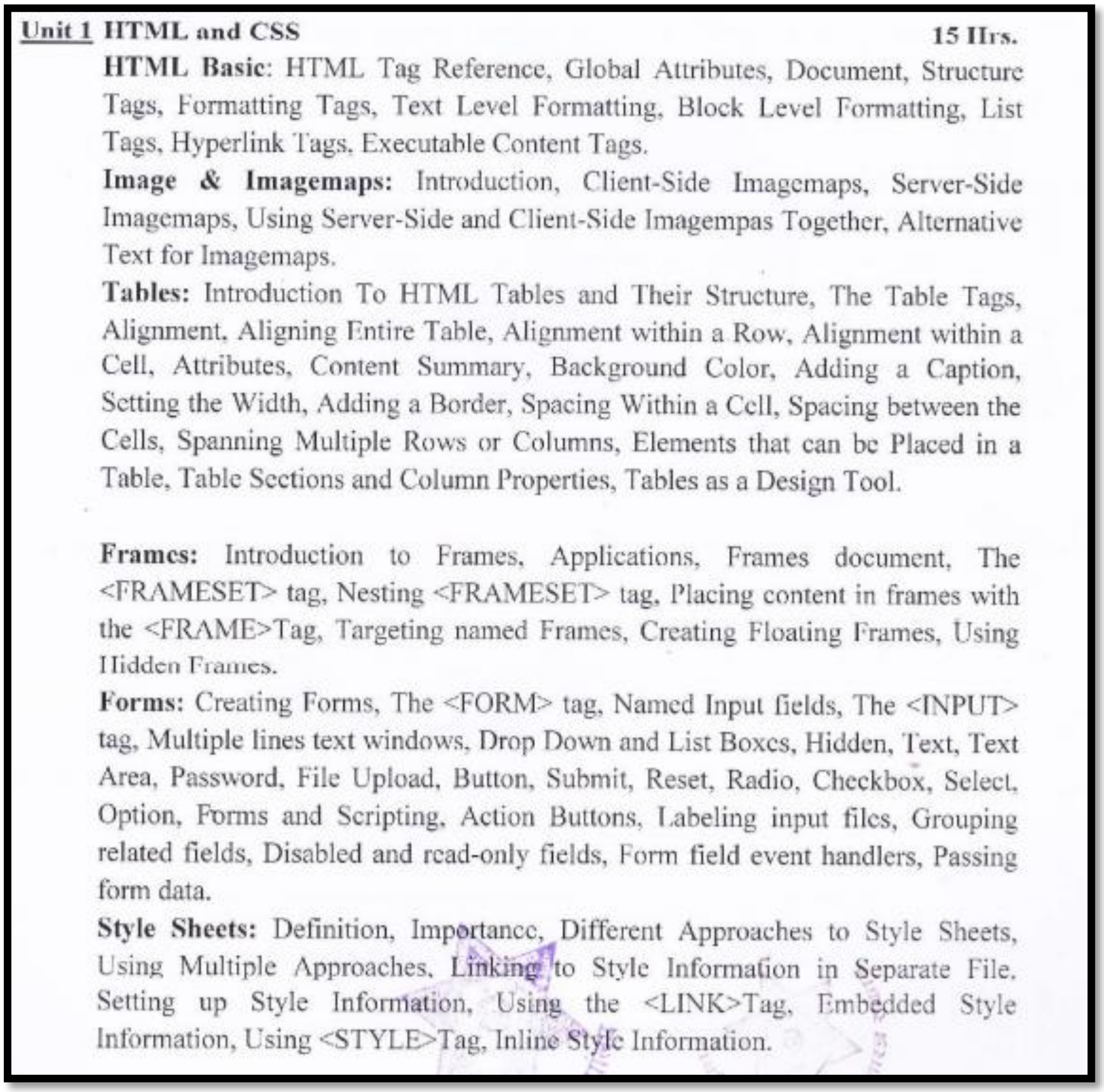
Unit-1

HTML and CSS



Unit 1 **HTML and CSS** **15 Hrs.**

HTML Basic: HTML Tag Reference, Global Attributes, Document, Structure Tags, Formatting Tags, Text Level Formatting, Block Level Formatting, List Tags, Hyperlink Tags, Executable Content Tags.

Image & Imagemaps: Introduction, Client-Side Imagemaps, Server-Side Imagemaps, Using Server-Side and Client-Side Imagempas Together, Alternative Text for Imagemaps.

