1. HTML and CSS

1.1 HTML Basic

1.1.1 Introduction

HTML stands for HyperText Markup Language, which is the most widely used language on the web to develop web pages. A markup language is a set of markups tags. HTML documents are described by HTML tags. Each HTML tag describes different document content.

In short

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content.
- HTML elements label pieces of content such as "this is a heading", "this is paragraph",..

Example:

```
<!DOCTYPE html>
<html>
	<head>
		<title>Page Title</title>
	</head>
	<body>
		<h1>My First Heading</h1>
		<p>My first paragraph.</p>
	</body>
</html>
```

Example Explained

- The <!DOCTYPE html> declaration defines that this document is an HTML5 document
- The <html> element is the root element of an HTML page
- The <head> element contains meta information about the HTML page
- The <title> element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
- The <body> element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- The <h1> element defines a large heading
- The element defines a paragraph

Concept of tag

HTML tags are the building blocks of an HTML document, used to define and structure content on a web page. Tags are keywords (tag names) surrounded by angle brackets (< >) and typically come in pairs: an opening tag and a closing tag. The content is placed between these tags, and the closing tag has a forward slash (/) before the tag name.

Basic Structure of HTML Tags:

- Opening tag: <tagname>
- Closing tag: </tagname>
- Content: The actual content that the tag wraps.

Syntax:

<tagname> contents </tagname>

Example :

 This is paragraph .

The first tag in a pair is the start tag, the second tag is the end tag. The end tag is written like the start tag but with a slash before the tag name.

HTML document structure

An HTML document structure defines how content is organized on a webpage. It consists of specific tags that form the skeleton of the page. The basic structure includes several key elements:

1. <!DOCTYPE html>

This section declares the document type and version of HTML (HTML5 in this case).It is called as the document type declaration (DTD). It ensures that the browser interprets the page correctly.

Technically <!DOCTYPE > is not a tag/element, it just an instruction to the browser about the document type. It is a null element which does not contain the closing tag, and must not include any content within it.

2. <html>

It is a root element that contains all the content of the webpage.

3. <head>

This tag contains metadata (data about the document) such as the title, character encoding, linked stylesheets, and scripts.

Key Elements:

<meta>: Defines metadata like character encoding (<meta charset="UTF-8">) and viewport settings for responsive design.

<title>: Specifies the title of the document, which appears in the browser tab.

<link>: Links external resources such as CSS stylesheets.

<script>: Links or includes JavaScript files.

Meta Tags

HTML <meta> tag is used to represent the metadata about the HTML document. It specifies page description, keywords, copyright, language, author of the documents, etc.

The metadata does not display on the webpage, but it is used by search engines, browsers and other web services which scan the site or webpage to know about the webpage.

The <meta> tag is placed within the <head> tag, and it can be used more than one times in a document.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="keywords" content="HTML, CSS, JavaScript, Tutorials">
    <meta name="description" content="Free Online tutorials">
    <meta name="author" content="thisauthor">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
      <h2>Example of Meta tag</h2>
      <p>This example shows the use of meta tag within an HTML document</p>
      </body>
</html>
```

4. <body>

This tag contains the visible content of the webpage, including text, images, links, and other media.

Key Elements:

<header>: Contains introductory content or navigational links (like a website header).

<nav>: Defines navigation links for the website.

<main>: Represents the main content area of the document.

<section>: Groups content into logical sections (often used for different topics or parts of the page).

<article>: Represents independent, self-contained content.

<footer>: Contains footer information, such as copyright details or contact info.

HTML Headings

HTML headings are titles or subtitles that we want to display on a webpage.
HTML headings are defined with the <h1> to <h6> tags.
<h1> defines the most important heading.
<h6> defines the least important heading.

Syntax:

```
<heading type> Heading title </heading type>
```

Example:

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

Search engines use the headings to index the structure and content of your web pages.
Users often finds a page by its headings. It is important to use headings to show the document structure.
<h1> headings should be used for main headings, followed by <h2> headings, then the less important <h3>, and so on.

HTML Paragraph

The HTML element defines a paragraph. These have both opening and closing tags. So anything mentioned within and is treated as a paragraph
A paragraph always starts on a new line.

Syntax:

```
<paragraph tag> Paragraph content here </paragraph tag>
```

Example:

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

HTML Comment

The comment tag (<!-- Comment-->) is used to insert comments in the HTML code. Contents inside the comment tag are not displayed in the browser. It is a good practice to use comments in our code for better understanding of codes.

Syntax:

```
<!-- Comments here -->
```

Notice that there is an exclamation point (!) in the start tag, but not in the end tag

Example:

```
<!-- This is a comment -->
<p>This is a paragraph.</p>
<!-- Remember to add more information here -->
```

Phrase Elements

Phrase elements are designed for special purpose. They add structural information to text fragments. The usual meanings of phrase elements are following −

<abbr>	Indicates an abbreviated form like pvt. inc. etc.
<acronym>	Indicates an acronym (e.g., WAC, radar, etc.).
	Indicates emphasis.
	Indicates stronger emphasis.
<cite>	Contains a citation or a reference to other sources.
<dfn>	Indicates that this is the defining instance of the enclosed term.
<code>	Designates a fragment of computer code.
<samp>	Designates sample output from programs, scripts, etc.
<kbd>	Indicates text to be entered by the user.
<var>	Indicates an instance of a variable or program argument.

HTML elements and attributes

HTML Elements

An HTML element usually consists of a start tag and end tag, with the content inserted in between, for example <tagname>Content goes here...</tagname>. The HTML element is everything from the start tag to the end tag. for example: This is paragraph tag .

HTML elements can be nested (elements can contain elements). All HTML documents consist of nested HTML elements.

HTML elements with no content are called empty elements.
 is an empty element without a closing tag (the
 tag defines a line break). Empty elements can be "closed' in the opening tag like this
.

HTML5 does not require empty elements to be closed. But if we want stricter validation, or if we need to make our document readable by XAL parsers, we must close all HTML elements properly.

HTML tags are not case sensitive: <P> means the same as The HTML5 standard does not require lowercase tags, but W3C recommends lowercase in HTML, and demands lowercase for stricter document types like XHTML.

Syntax:

```
<tagname>Content goes here...</tagname>
```

Examples of some HTML elements:

```
<h1>My First Heading</h1>
<p>My first paragraph.</p>
```

HTML Attributes

- HTML attributes are special words which provide additional information about the elements or attributes are the modifier of the HTML element.
- Each element or tag can have attributes, which defines the behaviour of that element.
- Attributes should always be applied with start tag.
- The Attribute should always be applied with its name and value pair.
- The Attributes name and values are case sensitive, and it is recommended by W3C that it should be written in Lowercase only.
- We can add multiple attributes in one HTML element, but need to give space between two attributes.

Syntax:

```
<element attribute_name="value"> Content Here </element>
```

Example 1 :

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
   <h1> This is Style attribute</h1>
   <p style="height: 50px; color: blue">It will add style property in element</p>
   <p style="color: red">It will change the color of content</p>
</body>
</html>
```

Example 2:

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

Global Attributes

HTML global attributes are attributes that can be applied to any HTML element, regardless of its specific tag type. These attributes provide additional functionality or behavior to HTML elements, and they help with styling, scripting, and accessibility.

Here are some key examples of global attributes in HTML:

1. **id**: Specifies a unique identifier for the element. It's used to target the element in JavaScript or CSS.
   Example: Content

2. **class**: Specifies one or more class names for the element, which can be targeted by CSS or JavaScript.
   Example: Text here

3. **style**: Specifies inline CSS styles for the element.
   Example: This is red

4. **title**: Provides additional information about the element, usually displayed as a tooltip when the mouse hovers over it.
   Example: <button title="Click me to submit">Submit</button>

5. **lang**: Specifies the language of the element's content.
   Example: Hello, world!

6. **dir**: Specifies the direction of text in the element (e.g., left-to-right or right-to-left).
   Example: This is right-to-left text

7. **tabindex**: Specifies the order in which the element should receive focus when the user presses the "Tab" key.
   Example: <input tabindex="1">

8. **data- attributes***: Custom attributes that allow you to store extra data on an element, which can be accessed with JavaScript.
   Example: User Info

9. **accesskey**: Specifies a shortcut key to activate or focus on an element.
   Example: <button accesskey="s">Save</button>

10. **hidden**: Indicates that the element is not relevant or should not be displayed (used to hide elements).
    Example: <div hidden>This is hidden

These attributes are considered global because they can be applied to any HTML element, not limited to specific ones like , , or <a>.

Formatting Tags

Basic Text Formatting Tags

Text formatting in HTML refers to the way text is displayed on a web page. It is the process of applying various styles, colors, fonts, sizes, and other visual enhancements to text content within an HTML document. HTML offers a range of tags that can be used to format text, including:

<b> - Bold text
 - Important text
<i> - Italic text
 - Emphasized text
<mark> - Marked text
<small> - Smaller text
<del> - Deleted text
<ins> - Inserted text
<sub> - Subscript text
<sup> - Superscript text

Block Level Formatting Tags

In HTML, **block-level elements** are those that take up the full width available to them and start on a new line. They are typically used to structure the layout of a webpage, creating sections and organizing content. When used in a block-level tag, the content will generally stack vertically, one element below the other.

- **Structure and layout**: Block-level elements help structure the page by organizing content into sections, paragraphs, lists, etc.
- **Start on a new line**: They typically take up the full width available and start on a new line.
- **Contain other elements**: Many block-level elements can contain other block-level or inline elements, helping to build complex layouts.
- These tags allow you to create a logical and organized structure for your web pages, ensuring proper presentation and accessibility.

