved.		
HTML form elements are used.		
HTML input attributes are used.		
HTML form validation is used.		
A webpage is created using form attributes.		

I now feel ready to undertake my formal competency assessment.

Signed:

Date:

Reference

https://www.google.com/search?q=Data+Layer&tbm=isch&ved=2ahUKEwj0o9DW1tGBAxUUjmMGHaBZBEIQ2-cCegQIABAA&oq=Data+Layer&gs_lcp=CgNpbWcQAzIFCAAQgAQyBQgAEIAEMgUIABCABDIFCAAQgAQyBQgAEIAEMgUIABCABDIFCAAQgAQyBQgAEIAEMgUIABCABDIFCAAQgARQ1QZY7xNg0xZoAHAAeACAAdcBiAHXCJIBBTAuNC4ymAEAoAEBqgELZ3dzLXdpei1pbWfAAQE&sclient=img&ei=KLwXZfTvKpScjuMPoLORkAQ&bih=629&biw=1366&hl=en#imgrc=G30Pvdc79Lk2QM

https://www.google.com/search?q=XAMPP%2C+WAMP%2C+MAMP%2C+and+LAMP&sca_esv=569660528&hl=en&tbm=isch&source=hp&biw=1366&bih=629&ei=I7wXZc-IFfbi2roPyK2AwAI&iflsig=AO6bgOgAAAAAZRfKMy7gVqgROlsU4in5HyWY6hDrV-nv&ved=0ahUKEwiPponU1tGBAxV2sVYBHcgWACgQ4dUDCAc&uact=5&oq=XAMPP%2C+WAMP%2C+MAMP%2C+and+LAMP&gs_lp=EgNpbWciG1hBTVBQLCBXQU1QLCBNQU1QLCBhbmQgTEFNUEiBElCcBVjpD3ABeACQAQCYAa4BoAHMAqoBAzAuMrgBA8gBAPgBAvgBAYoCC2d3cy13aXotaW1nqAIA&sclient=img#imgrc=NhjcFF8drA0YHM

https://www.google.com/imghp?hl=en&tab=ri&ogbl

https://www.google.com/search?q=Step+by+steoExport+the+Object+Adobe+Photoshop+-+wikihow&tbm=isch&ved=2ahUKEwiN98mI39GBAxWgz6ACHW6JCdEQ2-cCegQIABAA&oq=Step+by+steoExport+the+Object+Adobe+Photoshop+-+wikihow&gs_lcp=CgNpbWcQA1DaF1jdPGCFQ2gAcAB4AIABqgKIAYQNkgEDMi03mAEAoAEBqgELZ3dzLXdpei1pbWfAAQE&sclient=img&ei=9MQXZY2tOaCfg8UP7pKmiA0&bih=629&biw=1366&hl=en

Development of CBLM:

The Competency Based Learning Material (CBLM) of '**Work with HTML' (**Occupation: Web Design, Level-3**)** for National Skills Certificate is developed by NSDA with the assistance of SIMEC System, ECF consultancy & SIMEC Institute JV (Joint Venture Firm) in the month of June 2023 under the contract number of package SD-9A dated 07th May 2023.

Sl No.	Name & Address	Designation	Contact number
1	Khondoker Ali Asgor Pavel	Writer	01711 873 008
2	Fatema Tuj Johura Arzu	Editor	01912 464 747
3	Md. Amir Hossain	Co-Ordinator	01631 670 445
4	Md. Saif Uddin	Reviewer	01723 004 419