Tables: Introduction To HTML Tables and Their Structure, The Table Tags, Alignment, Aligning Entire Table, Alignment within a Row, Alignment within a Cell, Attributes, Content Summary, Background Color, Adding a Caption, Setting the Width, Adding a Border, Spacing Within a Cell, Spacing between the Cells, Spanning Multiple Rows or Columns, Elements that can be Placed in a Table, Table Sections and Column Properties, Tables as a Design Tool.

Frames: Introduction to Frames, Applications, Frames document, The <FRAMESET> tag, Nesting <FRAMESET> tag, Placing content in frames with the <FRAME>Tag, Targeting named Frames, Creating Floating Frames, Using Hidden Frames.

Forms: Creating Forms, The <FORM> tag, Named Input fields, The <INPUT> tag, Multiple lines text windows, Drop Down and List Boxes, Hidden, Text, Text Area, Password, File Upload, Button, Submit, Reset, Radio, Checkbox, Select, Option, Forms and Scripting, Action Buttons, Labeling input files, Grouping related fields, Disabled and read-only fields, Form field event handlers, Passing form data.

Style Sheets: Definition, Importance, Different Approaches to Style Sheets, Using Multiple Approaches, Linking to Style Information in Separate File, Setting up Style Information, Using the <LINK>Tag, Embedded Style Information, Using <STYLE>Tag, Inline Style Information.

HTML Basics:

HTML stands for Hypertext Markup Language, and it is the most widely used language to write Web Pages. Hypertext refers to the way in which Web pages (HTML documents) are linked together and mark up language means it is the language that contains tags to “mark-up” the text to tell the web browser about how to structure and display the marked content.

HTML (Hypertext Markup Language) is the standard markup language used for creating web pages and applications. It forms the foundation of the World Wide Web and is essential for designing and structuring the content of a webpage.

HTML uses a set of tags to define the structure and presentation of the content within a webpage. These tags are enclosed in angle brackets (<>) and are interpreted by web browsers to render the page correctly.

HTML can work in conjunction with CSS (Cascading Style Sheets) and JavaScript to create visually appealing front end of a web page.

Common HTML Features:

- HTML stands for Hypertext Markup Language.
- HTML is the standard markup language used for creating web pages and applications.
- It provides a structure and format for organizing content on the web.
- HTML uses tags to define different elements of a web page, such as headings, paragraphs, links, images, and forms.
- HTML documents are hierarchical, with a root element called the <html> element.
- The structure of an HTML document consists of a head section <head> and a body section <body>.
- HTML tags are enclosed in angle brackets, such as <tagname>.
- Tags can have attributes, which provide additional information about the elements. Attributes are specified within the opening tag, like <tagname attribute="value">.
- HTML supports various types of content, including text, images, audio, video, and interactive elements.
- HTML is not a programming language but rather a markup language used to structure content.
- HTML5 is the latest version of HTML and introduced new features such as semantic elements, multimedia support, and improved form controls.
- HTML documents are interpreted and rendered by web browsers to display the content to users.
- CSS (Cascading Style Sheets) is commonly used with HTML to apply visual styles and layout to web pages.
- HTML is supported by all major web browsers and is a fundamental technology for the World Wide Web.
- With HTML, you can create hyperlinks to link web pages together and navigate through different websites.

Important Features of HTML:

- It is a markup language (not a programming language).
- It is used to create the structure of a webpage.
- It can be combined with server-side scripting language such as PHP as well as client-side scripting language such as JavaScript.
- It includes both formatting tags as well as semantic tags.

- It allows to create hyperlinks to navigate between the web pages.
- It supports the inclusion of multimedia elements such as images, audio, and video.
- It allows to create forms and different types of input elements.
- It can be combined with CSS to make web page more attractive.
- It is supported by all types of browsers and platforms.

Basic Structure of a HTML Document (Web Page):

A basic structure of an HTML document consists of the following elements:

a. **<!DOCTYPE> declaration:** This is the first line of an HTML document that specifies the version of HTML being used. For example, <!DOCTYPE html> indicates the use of HTML5.
b. **<html> element:** This is the root element of an HTML document. It contains all the other elements of the document.
c. **<head> element**: This element provides information about the document, such as its title, character encoding, and links to external stylesheets or scripts. It does not contain visible content.
d. **<title> element:** This element is placed within the <head> element and defines the title of the document, which appears in the browser's title bar or tab.
e. **<body> element:** This element contains the visible content of the web page, such as text, images, links, and other HTML elements.