Here are some commonly used block-level formatting tags in HTML:

1. **`<div>` (Division) :** The `<div>` tag is a generic block-level container used to group elements together for styling or scripting purposes. It has no inherent style or meaning but is useful for creating layout structures.
   Example:
   ```
   <div>
     <p>This is a paragraph inside a div.</p>
   </div>
   ```

2. **`<p>` (Paragraph) :** The `<p>` tag is used to define a paragraph. It automatically adds space before and after the text, creating a visual separation between blocks of text.
   Example:
   ```
   <p>This is a paragraph.</p>
   ```

3. **`<hr>` (Horizontal Rule) :** The `<hr>` tag creates a thematic break or horizontal line in the page. It is used to visually separate sections of content.
   Example:
   ```
   <hr>
   ```

4. **`<h1>`, `<h2>`, ..., `<h6>` (Headings) :** These tags are used to define headings, with `<h1>` being the most important (largest) and `<h6>` being the least important (smallest). They help structure the content by defining sections and subsections.
   Example:
   ```
   <h1>Main Heading</h1>
   <h2>Subheading</h2>
   <h3>Sub-subheading</h3>
   ```

5. **`<blockquote>` (Block of Quoted Text) :** The `<blockquote>` tag is used to display block of text that is quoted from another source. It is usually indented to indicate that it's a quotation.
   Example:
   ```
   <blockquote>
       This is a quoted block of text.
   </blockquote>
   ```

6. **`<ul>` (Unordered List) :** The `<ul>` tag is used to define an unordered (bulleted) list, containing list items (`<li>`). The list items are displayed with bullet points by default.
   Example:
   ```
   <ul>
     <li>Item 1</li>
     <li>Item 2</li>
   </ul>
   ```

7. **`<ol>` (Ordered List) :** The `<ol>` tag defines an ordered (numbered) list, also containing list items (`<li>`). Items in an ordered list are numbered automatically.
   Example:
   ```
   <ol>
       <li>First item</li>
       <li>Second item</li>
   </ol>
   ```

8. **`<li>` (List Item) :** The `<li>` tag is used to define an individual item within a list, whether it's ordered or unordered.
   Example:
   ```
   <ul>
     <li>First item</li>
     <li>Second item</li>
   </ul>
   ```

9. **`<form>` (Form) :** The `<form>` tag is used to collect user input through various controls like text fields, buttons, checkboxes, and more. It is a block-level element that defines the area where users can submit data.
   Example:
   ```
   <form>
       <label for="name">Name:</label>
       <input type="text" id="name" name="name">
       <button type="submit">Submit</button>
   </form>
   ```

10. **`<header>` (Header) :** The `<header>` tag represents the introductory content of a section or page. It typically contains headings, navigation, and other introductory information.

Example:

```
<header>
  <h1>Welcome to My Website</h1>
  <nav>Navigation menu here</nav>
</header>
```

11. **`<footer>` (Footer) :** The `<footer>` tag defines the footer section of a document or section. It often contains information such as copyright statements or contact information.
    Example:

```
<footer>
  <p>© 2025 My Website. All rights reserved.</p>
</footer>
```

12. **`<section>` (Section) :** The `<section>` tag is used to define a section of content in a document. It can be used to group related content together.
    Example:

```
<section>
  <h2>Section Title</h2>
  <p>Content for this section goes here.</p>
</section>
```

13. **`<article>` (Article) :** The `<article>` tag is used to define a self-contained piece of content that can be distributed or reused, such as blog posts, news articles, or forum posts.
    Example:

```
<article>
  <h2>Article Title</h2>
  <p>Content of the article goes here.</p>
</article>
```

14. **`<aside>` (Aside) :** The `<aside>` tag is used to represent content that is tangentially related to the main content, often used for sidebars or pull quotes.
    Example:

```
<aside>
  <h3>Related Articles</h3>
  <ul>
    <li>Article 1</li>
    <li>Article 2</li>
  </ul>
</aside>
```

15. **`<nav>` (Navigation)** : The `<nav>` tag is used to define navigation links or menus. It helps search engines and assistive technologies recognize sections of links for navigation.
    Example:

```
<nav>
  <ul>
    <li><a href="#home">Home</a></li>
    <li><a href="#services">Services</a></li>
  </ul>
</nav>
```

16. **'<pre>' (pre-formatted text) :** The `<pre>` tag is used to define preformatted text. Text within a `<pre>` element is displayed in a fixed-width font, and whitespace is preserved.
Example:

```
<pre>
  This is preformatted text.
  It preserves spaces and line breaks.
</pre>
```

HTML Lists : Unordered, ordered and description list, nested list

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

 − An unordered list. This will list items using plain bullets.

 − An ordered list. This will use different schemes of numbers to list your items.

<dl> − A description list. This arranges your items in the same way as they are arranged in a dictionary.

HTML Unordered Lists

An unordered list is a collection of related items that have no special order or sequence. This list is created by using HTML <ul> tag. Each item in the list is marked with a bullet. We can use type attribute for <ul> tag to specify the type of bullet we like. By default, it is a disc.

<ul type="square">
<ul type="disc">
<ul type="circle">

Example:

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

This will produce the following result −

- Beetroot
- Ginger
- Potato
- Radish

HTML Ordered Lists

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

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

<ol type = "1"> - Default-Case Numerals.
<ol type = "I"> - Upper-Case Numerals.
<ol type = "i"> - Lower-Case Numerals.
<ol type = "A"> - Upper-Case Letters.
<ol type = "a"> - Lower-Case Letters.

Example:

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

This will produce following output :

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

HTML Description Lists

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

(XHTML (EXtensible HyperText Markup Language) is the next step to evolution of internet. It was developed by World Wide Web Consortium (W3C). It helps web developers to make the transition from HTML to XML)

Description List makes use of following three tags.

<dl> − Defines the start of the list
<dt> − A term
<dd> − Term description
</dl> − Defines the end of the list

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>HTML Description List</title>
   </head>
   <body>
      <dl>
         <dt><b>HTML</b></dt>
         <dd>This stands for Hyper Text Markup Language</dd>
         <dt><b>HTTP</b></dt>
         <dd>This stands for Hyper Text Transfer Protocol</dd>
      </dl>
   </body>
</html>
```

This will produce following output

HTML
This stands for Hyper Text Markup Language

HTTP
This stands for Hyper Text Transfer Protocol

Nested Lists

HTML nested lists simply mean a list inside another list. It can be an unordered list or an ordered list

```
<!DOCTYPE html>
<html>
      <head>
            <title>HTML Nested List</title>
      </head>
      <body>
            <h2>A Nested List</h2>
            <p>Lists can be nested (list inside list):</p>
            <ol>
                  <li>Coffee
                        <ul>
                              <li>Cappuccino</li>
                              <li>Americano</li>
                              <li>Latte</li>
                        </ul>
                  </li>
                  <li>Tea
                        <ol type="a">
                              <li>Black tea</li>
                              <li>Green tea</li>
                        </ol>
                  </li>
                  <li>Milk</li>
            </ol>
      </body>
</html>
```

Another Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Nested List</title>
  </head>
  <body>
    <ul>
      <li>
        Recipes
        <ul>
          <li>
            Christmas Recipes
            <ul>
              <li>Rolled Sugar Cookies</li>
              <li>Sweet Potato Casserole</li>
              <li>Apple Pie</li>
```

```
            </ul>
          </li>
          <li>
            Fall Recipes
            <ul>
              <li>Pasta</li>
              <li>
                Pancakes
                <ul>
                  <li>Pancakes 1</li>
                  <li>Pankes 2</li>
                </ul>
              </li>
              <li>Pumpkin Pie</li>
            </ul>
          </li>
          <li>
            Indian Recipes
            <ul>
              <li>Butter Chicken</li>
              <li>Chicken Tikka</li>
              <li>Chicken Biryani</li>
            </ul>
          </li>
        </ul>
      </li>
      <li>Ingredients</li>
      <li>Kitchen Tips</li>
    </ul>
  </body>
</html>
```

Creating Hyperlinks

HTML Links

A hyperlink (often referred to simply as a "link") in HTML is an element that allows users to navigate from one resource (web page, file, or location) to another. In HTML, hyperlinks are created using the <a> (anchor) element. The <a> tag stands for "anchor," and it defines a clickable link to a specified destination.

It has the following syntax:

Syntax:

```
<a href="destination_url" attribute-lists> Clickable Text</a>
```

Example:

```
<a href="https://www.ncit.edu.np">Visit NCIT Website</a>
```

Key Components of a Hyperlink:

- **<a> Element:** The anchor tag used to create a hyperlink.
- **href Attribute**: Specifies the URL or the destination address where hyperlink points.
- **Link Text**: The text between the opening <a> and closing tags is the clickable text that users see and click on.

Target Attribute:

The target attribute specifies where to open the linked document.

Value	Description
_blank	Opens the linked document in a new window or tab
_self	Opens the linked document in the same frame as it was clicked (this is default)
_parent	Opens the linked document in the parent frame
_top	Opens the linked document in the full body of the window
framename	Opens the linked document in the named iframe

Syntax:

```
<a target="_blank|_self|_parent|_top|framename">
```

Types of Hyperlinks in HTML:

1. **External Hyperlinks**: These links point to external websites or resources outside of the current website or domain. The **`href`** attribute specifies the URL of the external website.

   Syntax:

   ```
   <a href="external_url" attribute-lists> Clickable Text</a>
   ```

   Example:

   ```
   <a href="https://www.google.com">Go to Google Website</a>
   ```

   Here, the link points to an external site (https://www.google.com), so clicking the link will navigate the user away from the current site.

2. **Internal Hyperlinks**: These links point to other pages or sections within the same website or domain.

   Syntax:

   ```
   <a href="external_url" attribute-lists> Clickable Text</a>
   ```

   Example:

   ```
   <a href="about.html">About Us</a>
   ```

   The link points to an internal file **about.html** within the same domain. Clicking the link will navigate to the "**About Us**" page within the same website.

3. **Anchor Links (In-Page Links)**: Anchor links allows us to link to a specific section within the same page or to another page. This is useful for long pages with multiple sections or for navigating within one document.

   Syntax:

   ```
   <a href="pageSection"> Clickable Text</a>
   ```

   Example:

   ```
   <a href="#section2">Go to Section 2</a>
   <h2 id="section2">Section 2</h2>
   ```

   The link points to a specific element with the id attribute (#section2). When the user clicks the link, the browser scrolls to that section within the same page.

4. **Mailto Links (Linking to email) :** HTML <a> tag provides us option to specify an email address to send an email. While using <a> tag as an email tag, we will use *`mailto: email address`* along with href attribute.

   Syntax:

   ```
   <a href = "mailto: email-address">Email Text</a>
   ```

   Example:

   ```
   <a href = "mailto: pradip@ncit.edu.np">Click here to email me</a>
   ```

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

   Default Settings

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

   ```
   <a href = "mailto:pradip@ncit.edu.np?subject = Feedback&body = Message">
   Send Email</a>
   ```

5. **Telephone Links**: Telephone links (tel:) allow users to click a link to call a specified phone number, usually on mobile devices.

   Syntax:

   ```
   <a href="tel:phone_number">Clickable Link</a>
   ```

   Example:

   ```
   <a href="tel:+1234567890">Call Us</a>
   ```

   Clicking this link on a mobile device will open the phone app and initiate a call to the number provided.

6. **File Links (Downloads)**: Links can point to downloadable files such as PDFs, images, or other documents. Clicking such links will either download the file or open it in the browser, depending on the browser settings.

   Syntax:

   ```
   <a href="path_to_file" download>Link to download file</a>
   ```

   Example:

   ```
   <a href="files/document.pdf" download>Download Document</a>
   ```

   The download attribute in the link tells the browser to download the file instead of opening it.

Executable Content Tags

In HTML, there are no specific "executable content tags" because HTML itself is a markup language used for structuring content, not for directly executing code. However, executable code is commonly incorporated into HTML using JavaScript and certain tags. Here are some key elements related to executing code in HTML:

1. **<script>**: This tag is used to include JavaScript, which is executable content. It allows you to add interactivity, perform calculations, and dynamically update content on a webpage.

   Example:

```
<script>
  alert("Hello, world!");
</script>
```

2. **<noscript>**: This tag is used to define content that is displayed if the user's browser does not support JavaScript or if JavaScript is disabled.

   Example:

   ```
   <noscript>
     JavaScript is required to view this page.
   </noscript>
   ```

3. **<iframe>**: This tag is used to embed another HTML document or external content. If the embedded content is a program (like a game or interactive tool), that content can be executable.

   Example:

   ```
   <iframe src="https://example.com"></iframe>
   ```

4. **<object>** and **<embed>**: These tags can be used to embed executable content like Flash, Java applets, or other types of interactive content, though modern web standards no longer widely support these (e.g., Flash is deprecated).

   Example with <object>:

   ```
   <object data="game.swf" type="application/x-shockwave-flash">
     Your browser does not support Flash.
   </object>
   ```

1.2 Images and Imagemaps

Images

Understanding Directories and Directories structure

A file path describes the location of a file in a web site's folder structure. Its like an address of a file which helps the web browser to access the files. File paths are used to link external resources such as images, videos, style sheets, JavaScript, displaying other web pages etc.

To insert a file in a web page its source must be known. For example, the syntax (<img src=" " >) is used to insert an image file, where the path of the file is mentioned in the source (src).

Path	Description
<img src="picture.jpg">	The "picture.jpg" file is located in the same folder as the current page
<img src="images/picture.jpg">	The "picture.jpg" file is located in the images folder in the current folder
<img src="/images/picture.jpg">	The "picture.jpg" file is located in the images folder at the root of the current web
<img src="../picture.jpg">	The "picture.jpg" file is located in the folder one level up from the current folder

File paths are of two types:

- Absolute File Paths
- Relative File Paths

Absolute File Paths: It describes the full address(URL) to access an internet file.

Example:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Absolute file path</title>
    </head>
    <body>
        <img src="https://ncit.edu.np/images/logo.png" alt="logo" title="NCIT">
    </body>
</html>
```

Relative File Path: It describes the path of the file relative to the location of the current web page file. Below Example shows the path of the file present in the same folder of the current web page file.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Relative file path</title>
    </head>
    <body>
        <img src="logo1.jpeg" alt="logo" title="NCIT">
    </body>
</html>
```

Below Example shows the path of the file present in the img folder in the the current folder where web page file exists.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Relative file path</title>
    </head>
    <body>
        <img src=”img/logo2.jpeg” alt=”logo” title=”NCIT”>
    </body>
</html>
```

Below Example shows the path of the file present in the a folder one level above current web page file.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Relative file path</title>
    </head>
    <body>
        <img src=”../images/logo.png” alt=”logo” title=”NCIT”>
    </body>
</html>
```

Below Example shows the file path points to a file in the images folder located at the root of the current web:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Relative file path</title>
    </head>
    <body>
        <img src=”/images/logo.png” alt=”logo” title=”NCIT”>
    </body>
</html>
```

Inserting an image

The HTML <img> tag is used to embed an image in a web page. Images are not technically inserted into a web page; images are linked to web pages. The <img> tag creates a holding space for the referenced image.

- The <img> tag is empty, it contains attributes only, and does not have a closing tag.
- The <img> tag has two required attributes:
  - src - Specifies the path to the image
  - alt - Specifies an alternate text for the image

Syntax:

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

The src Attribute :The required src attribute specifies the path (URL) to the image.

The alt Attribute : The required alt attribute provides an alternate text for an image, if the user for some reason cannot view it (because of slow connection, an error in the src attribute,

or if the user uses a screen reader). The value of the alt attribute should describe the image:

Example:

```
<img src=”img/logo2.jpeg” alt=”logo” title=”NCIT Logo”>
```

Image as a Link

We can add image as a link (other HTML elements can also be used as a link). A link is a connection from one Web page to another web page. The <a> tag defines a hyperlink and used to link from one page to another. The href attribute is used with the <a> tag, which indicates the link's destination.

To use image as a link, we just need to use image inside the hyperlink at the place of text as shown in below example.

Syntax:

```
<a href="link address" attribute-lists><img src=”image-link”></a>
```

Example:

```
<a href="https://www.facebook.com"><img src=”facebook-logo.jpeg” alt=”facebook
logo”></a>
```

Image Maps

An image map is an image with clickable areas. Image maps are images that provide multiple active regions that users can activate to navigate.

The HTML <map> tag defines an image map. An image map is an image with clickable areas. The areas are defined with one or more <area> tags.

Try to click on the computer, phone, or the cup of coffee in the image of below example:

Example:

```
<!DOCTYPE html>
<html>
    <body>
    <h2>Image Maps</h2>
    <p>Click on the computer, the phone, or the cup of coffee to go to a new page
and read more about the topic:</p>
    <img src="imagemaps/workplace.jpeg" alt="Workplace" usemap="#workmap">
    <map name="workmap">
        <area shape="rect" coords="34,44,270,350" alt="HTML hidden fields"
href="2.6.8-hidden-field.html">
        <area shape="rect" coords="290,172,333,250" alt="html relative path"
href="2.3.9-relative-paths.html">
        <area shape="circle" coords="337,300,44" alt="Cup of coffee" href="2.3.9-
relative-paths.html">
    </map>
    </body>
</html>
```

Client side image maps

Client side image maps provide multiple active regions through the browser and they can be made accessible relatively easily.

Client side image maps are enabled by the usemap attribute of the <img /> tag and defined by special

<map> and <area> extension tags. The image that is going to form the map is inserted into the page using the <img /> tag as normal image, except it carries an extra attribute called usemap. The value of the usemap attribute is the value that will be used in a <map> tag to link map and image tags. The <maps along with <area> tags define all the image coordinates and corresponding links.

The <area> tag inside the map tag specifies the shape and the coordinates to define the boundaries of each clickable hotspot available on the image.

Following is an example of client side image map:

```
<!DOCTYPE html>
<html>
    <body>
    <h2>Client Side Image Maps</h2>
    <p>Click on the computer, the phone, or the cup of coffee to go to a new page
and read more about the topic:</p>
    <img src="imagemaps/workplace.jpeg" alt="Workplace" usemap="#workmap">
    <map name="workmap">
        <area shape="rect" coords="34,44,270,350" alt="HTML hidden fields"
href="2.6.8-hidden-field.html">
        <area shape="rect" coords="290,172,333,250" alt="html relative path"
href="2.3.9-relative-paths.html">
        <area shape="circle" coords="337,300,44" alt="Cup of coffee" href="2.3.9-
relative-paths.html">
    </map>
    </body>
</html>
```

Server side image maps

Server side image maps require a mouse to be used and transfer click data to the server for processing. Server side image maps present significant accessibility challenges across a variety of disability types. Whenever possible, the use of server side image maps should be avoided.

The “ismap” attribute is a boolean attribute. When present, it specifies that the image is part of a server-side image map. When the user clicks someplace within the image, the browser passes the coordinates of the mouse pointer along with the URL specified in the <a> tag to the web server.

When clicking on a server-side image map, the click coordinates are sent to the server as a URL query string.

The ismap attribute is allowed only if the <img> element is a descendant of an <a> element with a valid href attribute.

Example:

```
<!DOCTYPE html>
<html>
    <body>
    <h2>Server Side Image Maps</h2>
    <p>Click the link below</p>
    <a href="/action_page.php" target=”_self”>
      <img src="magemaps/workplace.jpeg"" alt="Server Side image map" ismap>
    </a>
    </body>
</html>
```

1.3 Table

The HTML tables allow web authors to arrange data like text, images, links, other tables, etc. into rows and columns of cells. HTML tables are created using the <table> tag in which the <tr> tag is used to create table rows and <td> tag is used to create data cells. The elements under <td> are regular and left aligned by default.

Example:

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

Table Heading

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

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>HTML Table Header</title>
   </head>
   <body>
      <table border = "1">
         <tr>
            <th>Name</th>
            <th>Salary</th>
         </tr>
         <tr>
            <td>Ramesh Raman</td>
            <td>5000</td>
         </tr>
         <tr>
            <td>Shabbir Hussein</td>
            <td>7000</td>
         </tr>
      </table>
   </body>
</html>
```

Cellpadding and Cellspacing Attributes

There are two attributes called cellpadding and cellspacing which we will use to adjust the

white space in your table cells. The **cellspacing attribute defines space between table cells**, while **cellpadding represents the distance between cell borders and the content within a cell**.

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>HTML Table Cellpadding</title>
   </head>
   <body>
      <table border = "1" cellpadding = "5" cellspacing = "5">
         <tr>
            <th>Name</th>
            <th>Salary</th>
         </tr>
         <tr>
            <td>Ramesh Raman</td>
            <td>5000</td>
         </tr>
         <tr>
            <td>Shabbir Hussein</td>
            <td>7000</td>
         </tr>
      </table>
   </body>
</html>
```

Output:

Name	Salary
Ramesh Raman	5000
Shabbir Hussein	7000

Nested Tables

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

Example:

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

```
                  <th>Name</th>
                  <th>Salary</th>
               </tr>
               <tr>
                  <td>Ramesh Raman</td>
                  <td>5000</td>
               </tr>
               <tr>
                  <td>Shabbir Hussein</td>
                  <td>7000</td>
               </tr>
            </table>
         </td>
         <td>This is another cell</td>
      </tr>
   </table>
</body>
</html>
```

Colspan and Rowspan Attributes

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

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>HTML Table Colspan/Rowspan</title>
   </head>
   <body>
      <table border = "1">
         <tr>
            <th>Column 1</th>
            <th>Column 2</th>
            <th>Column 3</th>
         </tr>
         <tr>
            <td rowspan = "2">Row 1 Cell 1</td>
            <td>Row 1 Cell 2</td>
            <td>Row 1 Cell 3</td>
         </tr>
         <tr>
            <td>Row 2 Cell 2</td>
            <td>Row 2 Cell 3</td>
         </tr>
         <tr>
            <td colspan = "3">Row 3 Cell 1</td>
         </tr>
      </table>
   </body>
</html>
```

Output:

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

Table Caption

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

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>HTML Table Caption</title>
   </head>
   <body>
      <table border = "1" width = "100%">
         <caption>This is the caption</caption>

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

Output:

This is the caption

row 1, column 1	row 1, column 2
row 2, column 1	row 2, column 2

Table Header, Body, and Footer

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

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

1. <thead> − to create a separate table header.
2. <tbody> − to indicate the main body of the table.
3. <tfoot> − to create a separate table footer.

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

```
<!DOCTYPE html>
<html>
   <head>
      <title>HTML Table</title>
   </head>
   <body>
      <table border = "1" width = "100%">
         <thead>
            <tr>
               <td colspan = "4">This is the head of the table</td>
            </tr>
         </thead>

         <tfoot>
            <tr>
               <td colspan = "4">This is the foot of the table</td>
            </tr>
         </tfoot>

         <tbody>
            <tr>
               <td>Cell 1</td>
               <td>Cell 2</td>
               <td>Cell 3</td>
               <td>Cell 4</td>
            </tr>
         </tbody>
      </table>
   </body>
</html>
```

1.4 FramesF

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

Advantages of frames

- The main advantage of frames is that it allows the user to view multiple documents within a single Web page.
- It is possible to load pages from different servers in a single frameset.

Although there are very few advantages of frames, its not recommended to use frames in modern websites.

Disadvantages of Frames

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

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

Creating Frames

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

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>HTML Frames</title>
   </head>
   <frameset rows = "10%,80%,10%">
      <frame name = "top" src = "/html/top_frame.htm" />
      <frame name = "main" src = "/html/main_frame.htm" />
      <frame name = "bottom" src = "/html/bottom_frame.htm" />
      <noframes>
         <body>Your browser does not support frames.</body>
      </noframes>
   </frameset>
</html>
```

Example:

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

```
      <title>HTML Frames</title>
   </head>
   <frameset cols = "25%,50%,25%">
      <frame name = "top" src = "form 2.6.14-readonly.html" />
      <frame name = "main" src = "form 2.6.13-disabled.html" />
      <frame name = "bottom" src = "form 2.6.4-checkbox.html" />
      <noframes>
         <body>Your browser does not support frames.</body>
      </noframes>
   </frameset>
</html>
```

The <frameset> Tag Attributes

Following are important attributes of the <frameset> tag −

S.N.	Attribute	Description
1	cols	Specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of the four ways − Absolute values in pixels. For example, to create three vertical frames, use cols = "100, 500, 100". A percentage of the browser window. For example, to create three vertical frames, use cols = "10%, 80%, 10%". Using a wildcard symbol. For example, to create three vertical frames, use cols = "10%, *, 10%". In this case wildcard takes remainder of the window. As relative widths of the browser window. For example, to create three vertical frames, use cols = "3*, 2*, 1*". This is an alternative to percentages. You can use relative widths of the browser window. Here the window is divided into sixths: the first column takes up half of the window, the second takes one third, and the third takes one sixth.
2	rows	This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset. For example, to create two horizontal frames, use rows = "10%, 90%". You can specify the height of each row in the same way as explained above for columns.
3	border	This attribute specifies the width of the border of each frame in pixels. For example, border = "5". A value of zero means no border.
4	frameborder	This attribute specifies whether a three-dimensional border should be displayed between frames. This attribute takes value either 1 (yes) or 0 (no). For example frameborder = "0" specifies no border.
5	framespacing	This attribute specifies the amount of space between frames in a frameset. This can take any integer value. For example framespacing = "10" means there should be 10 pixels spacing between each frames.

The <frame> Tag Attributes

Following are the important attributes of <frame> tag

S.N.	Attribute	Description
1	src	This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, src = "/html/top_frame.htm" will load an HTML file available in html directory.

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

Browser Support for Frames

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

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

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

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

Nesting of Frame sets

Framesets may be nested to any level.

In the following example, the outer FRAMESET divides the available space into three equal rows. The inner FRAMESET then divides the second area into three columns of unequal width.

```
<FRAMESET rows="33%, 33%, 34%">
     <FRAMESET cols="100%">
        ...contents of first frame, and a column, it only contains one column
     </FRAMESET>
     <FRAMESET cols="*,*,*">
        ...contents of second frame, first column...
        ...contents of second frame, second column...
        ...contents of second frame, third column...
     </FRAMESET>
```

```
    <FRAMESET cols="100%">
       ...contents of third frame, and a column, it only contains one column
    </FRAMESET>
</FRAMESET>
```

Iframe

An HTML iframe is used to display a web page within a web page.

The HTML <iframe> tag specifies an inline frame. An inline frame is used to embed another document within the current HTML document.

Syntax

```
<iframe src="url" title="description" ></iframe>
```

Example

```
 <iframe src="https://www.ncit.edu.np" height="200" width="300" title="Nepal
College of IT"></iframe>
```

iframe Attributes

S.N.	Attributes	Description
1	src	This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, src = "/html/top_frame.htm" will load an HTML file available in html directory.
2	name	This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link.
3	frameborder	This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no).
4	marginwidth	This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example marginwidth = "10".
5	marginheight	This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example marginheight = "10".
6	height	This attribute specifies the height of <iframe>. default height is 150 px.
7	width	This attribute specifies the width of <iframe>. default width is 300 px.
8	scrolling	This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example scrolling = "no" means it should not have scroll bars.
9	longdesc	This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example longdesc = "framedescription.htm"
10	allowfullscreen	If true then that frame can be opened in full screen.

1.5 HTML Form

HTML Forms are required, when we want to collect some data from the site visitor. For example, during user registration we would like to collect information such as name, email address, credit card, etc.

A form will take input from the site visitor and then will post it to a back-end application such as CGI, ASP Script or PHP script etc. The back-end application will perform required processing on the passed data based on defined business logic inside the application.

There are various form elements available like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.

The HTML <form> tag is used to create an HTML form and it has following syntax −

```
<form action = "Script URL" method = "GET|POST">
   form elements like input, textarea etc.
</form>
```

Form Attributes:

S.N.	Attribute	Description
1	action	Backend script ready to process your passed data.
2	method	Method to be used to upload data. The most frequently used are GET and POST methods.
3	target	Specify the target window or frame where the result of the script will be displayed. It takes values like _blank, _self, _parent etc.
4	enctype	We can use the enctype attribute to specify how the browser encodes the data before it sends it to the server. Possible values are **- application/x-www-form-urlencoded** − This is the standard method most forms use in simple scenarios. **- mutlipart/form-data** − This is used when you want to upload binary data in the form of files like image, word file et

HTML Form Controls

There are different types of form controls that you can use to collect data using HTML form −

1. Text Input Controls
2. Checkboxes Controls
3. Radio Box Controls
4. Select Box Controls
5. File Select boxes
6. Hidden Controls
7. Button Controls

1. **Text Input Controls:** There are three types of text input used on forms −

   Single-line text input controls − This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML <input> tag.

   Attributes:

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

S. No	Attribute	Description
1	type	Indicates the type of input control and for text input control it will be set to text.
2	name	Used to give a name to the control which is sent to the server to be recognized

		and get the value.
3	value	This can be used to provide an initial value inside the control.
4	size	Allows to specify the width of the text-input control in terms of characters.
5	maxlength	Allows to specify the maximum number of characters a user can enter into the text box.

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Password Input Control</title>
   </head>
   <body>
      <form >
         User ID : <input type = "text" name = "user_id" value="" size="10"
maxlength="30"/> <br>
         User name : <input type = "text" name = "user_name" value="" size="30"
maxlength="50"/>
      </form>
   </body>
</html>
```

Multi-line text input controls − This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML <textarea> tag.

Attributes:

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

S.N.	Attribute	Description
1	name	Used to give a name to the control which is sent to the server to be recognized and get the value.
2	rows	Indicates the number of rows of text area box.
3	cols	Indicates the number of columns of text area box

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Multiple-Line Input Control</title>
   </head>
   <body>
      <form>
         Description : <br />
         <textarea rows = "5" cols = "50" name = "description" placeholder="Enter
description here..."></textarea>
      </form>
   </body>
</html>
```

Password input controls − This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTMl <input> tag.

Attributes

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

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

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Password Input Control</title>
   </head>
   <body>
      <form >
         User ID : <input type = "text" name = "user_id" value="" size="10"
maxlength="30"/>
         <br>
         Password: <input type = "password" name = "password" />
      </form>
   </body>
</html>
```

2. Checkboxes Controls

Checkboxes are used when more than one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to checkbox..

Attributes

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

S.N.	Attribute	Description
1	type	Indicates the type of input control and for checkbox input control it will be set to checkbox..
2	name	Used to give a name to the control which is sent to the server to be recognized and get the value.
3	value	The value that will be used if the checkbox is selected.
4	checked	Set to checked if you want to select it by default.

Example:

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

3. Radio Box Controls

Radio buttons are used when out of many options, just one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to radio.

Attributes:

Following is the list of attributes for radio button.

S.N.	Attribute	Description
1	type	Indicates the type of input control and for checkbox input control it will be set to radio.
2	name	Used to give a name to the control which is sent to the server to be recognized and get the value.
3	value	The value that will be used if the radio box is selected.
4	checked	Set to checked if you want to select it by default.

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Radio Box Control</title>
   </head>
   <body>
      <form>
         <input type = "radio" name = "subject" value = "maths"> Maths
         <input type = "radio" name = "subject" value = "physics" checked> Physics
      </form>
   </body>
</html>
```

4. Select Box Controls

A select box, also called drop down box which provides option to list down various options in the form of drop down list, from where a user can select one or more options.

Attributes:

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

S.N.	Attribute	Description
1	name	Used to give a name to the control which is sent to the server to be recognized and get the value.
2	size	This can be used to present a scrolling list box.
3	multiple	If set to "multiple" then allows a user to select multiple items from the menu.

Following is the list of important attributes of <option> tag −

S.N.	Attribute	Description
1	value	The value that will be used if an option in the select box box is selected.
2	selected	Specifies that this option should be the initially selected value when the page loads.
3	label	An alternative way of labeling options

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Select Box Control</title>
   </head>
   <body>
      <form>
         <select name = "dropdown" size="10" multiple>
            <option value = "Maths" selected>Maths</option>
            <option value = "Physics">Physics</option>
            <option value = "Engish">English</option>
            <option value = "Calculus">Calculus</option>
         </select>
      </form>
   </body>
</html>
```

5. File Select boxes

If you want to allow a user to upload a file to your web site, you will need to use a file upload box, also known as a file select box. This is also created using the <input> element but type attribute is set to file.

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

S.N.	Attribute	Description
1	name	Used to give a name to the control which is sent to the server to be recognized and get the value.
2	accept	Specifies the types of files that the server accepts.

Example:

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

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

6. Hidden Controls

A hidden field lets web developers include data that cannot be seen or modified by users when a form is submitted. Hidden field often stores what database record that needs to be updated when the form is submitted.

Note: While the value is not displayed to the user in the page's content, it is visible (and can be edited) using any browser's developer tools or "View Source" functionality. Do not use hidden inputs as a form of security!

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Hidden Input fields</title>
   </head>
   <body>
      <form>
         <p>This is page 10</p>
         <input type = "hidden" name = "pagename" value = "10" />
         <input type = "submit" name = "submit" value = "Submit" />
      </form>
   </body>
</html>
```

7. Buttons Control

There are various ways in HTML to create clickable buttons. You can also create a clickable button using <input>tag by setting its type attribute to button.

The type attribute can take the following values −

S.N.	Type	Description
1	submit	This creates a button that automatically submits a form.
2	reset	This creates a button that automatically resets form controls to their initial values.
3	button	This creates a button that is used to trigger a client-side script when the user clicks that button.
4	image	This creates a clickable button but we can use an image as background of the button.

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>File Upload Box</title>
```

```
   </head>
   <body>
      <form>
         <input type = "submit" name = "submit" value = "Submit" />
         <input type = "reset" name = "reset"  value = "Reset" />
         <input type = "button" name = "ok" value = "OK" />
         <input type = "image" name = "imagebutton" src = "button.png" />
      </form>
   </body>
</html>
```

HTML form tags

Tag	Description
<form>	It defines an HTML form to enter inputs by the used side.
<input>	It defines an input control.
<textarea>	It defines a multi-line input control.
<label>	It defines a label for an input element.
<fieldset>	It groups the related element in a form.
<legend>	It defines a caption for a <fieldset> element.
<select>	It defines a drop-down list.
<optgroup>	It defines a group of related options in a drop-down list.
<option>	It defines an option in a drop-down list.
<button>	It defines a clickable button.

Focus

The focus() method gives focus to an element (if it can be focused). (*We will further study it in next chapter (javascript)).*

AccessKey Property

The HTML DOM accessKey property is used to set the accessKey attribute of an element. However, the accesskey attribute in HTML is used to set a shortcut key to activate or focus on an element.

Example:

```
<a id="myid" accesskey="g" href="https://www.google.com/">Google</a>
```

Tabindex

This attribute is used to specify the tab order of an element. It is used when the tab button is used for navigating. It supports all HTML element.

Syntax:

<element tabindex = "number">

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Tabindex</title>
   </head>
   <body>
        <div>
        first Name : <input type = "text" name = "fname" value="" tabindex="1"/>
        <br>
        Last Name : <input type ="text" name ="lname" value="" tabindex="2"/>
        <br>
        User ID : <input type ="text" name="user_id" value="" tabindex="5"/><br>
        Password: <input type = "password" name = "password" tabindex="4" /><br>
    </div>
   </body>
</html>
```

Disabled and Read Only Controls

Disabled Control

The disabled attribute is a boolean attribute. When present, it specifies that the element should be disabled. A disabled element is unusable.

The disabled attribute can be used on the following elements:

Elements	**Attribute**
<button>	disabled
<fieldset>	disabled
<input>	disabled
<optgroup>	disabled
<option>	disabled
<select>	disabled
<textarea>	disabled

Example:

```
<button type="button" disabled>Click Me!</button>
```

Form example with disabled input field

```
<form action="/action_page.php">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname" disabled><br>
  <input type="submit" value="Submit">
</form>
```

Disabled <input> elements in a form will not be submitted!

Read Only Control

The readonly attribute is a boolean attribute. When present, it specifies that an element/field is read-only.

A read-only input field cannot be modified (however, a user can tab to it, highlight it, and copy the text from it).

The readOnly property makes the element non-editable but it can still be focused by tab or click. If there is a default value inside a read-only element then it is sent to a server on submit.

Example:

```
<input type="text" name="lname" value="bhattarai" readonly>
```

A form will still submit an input field that is readonly, but will not submit an input field that is disabled!

Form Methods

The two most common HTTP methods are: **GET** and **POST**.

(Don't get confused with the http methods - GET, POST, PUT, HEAD, DELETE, PATCH, OPTIONS, CONNECT, TRACE)

Before the browser sends the information, it encodes it using a scheme called URL encoding. In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand.

name1=value1&name2=value2&name3=value3

Spaces are removed and replaced with the + character and any other non alphanumeric characters are replaced with a hexadecimal values. After the information is encoded it is sent to the server.

The GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the ? Character.

http://www.test.com/index.htm?name1=value1&name2=value2

- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.

- The data sent by GET method can be accessed using QUERY_STRING environment variable.
- The PHP provides $_GET associative array to access all the sent information using GET method.

```
<?php
   if( $_GET["name"] || $_GET["age"] ) {
      echo "Welcome ". $_GET['name']. "<br />";
      echo "You are ". $_GET['age']. " years old.";
      exit();
   }
?>
<html>
   <body>
      <form action = "<?php $_PHP_SELF ?>" method = "GET">
         Name: <input type = "text" name = "name" />
         Age: <input type = "text" name = "age" />
         <input type = "submit" />
      </form>
   </body>
</html>
```

The Post Method

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides $_POST associative array to access all the sent information using POST method.

```
<?php
   if( $_POST["name"] || $_POST["age"] ) {
      echo "Welcome ". $_POST['name']. "<br />";
      echo "You are ". $_POST['age']. " years old.";
      exit();
   }
?>
<html>
   <body>
      <form action = "<?php $_PHP_SELF ?>" method = "POST">
         Name: <input type = "text" name = "name" />
         Age: <input type = "text" name = "age" />
         <input type = "submit" />
      </form>
   </body>
</html>
```

1.6 Style Sheets

Introduction to stylesheets

Stylesheets are a crucial component of web development that allow developers to control the presentation and layout of web pages. They are written in languages like CSS (Cascading Style Sheets) and are responsible for styling HTML (Hypertext Markup Language) documents.

Here's an explanation of stylesheets:

1. **Purpose:**

   Separation of Concerns: Stylesheets separate the content (HTML) from its presentation (CSS). This makes it easier to manage and maintain a website's design.

2. **Languages:**
   - **CSS (Cascading Style Sheets)**: The most common stylesheet language. It defines how HTML elements are displayed on the screen, in print, or other media.
   - **Sass (Syntactically Awesome Stylesheets):** A preprocessor scripting language that is interpreted or compiled into CSS.
   - **LESS**: Similar to Sass, it is also a CSS preprocessor.

3. **Syntax**:
   - **Selectors**: Elements or groups of elements that you want to style.
   - **Properties**: Define the style rules for the selected elements.
   - **Values**: Specify the style property values.

   Example:

```
/* Selector */
h1 {
      /* Property: Value */
      color: blue;
      font-size: 24px;
}
```

4. **Cascading**:

   Inheritance: Styles can be inherited from parent elements to their children.

   Specificity: Rules can be more or less specific, affecting their priority in the rendering.

5. **Selectors**:
   - **Element Selectors**: Target specific HTML elements (e.g., p, h1, div).
   - **Class Selectors**: Target elements with a specific class (e.g., .header).
   - **ID Selectors**: Target a specific element with a unique ID (e.g., #main-content).
   - **Attribute Selectors**: Target elements with a specific attribute (e.g., [type="text"]).

6. **Box Model**:
   - Content: The actual content of the box.
   - Padding: Clears an area around the content inside the box.
   - Border: A border surrounding the padding.
   - Margin: Clears an area outside the border.

7. **Layout:**
   - **Positioning**: Determines how an element is positioned in a document.

- **Flexbox**: A one-dimensional layout method for laying out items in rows or columns.
- **Grid**: A two-dimensional layout system for the web.

8. **Responsive Design**:
   - **Media Queries**: Used to apply different styles for different devices or screen sizes.
   - **Viewport Units**: Units relative to the viewport dimensions (e.g., vw for viewport width).
9. **Animations and Transitions:**
   - **Keyframes**: Define the stages and styles of an animation.
   - **Transitions**: Smoothly animate changes between CSS property values.
10. **Vendor Prefixes:** Used to add browser-specific CSS rules to ensure compatibility.
11. **Importing Stylesheets:** Stylesheets can be linked externally using the <link> tag or included within HTML using the <style> tag.
12. **Frameworks:** Libraries like Bootstrap or Foundation provide pre-designed and pre-coded stylesheets for easier and faster development.
13. **Performance:**
    - **Minification**: Removing unnecessary characters from the stylesheet to reduce file size.
    - **Concatenation**: Combining multiple stylesheets into one to reduce HTTP requests.
14. **Tools:**
    - **DevTools:** Browser tools for inspecting and debugging styles.
    - **Preprocessors:** Tools like Sass and LESS to enhance CSS with variables, functions, etc.
15. **Best Practices:**

- **Consistency**: Maintain a consistent naming convention and style across the project.
- **Modularity**: Break styles into manageable, reusable components.

Stylesheets play a crucial role in creating visually appealing and responsive web pages by providing a means to control the look and feel of the content. They enable developers to create a consistent and user-friendly experience across various devices and screen sizes.

Introduction to Cascading Style Sheets (CSS)

CSS is an acronym for Cascading Style Sheet and it was created by HÃƒÂ¥kon Wium Lie in 1996 for the web developers to change the layout, colors, and fonts of their websites.

CSS is a style sheet language used for describing the look and formatting of a document written in a markup language like HTML or XML. CSS allows us to control the appearance of web pages by defining styles for various elements. It can control the layout of multiple web pages all at once.

Advantages of CSS

Cascading Style Sheets (CSS) offer several advantages for web development, helping to separate the structure and presentation of a web page from its content. Here are some key advantages of using CSS:

1. **Separation of Concerns:** CSS allows for a clear separation of content (HTML) and presentation (CSS). This separation makes it easier to maintain and update a website since changes to the design can be made without altering the HTML structure.
2. **Consistency:** With CSS, you can apply consistent styling across multiple pages or an entire website. This ensures a uniform look and feel, making the site more professional and user-friendly.
3. **Reusability:** Styles defined in CSS can be reused across different pages, reducing redundancy and making it more efficient to manage styles. This leads to a more modular and maintainable codebase.
4. **Easy Maintenance:**Making changes to the design or layout of a website is simplified with CSS. Since styles are centralized in separate style sheets, updating the appearance of a site can be done globally by modifying a single file.
5. **Bandwidth Efficiency:**CSS files can be cached by the browser, resulting in faster load times for subsequent visits to a website. This is because the styles are stored locally, reducing the need to download them again from the server.
6. **Responsive Design:**CSS provides features like media queries, which enable the creation of responsive designs. With responsive design, a website can adapt its layout and styling based on the device screen size, providing a better user experience on various devices.
7. **Print-friendly Styles:**CSS allows you to define separate styles for printing. This ensures that a web page can be printed with a layout optimized for paper, avoiding unnecessary elements like navigation bars and background colors.
8. **Accessibility:**CSS supports the creation of accessible designs by allowing developers to use semantic HTML and apply styles accordingly. This ensures that the content is well-structured and can be easily consumed by users with disabilities or using assistive technologies.
9. **Search Engine Optimization (SEO):**By using CSS to format content and structure, web developers can create cleaner and more semantically meaningful HTML. This can positively impact SEO, as search engines tend to favor well-organized and descriptive content.
10. **Browser Compatibility:** CSS helps in achieving better cross-browser compatibility. By using a consistent set of styles, developers can create a unified experience for users across different browsers and platforms.

CSS Rules and Syntax:

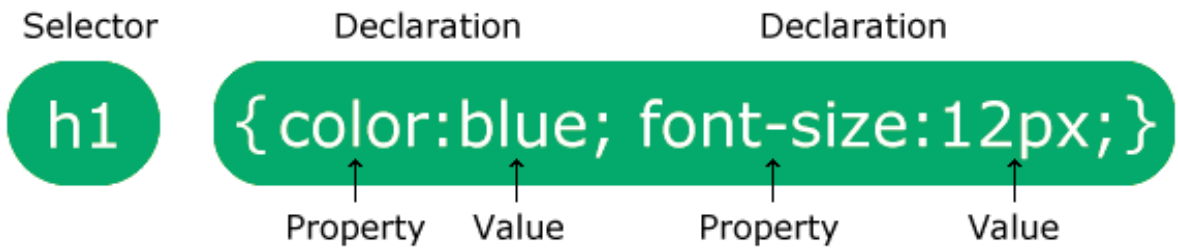
A CSS Syntax rule consists of a selector, property, and its value. The selector points to the HTML element where the CSS style is to be applied. The CSS property is separated by semicolons. It is a combination of the selector name followed by the property: value pair that is defined for the specific selector.

Syntax:

```
selector { Property: value; }
```

Example:

```
h1{color:blue; font-size:12px;}
```



CSS Selectors:

The selector points to the HTML element we want to style.

CSS selectors are patterns that select and style HTML elements based on their attributes, IDs, classes, etc. Examples include element selectors (p, div), class selectors (.classname), ID selectors (#idname), attribute selectors ([attribute=value]), and more.

CSS Declaration:

The declaration contains CSS property name and a value, separated by a colon. This block contains one or more declarations separated by semicolons. Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

Example

In this example all <p> elements will be center-aligned, with a red text color:

```
p {
      color: red;
      text-align: center;
}
```

Example Explained

p is a selector in CSS (it points to the HTML element we want to style: <p>).

color is a property, and red is the property value

text-align is a property, and center is the property value

4.2 Creating Simple stylesheets, Adding Comments in stylesheets, ...

CSS Comments

CSS comments are not displayed in the browser (ignored by browsers), but they can help document our source code. They are used to explain the code, and may help when we edit the source code at a later date.
A CSS comment is placed inside the <style> element, and starts with /* and ends with */:

Example:

```
/* This is a single-line comment */
p {
  color: red;
}
```

We can use /* ..... */ multi line comments on css in similar way we do in C and C++ programming languages.

CSS Selectors

1. **The Tag/Type Selector:** We can apply styles to HTML elements with particular attributes. The style rule below will match all the input elements having a type attribute with a value of text.
   Example:

   ```
   p {
         text-align: center;
         color: red;
   }
   ```

2. **The ID Selector**: The id selector uses the id attribute of an HTML element to select a specific element. The id of an element is unique within a page, so the id selector is used to select one unique element. To select an element with a specific id, write a hash (#) character, followed by the id of the element.
   Note:
   - An id name cannot start with a number
   - The id of an element is unique within a page.

   Example:

   ```
   /* CSS rule below will be applied to the HTML element with “myHeader” id */
   # myHeader {
      text-align: center;
      color: black;
   }

   /* This rule renders the content in myHeader for only <h1> elements with id
   attribute set to myHeader. */
   h1#myHeader {
         text-align: center;
         color: #000000;
   }

   /* This rule renders the content in myHeader for all level 2 headings. */
   #myHeader h2{
         text-align: center;
         color: #000000;
   }
   ```

3. **The Class Selector:** We can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule. To select elements with a specific class, write a period (.) character, followed by the class name. Same class name can be specified in multiple elements.
   Example:

```
/* This rule renders the content in the element with class “black”. All the elements having that class will be formatted according to the defined rule. */
.black {
      color: #000000;
}

/* This rule renders the content in the element with class “black” for only <h1> element. */
h1.black {
      color: #000000;
}

/* We can apply more than one class selectors to given element.  */
<p class = "center bold">
   This para will be styled by the classes center and bold.
</p>
```

4. **The Attribute Selector**: We can also apply styles to HTML elements with particular attributes.
   Example:
   The style rule below will match all the input elements having a type attribute with a value of text

```
input[type = "text"] {
   color: #000000;
}
```

   The advantage to this method is that the <input type = "submit" /> element is unaffected, and the color applied only to the desired text fields.

   There are following rules applied to attribute selector.
   - p[lang] − Selects all paragraph elements with a lang attribute.
   - p[lang="fr"] − Selects all paragraph elements whose lang attribute has a value of exactly "fr".
   - p[lang~="fr"] − Selects all paragraph elements whose lang attribute contains the word "fr".
   - p[lang|="en"] − Selects all paragraph elements whose lang attribute contains values that are exactly "en", or begin with "en-".

5. **Grouping Selectors:** You can apply a style to many selectors if you like. Just separate the selectors with a comma, as given in the following example −

```
h1, h2, h3 {
      color: #36C;
      font-weight: normal;
      letter-spacing: .4em;
      margin-bottom: 1em;
      text-transform: lowercase;
}
```

This define style rule will be applicable to h1, h2 and h3 element as well. The order of the list is irrelevant. All the elements in the selector will have the corresponding declarations applied to them.

We can combine the various id selectors together as shown below −

```
#content, #footer, #supplement {
      position: absolute;
      left: 510px;
      width: 200px;
}
```

6. **The Descendant Selectors** : Suppose we want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, style rule will apply to element only when it lies inside <ul> tag.
   Example:
```
ul em {
   color: #000000;
}
```

7. **The child Selector**: We have seen the descendant selectors. There is one more type of selector, which is very similar to descendants but have different functionality.
   =Consider the following example −
```
body > p {
   color: #000000;
}
```
   This rule will render all the paragraphs in black if they are direct child of <body> element. Other paragraphs put inside other elements like or <td> would not have any effect of this rule.

8. **Universal Selector:** Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type −
```
* {
   color: #000000;
}
```
   This rule renders the content of every element in our document to black color.

CSS Pseudo-classes

A CSS pseudo-class is a keyword added to a selector that specifies a special state of the selected element(s).

Syntax:
```
selector:pseudo-class {
   property: value;
}
```
Example:
```
button:hover {
  color: blue;
}
```
In above example, the pseudo-class :hover can be used to select a button when a user's pointer hovers over the button and this selected button can then be styled with color blue;

Commonly used pseudo-classes in CSS

The following table shows the most commonly used pseudo-classes in CSS:

Pseudo-class	Description
:link	Adds special style to an unvisited link.
:visited	Adds special style to a visited link.
:hover	Adds special style to an element when you mouse over it.
:active	Adds special style to an active element.

:focus	Adds special style to an element while the element has focus.

Other Pseudo classes

Selector	Example	Example description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(language)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
:not(selector)	:not(p)	Selects every element that is not a <p> element
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent
:optional	input:optional	Selects <input> elements with no "required" attribute
:out-of-range	input:out-of-range	Selects <input> elements with a value outside a specified range

:read-only	input:read-only	Selects <input> elements with a "readonly" attribute specified
:read-write	input:read-write	Selects <input> elements with no "readonly" atttribute
:required	input:required	Selects <input> elements with a "required" attribute specified
:root	root	Selects the document's root element
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all <input> elements with a valid value
:visited	a:visited	Selects all visited links

CSS Pseudo-elements

A CSS pseudo-element is a keyword added to a selector that specifies a special parts of an element. For example, it can be used to:

- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

Syntax:

```
selector::pseudo-element {
  property: value;
}
```

Example:

```
p::first-line {
  color: #ff0000;
  font-variant: small-caps;
}
```

Commonly used pseudo-elements in CSS

The following table shows the most commonly used pseudo-elements in CSS:

Pseudo-element	Description
::first-line	Selects the first line of a block level element.
::first-letter	Applies styles to the first letter of the first line of a block level element.
::after	Selects a pseudo-element that is the last child of the selected element.
::before	Selects a pseudo-element that is the first child of the selected element.
::selection	Selects a pseudo-element that is selected by a user.
::marker	Selects a pseudo-element that is markers of list items. (marker is the first part of the list that is generated for the list item)

Linking stylesheets

CSS can be added to HTML documents in 4 ways

1. **Inline CSS:**

Usage: Inline styles are applied directly within the HTML document, using the style attribute of an HTML element.

Example:

```
<p style="color:red; padding:30px;">Paragraph with inline styles.</p>
```

Pros and Cons: Simple to implement, but can lead to code duplication and may be harder to maintain for larger projects.

2. **Internal or Embedded CSS:**

Usage: Internal styles are defined within the <style> tag in the head section of an HTML document.

Example:

```
<head>
      <style>
            p {
                  color: blue;
                  font-size: 18px;
            }
      </style>
</head>
<body>
      <p>This is a paragraph with internal styles.</p>
</body>
```

Pros and Cons: More organized than inline styles, but still affects only the specific document where it's defined.

3. **External CSS:**

Usage: External styles are stored in a separate CSS file and linked to the HTML document using the <link> tag.

Example:

```
<!-- Filename : index.html -->
      <html
            <head>
                  <link rel="stylesheet" type="text/css" href="styles.css">
            </head>
            <body>
                  <p class="highlight">Paragraph with external styles.</p>
            </body>
   <!-- Filename : styles.css -->
      .highlight {
         color: green;
         font-size: 20px;
      }
```

Pros and Cons: Promotes code reusability and easier maintenance. Changes in styling can be applied across multiple pages by modifying a single external CSS file.

4. **Import CSS - @import:**

Usage: The @import rule allows us to import a style sheet into another style sheet. The @import rule must be at the top of the document (but after any @charset declaration).The @import rule also supports media queries, so we can allow the import to be media-dependent.

Example:

```
<@import "printstyle.css" print;
/* Import the "printstyle.css" style sheet ONLY if the media is print:*/
@import "mobstyle.css" screen and (max-width: 768px);
/* Import the "mobstyle.css" style sheet ONLY if the media is screen and
the viewport is maximum 768 pixels:*/
```

Pros and Cons: Promotes code reusability and easier maintenance. CSS can be imported from external sources also.

CSS Rules Overriding (Rule Priority)

We have discussed four ways to include style sheet rules in a HTML document. Here is the rule to override any Style Sheet Rule.

1. Any inline style sheet takes highest priority. So, it will override any rule defined in <style>….</style> tags or rules defined in any external style sheet file.
2. Any rule defined in <style>….</style> tags will override rules defined in any external style sheet file.
3. Any rule defined in external style sheet file takes the lowest priority, and rules defined in this file will be applied only when the above two rules are not applicable.

Creating CSS File

To create a CSS (Cascading Style Sheets) file, you can follow these steps. I'll provide a simple example to illustrate each step:

Step 1: Create a New File : First step is creating a new file using a text editor of our choice. We can use Notepad, Visual Studio Code, Sublime Text, or any other text editor of our preference.

Step 2: Save the File with a .css Extension : Save the file with a .css extension. This extension indicates that the file contains CSS code.

Step 3: Write CSS Code: Inside the file, you can write your CSS code. Here's a simple example:

```
/* styles.css */
body {
    font-family: 'Arial', sans-serif;
    background-color: #f2f2f2;
    padding: 0;
}
header {
    background-color: #333;
    text-align: center;
    padding: 10px;
}
h1 {
    color: #ff9900;
}
.container {
    width: 80%;
    margin: 0 auto;
}
p {
    line-height: 1.5;
}
```

Step 4: Link the CSS File to HTML : Create an HTML file and link the CSS file to it. Here's a simple HTML example:

```
<!-- index.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Website</title>
```

```
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <section class="heading">
        <h1>Welcome to My Website</h1>
    </section>
    <section class="container">
        <p>This is a simple example of using CSS in a web page.</p>
    </div>
</body>
</html>
```

Step 5: Save and Open HTML File

Save the HTML file and open it in a web browser. You should see the styles applied from your CSS file.

This is a basic example, and you can expand and customize your CSS code based on your project requirements. As your project grows, you may want to organize your CSS code into separate files or use preprocessors like Sass or Less for more advanced features.

Linking Multiple CSS file to HTML

```
<!DOCTYPE html>
<html>
   <head>
      <title>HTML External CSS</title>
      <!-- External CSS-->
      <link rel = "stylesheet" type = "text/css" href = "/html/style.css">
      <!-- Internal or embeded CSS-->
      <style type = "text/css">
         .red {
            color: red;
         }
         .thick{
            font-size:20px;
         }
         .green {
            color:green;
         }
      </style>
   </head>
   <body>
      <p class = "red" style="font-size:20px;">This is red</p>
      <p class = "thick">This is thick</p>
      <p class = "green">This is green</p>
      <p class = "thick green">This is thick and green</p>
   </body>
</html>
```

CSS-Fonts

A font is a set of text characters with a consistent design and style. It includes the shape, size, weight, and other attributes of the characters in a typeface.

Choosing the right font and right font formatting for our website is very important.

Font Family

In CSS there are five generic font families:

- **Serif** fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.
- **Sans-serif** fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.
- **Monospace fonts** - here all the letters have the same fixed width. They create a mechanical look.
- **Cursive** fonts imitate human handwriting.
- **Fantasy** fonts are decorative/playful fonts.

All the different font names belong to one of the generic font families.

Some Font Examples

Generic Font Family	Examples of Font Names
Serif	Times New Roman Georgia Garamond
Sans-serif	Arial Verdana Helvetica
Monospace	Courier New Lucida Console Monaco
Cursive	Brush Script MT Lucida Handwriting
Fantasy	COPPERPLATE Papyrus

Difference Between Serif and Sans-serif Fonts

Example:

```
<html>
   <head>
      <title>CSS Font Family</title>
      <style>
        .p1 {
            font-family: "Times New Roman", Times, serif;
        }
        .p2 {
            font-family: Arial, Helvetica, sans-serif;
        }
        .p3 {
```

```
            font-family: "Lucida Console", "Courier New", monospace;
        }
      </style>
   </head>

   <body>
    <h1>CSS font-family</h1>
    <p class="p1">This is a paragraph, shown in the Times New Roman font.</p>
    <p class="p2">This is a paragraph, shown in the Arial font.</p>
    <p class="p3">This is a paragraph, shown in the Lucida Console font.</p>
    </body>
</html>
```

Font Style

The font-style property is mostly used to specify italic text.

This property has three values:

- **normal** - The text is shown normally
- **italic** - The text is shown in italics
- **oblique** - The text is "leaning" (oblique is very similar to italic, but less supported)

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p.normal {
        font-style: normal;
      }
      p.italic {
        font-style: italic;
      }
      p.oblique {
        font-style: oblique;
      }
    </style>
  </head>
<body>

<h1>The font-style property</h1>
  <p class="normal">This is a paragraph in normal style.</p>
  <p class="italic">This is a paragraph in italic style.</p>
  <p class="oblique">This is a paragraph in oblique style.</p>
</body>
</html>
```

Font Size

The font-size property sets the size of the text.

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Font Size Example</title>
    <style>
      h1 {
        font-size: 40px;
      }
      h2 {
```

```
        font-size: 30px;
      }
      p {
        font-size: 14px;
      }
    </style>
  </head>
  <body>
      <h1>This is heading 1</h1>
      <h2>This is heading 2</h2>
      <p>This is a paragraph.</p>
      <p>This is another paragraph.</p>
  </body>
</html>
```

CSS Font Weight

Font weight refers to the thickness or boldness of the characters. Different weights within a font family can be used to create visual hierarchy or emphasis within a text.

The possible values for font-weight are as follows:

- **normal**: Font weight normal. Equivalent to 400.
- **bold**: Font weight bold. Equivalent to 700.
- <number>: Value between 1 and 1000 define thickness of font. Higher numbers represent weights that are bolder than (or as bold as) lower numbers.
- **lighter**: Lighter font weight relative to the parent element's font weight.
- **bolder**: Bolder font weight relative to the parent element's font weight.

Example:

```
<html>
   <head>
      <title>CSS Font Weight</title>
      <style>
        p {
            padding: 5px;
            border: 2px solid blue;
         }
        .p1 {
            font-weight: normal;
        }
        .p2 {
            font-weight: bold;
        }
        .p3 {
            font-weight: 500;
        }
      </style>
   </head>
   <body>
    <h1>CSS font-Weight</h1>
      <p class="p1">This is a paragraph, font weight "normal".</p>
      <p class="p2">This is a paragraph, font weight "bold".</p>
      <p class="p3">This is a paragraph, font weight "500".</p>
    </body>
</html>
```

CSS font-variant

The font-variant property specifies whether or not a text should be displayed in a small-caps font.

Value	Description
normal	The browser displays a normal font. This is default
small-caps	The browser displays a small-caps font
initial	Sets this property to its default value (even if its parent is changed). See initials
inherit	Inherits this property from its parent element.

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>CSS Font Variant</title>
      <style>
         p.normal {
            font-variant: normal;
         }
         p.small {
            font-variant: small-caps;
         }
      </style>
   </head>
<body>
<h1>The font-variant Property</h1>
   <p class="normal">My name is Pradip.</p>
   <p class="small">My name is Pradip.</p>
</body>
</html>
```

Font Stretch

The font-stretch property allows us to make text narrower (condensed) or wider (expanded).

Syntax:

```
font-stretch: ultra-condensed|extra-condensed|condensed|semi-condensed|normal|
semi-expanded|expanded|extra-expanded|ultra-expanded|initial|inherit;
```

Example:

```
div {
  font-family: sans-serif, "Helvetica Neue", "Lucida Grande", Arial;
  font-stretch: expanded;
}
```

Value	Description
ultra-condensed	Makes the text as narrow as it gets
extra-condensed	Makes the text narrower than condensed, but not as narrow as ultra-condensed
condensed	Makes the text narrower than semi-condensed, but not as narrow as extra-condensed
semi-condensed	Makes the text narrower than normal, but not as narrow as condensed
normal	Default value. No font stretching.
semi-expanded	Makes the text wider than normal, but not as wide as expanded
expanded	Makes the text wider than semi-expanded, but not as wide as extra-expanded

extra-expanded	Makes the text wider than expanded, but not as wide as ultra-expanded
ultra-expanded	Makes the text as wide as it gets
initial	Sets this property to its default value. Read about initial
inherit	Inherits this property from its parent element. Read about inherit

Example:
//create your own example

CSS-Text
CSS has a lot of properties for formatting text.

Text Color
The color property is used to set the color of the text. The color is specified by:

- a color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

Example:
```
<!DOCTYPE html>
<html>
    <head>
        <style>
            body {
                color: blue;
            }
            h1 {
                color: green;
            }
        </style>
    </head>
<body>
    <h1>This is heading 1</h1>
    <p>This is an ordinary paragraph. Notice that this text is blue. The default
text color for a page is defined in the body selector.</p>
    <p>Another paragraph.</p>
</body>
</html>
```

CSS Box Model
All HTML elements can be conceptualized as boxes. The term "box model" used to describe design and layout in CSS. The CSS box model essentially encapsulates a box around each HTML element, comprising content, padding, borders, and margins.

In any web page, various components are represented as one or more rectangular boxes. The CSS box model serves as a compartment housing multiple elements, including edges, borders, padding, and content. Its primary purpose is to shape and structure a web page's design. This model serves as a toolkit, allowing customization of the layout for different components. According to the CSS box model, each element is presented by the web browser as a square prism.

The CSS box model contains the different properties in CSS. These are listed below.

- **Border** : It is a region between the padding-box and the margin. Its proportions are determined by the width and height of the boundary.
- **Margin**: This segment consists of the area between the boundary and the edge of the

border. The proportion of the margin region is equal to the margin-box width and height. It is better to separate the product from its neighbor nodes.

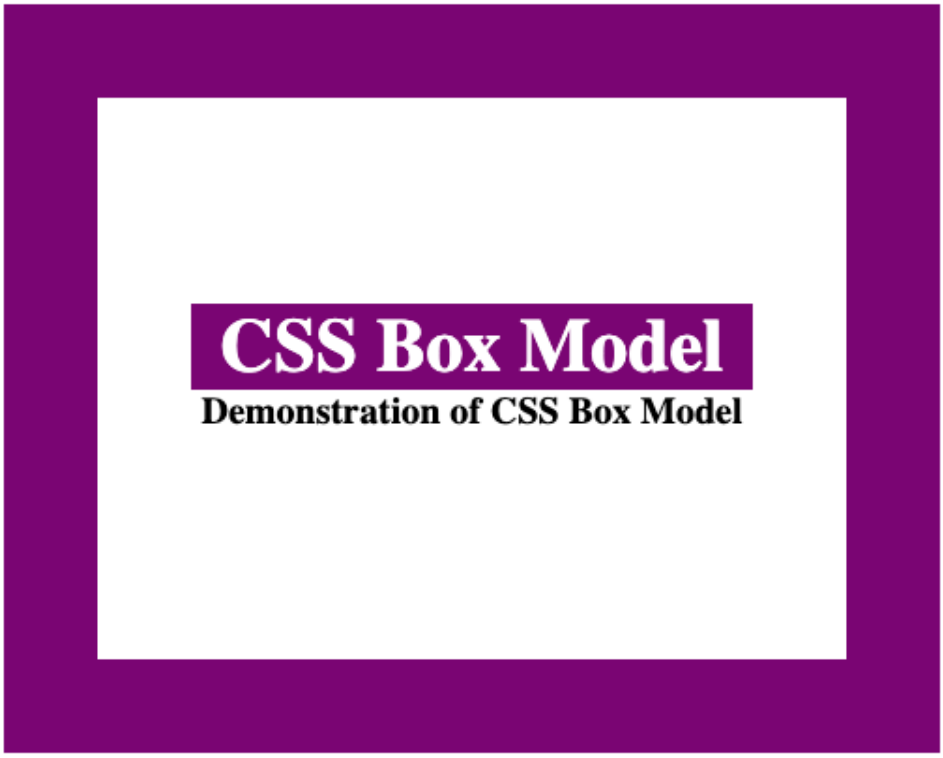
- **Padding** : This field requires the padding of the component. In essence, this area is the space around the subject area and inside the border-box. The height and the width of the padding box decide its proportions.
- **Content** : Material such as text, photographs, or other digital media is included in this area. It is constrained by the information edge, and its proportions are dictated by the width and height of the content enclosure.

Example:

```
<!DOCTYPE html>
<head>
    <title>CSS Box Model</title>
    <style>
        .main{
            font-size:30px;
            font-weight:bold;
            Text-align:center;
        }
        .box{
            margin-left:50px;
            border:50px solid Purple;
            width:300px;
            height:200px;
            text-align:center;
            padding:50px;
            font-weight:bold;
        }
        .box1{
            font-size:40px;
            color:rgb(255, 255, 255);
            margin-top:60px;
            background-color:purple;
        }
        .box2{
            font-size:20px;
            background-color:white;
        }
    </style>
</head>
      <body>
            <div class = "main">CSS Box-Model Property</div>
            <div class = "box">
                  <div class = "box1">CSS Box Model</div>
                  <div class = "box2">Demonstration of CSS Box Model</div>
```

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

Output:



References

- https://www.javatpoint.com/html-meta-tag
- https://oinam.github.io/entities
- https://www.tutorialspoint.com/html/html_fonts.htm
- https://www.w3schools.com/html/html_filepaths.asp
- https://www.w3schools.com/html/html_images_imagemap.asp
- https://www.w3schools.com/tags/att_area_coords.asp
- https://graphicscreator.laughingbirdsoftware.com/png-graphics-advantages-and-disadvantages/
- https://www.tutorialspoint.com/html/html_tables.htm
- https://www.tutorialspoint.com/html/html_forms.htm
- https://www.javatpoint.com/html-form
- https://www.tutorialspoint.com/php/php_get_post.htm
- https://www.tutorialspoint.com/html/html_frames.htm
- https://www.geeksforgeeks.org/how-to-include-frameset-inside-another-frameset-in-html/
- https://www.javatpoint.com/html-5-tags
- https://www.geeksforgeeks.org/explain-the-form-new-input-types-in-html5/