Here's an example of a basic HTML document structure:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <!-- Visible content goes here -->
  </body>
</html>
```

Remember that this is a simplified structure, and HTML documents can have additional elements and attributes depending on the specific requirements and complexity of the web page.

HTML Tags:

HTML tags are special keywords or elements used to structure and define the content of a HTML document. They provide instructions to web browsers on how to render and display the elements within the document. HTML tags are typically enclosed within angle brackets (< >).

An HTML document is created using different types of tags. HTML tags can be defined and divided based on a different basis. Following are the different types of tags in HTML:

a. **Unpaired and paired tags:**
   An HTML tag is called an unpaired tag when the tag only has an opening tag and does not have a closing tag or a companion tag. The Unpaired HTML tag does not require a closing tag; an opening tag is sufficient in this type. Unpaired tags are sometimes also named as Standalone Tags or Singular Tags since they do not require a companion tag.
   For example <hr> tag, <br> tag, <img> tag etc.

   The HTML tags that always appear in pair are called paired tags. They contain both starting as well as closing tags. Most of the tags in HTML are paired tags. They are used to mark up the content between start and closing tags. An HTML Paired tag starts with an opening tag: the tag name enclosed inside the angle brackets; for example, a paragraph opening tag is written as '<p>'. The content follows the opening tag, which ends with an ending tag: the tag name starting with a forward slash; for example, an ending paragraph tag is written as '</p>'. The first tag can be referred to as the 'Opening Tag', and the second tag can be called Closing Tag.
   Example of paired tags: <p> tag, <body> tag, <head> tag etc.

b. **Semantic and Formatting tags:**
   Semantic tags provide meaning and context to the structure and content of an HTML document. They help search engines, screen readers, and other tools understand the purpose and relationship of the content. Semantic tags improve the accessibility and maintainability of web pages by providing clear structure and meaning to the content. This type of tags help in Search Engine Optimization (SEO). For example, <head>, <body>, <nav>, <div>, <section>, <footer> etc.

   Formatting tags are used to modify the appearance or presentation of the content on a web page. We can use formatting tags to get the desired appearance but it is always recommended to use CSS for formatting rather than HTML itself. For example <b>, <i>, <u>, <sub>, <sup> etc.

c. **Empty and container tags:**
   Empty tags, also known as self-closing tags or void elements, do not require a closing tag because they do not contain any content. They are used to insert specific elements or attributes into the HTML structure. For example: <br>, <img>, <input>, <hr>, <meta> etc.
   Empty tags are written as standalone tags, usually without a closing slash in HTML5, like <br>, <img>, <input>. However, in XHTML or XML, empty tags must be closed with a slash, such as <br/>, <img/>, <input/>.

   Container tags, also known as paired tags or start/end tags, come in pairs and have both an opening tag and a closing tag. They are used to enclose content within them. The content between the opening and closing tags is affected by the tag's purpose or behavior. For example: <p>, <div>, <h1>, <span>, <table>, <form>, <ul>, <ol> etc.

HTML Attributes:

HTML attributes are additional properties that can be added to HTML elements to provide extra information, define behaviors, or modify the appearance of the elements. Attributes are added within the opening tag of an element and consist of a name-value pair, separated by an equal sign (=).

The general syntax for an HTML attribute is as follows:

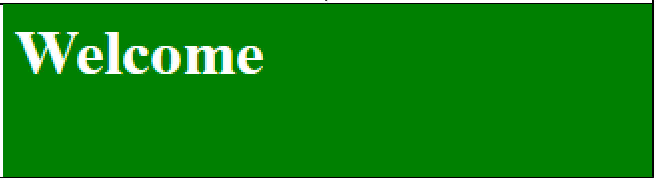
```
<element attribute_name="attribute_value">...</element>
```

Where;

- <element>: The HTML element to which the attribute is being added.
- attribute_name: The name of the attribute, which defines its purpose or behavior.
- "attribute_value": The value associated with the attribute, providing specific information or instructions.

For example, consider the following HTML code:

Source Code	Output

`<body bgcolor="green" text="white"`
`    <h1>Welcome</h1>`
`</body>`
	 Welcome

HTML attributes are used to enhance the functionality and appearance of HTML elements, and their usage varies depending on the element and the desired effect. Different elements support different attributes, and each attribute may have specific values or requirements associated with it.

Global Attributes: Global attributes in HTML are attributes that can be applied to most HTML elements. These attributes provide common functionality and behavior across different elements. Here are some commonly used global attributes:

a. class: Specifies one or more CSS class names to apply styles or group elements.
b. id: Specifies a unique identifier for an element.
c. style: Defines inline CSS styles for an element.
d. title: Provides additional information or a tooltip text for an element.
e. accesskey: Specifies a keyboard shortcut to focus or activate an element. Values may be ‘a’, ‘b’, ‘enter’ etc.
f. contenteditable: Indicates whether the content of an element is editable by the user. Values may be true or false.
g. draggable: Specifies whether an element can be dragged and dropped by the user. Values may be true or false.
h. hidden: Hides an element from being displayed. This attribute doesn’t need value.

i. lang: Specifies the language of the element's content. Values may be en, ne etc.

These attributes can be applied to various HTML elements, such as <div>, <p>, <span>, <a>, <input>, and more. They provide flexibility and control over the behavior, styling, and accessibility of elements within an HTML document. It's important to note that not all attributes are applicable to every element. The specific attributes available for an element depend on the element's purpose and intended use.

Document Structure Tags:

Document structure tags in HTML are used to organize and structure the content of a web page. These tags define the different sections and elements within an HTML document, providing a hierarchical structure and conveying the meaning and relationships of the content.

Following are some commonly used document structure tags in HTML:

- **<html>:** The root element that encloses the entire HTML document.
- **<head>:** Contains meta-information about the HTML document, such as the title, linking to external stylesheets or scripts, and other metadata.
- **<title>:** Specifies the title of the HTML document that appears in the browser's title bar or tab.
- **<body>:** Encloses the main content of the web page that will be displayed in the browser window.
- **<header>:** Represents the introductory or navigational section of a web page. Typically contains logos, site titles, and navigation menus.
- **<nav>:** Defines a section of navigation links.
- **<main>:** Represents the main content of the web page, excluding headers, footers, and sidebars.
- **<footer>:** Defines the footer section of a document, typically containing copyright information, contact details, or navigation links.
- **<section>:** Defines a standalone section within a document, such as chapters, tabbed content, or distinct areas.
  Example:

```html
<!DOCTYPE html>
<html>
<head>
  <title>Example Page</title>
</head>
<body>
  <header>
    <h1>Website Header</h1>
    <nav>
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">About</a></li>
        <li><a href="#">Services</a></li>
        <li><a href="#">Contact</a></li>
      </ul>
    </nav>
  </header>

  <main>
    <section>
      <h2>About Us</h2>
      <p>Welcome to our website! ... </p>
    </section>
    <section>
      <h2>Our Services</h2>
      <ul>
        <li>Service 1</li>
        <li>Service 2</li>
        <li>Service 3</li>
      </ul>
    </section>
  </main>

  <footer>
    <p>© 2023 Example Company. All rights reserved.</p>
  </footer>
</body>
</html>
```

- **<article>:** Represents a self-contained composition within a document, such as a blog post, news article, or forum post.
- **<aside>:** Represents content that is tangentially related to the main content, such as sidebars, pull quotes, or advertisements.
- **<div>:** A generic container element used to group and style other HTML elements.

```
<article>
  <h1>Main Article Heading</h1>
  <p>Main article content goes here.</p>

  <aside>
    <h3>Related Links</h3>
    <ul>
      <li><a href="#">Link 1</a></li>
      <li><a href="#">Link 2</a></li>
      <li><a href="#">Link 3</a></li>
    </ul>
  </aside>
</article>
```

Formatting Tags:

Formatting tags in HTML are used to apply specific formatting styles to text or elements within a web page. These tags allow us to control the appearance of the content. However, it's important to note that many formatting tags have been deprecated in favor of CSS (Cascading Style Sheets) for better separation of content and presentation. The formatting tags in HTML are of two types. They are text level formatting tags and block level formatting tags.

a. Text Level Formatting: Text-level formatting refers to the application of styles and formatting to individual portions of text within a document, such as words, phrases, or inline elements. Text-level formatting affects only the specific content it is applied to, rather than the overall structure of the document.

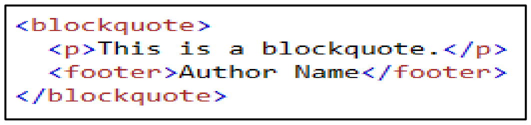
Some commonly used text level formatting tags are:

- <b>: Renders the enclosed text in bold.
- <i>: Renders the enclosed text in italics.
- <u>: Underlines the enclosed text.

- <strike>: Adds a strikethrough effect to the enclosed text.
- <strong>: Indicates strong importance, typically displayed as bold text.
- <em>: Emphasizes the enclosed text, typically displayed as italic text.
- <mark>: Highlights the enclosed text with a distinctive background color.
- <sub>: Formats the enclosed text as subscript.
- <sup>: Formats the enclosed text as superscript.
- <code>: Represents a snippet of computer code, typically displayed in a monospace font.
- <del>: Displays the enclosed text as deleted content.
- <ins>: Used to mark inserted text within a document. It is typically rendered as underlined text, indicating that the content has been added or inserted into the document.
- <big>: Used to display enclosed text in relatively bigger size. (depreciated in HTML 5).
- <small>: Used to display enclosed text in relatively smaller size. (depreciated in HTML 5)

b. Block Level Formatting: Block-level formatting involves the application of styles and formatting to larger sections or blocks of content within a document. Block-level elements typically start on a new line and take up the full width available. They contribute to the overall structure and layout of the document. Block-level formatting refers to the formatting applied to larger sections of content, such as paragraphs or headings. Here are some commonly used block-level formatting tags in HTML:

- <p>: Defines a paragraph of text.
- <h1> to <h6>: Headings of different levels, with <h1> being the highest level and <h6> the lowest.
- <div>: A generic container for grouping and applying styles to a block of content.
- <ul>: Defines an unordered (bulleted) list.
- <ol>: Defines an ordered (numbered) list.
- <li>: Represents a list item within an ordered or unordered list.
- <blockquote>: Indicates a block quotation.
- <pre>: Preformatted text, maintaining whitespace and line breaks as specified in the HTML code.
- <hr>: Represents a horizontal rule or divider.
  Example:



```
<blockquote>
  <p>This is a blockquote.</p>
  <footer>Author Name</footer>
</blockquote>
```

List Tags:

Lists are used to group related items or provide structured information. In HTML, there are several tags used to create lists. There are three different types of list tags in HTML:

a. **Unordered List:** In HTML, an unordered list is created using the <ul> (unordered list) tag. The <ul> tag is used to define a list of items without any particular order or sequence. Each list item is defined using the <li> (list item) tag. Here's an example of how to create an unordered list:

```
<ul>
 <li>C++</li>
 <li>Java</li>
 <li>Python</li>
 <li>C</li>
</ul>
```

By default, unordered lists use bullet points to represent each item. We can change the bullets type by using type attribute to <ul> tag. That possible values my be “disc/circle/square”.

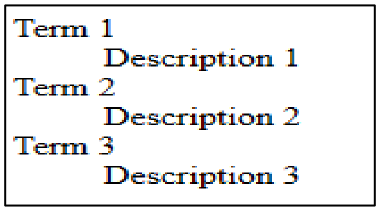
b. Ordered List: In HTML, an ordered list is created using the <ol> (ordered list) tag. The <ol> tag is used to define a list of items in a specific order, typically represented with numbers or letters. Each list item is defined using the <li> (list item) tag. Here's an example of how to create an ordered list:

```
<ol>
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
</ol>
```

By default, items are displayed in numeric order. We can change the order type sing type attribute of the <ol> tag with possible values “1/a/A/i/I”.

c. **Description/Definition List:** In HTML, a definition list is created using the <dl> (description list or definition list) tag. The <dl> tag is used to define a list of terms (definitions) and their associated descriptions. Each term is defined using the <dt> (definition term) tag, and each description is defined using the <dd> (definition description) tag. Here's an example of how to create a definition list:

```
<dl>
  <dt>Term 1</dt>
  <dd>Description 1</dd>
  <dt>Term 2</dt>
  <dd>Description 2</dd>
  <dt>Term 3</dt>
  <dd>Description 3</dd>
</dl>
```



```
Term 1
        Description 1
Term 2
        Description 2
Term 3
        Description 3
```

Hyperlink/Anchor Tags:

Hyperlinks are used to create clickable links that allow users to navigate between different web pages or sections within the same page. We can also create links to other website using hyperlink. Hyperlinks are created using the <a> (anchor) tag and the href attribute. Hyperlinks are fundamental for creating navigation menus, connecting different pages, linking to external resources, and providing a seamless browsing experience for users.

The <a> tag indicates where the hyperlink starts and the </a> tag indicates where it ends. Whatever text gets added inside these tags, will work as a hyperlink. The address of the target page is included as the value of href attribute.

Example: Following example shows how we can create hyperlink “Click me”, which is linked to an external website “abcd.com”.

```html
<a href="https://www.abcd.com">Click me!</a>
```

The output of the above code is as follows:

Click me!

When a user clicks on a hyperlink, the browser follows the URL specified in the href attribute and loads the corresponding web page.

Let's break down the different parts of a hyperlink:

- **<a> tag:** It is the anchor tag, which defines a hyperlink. It is an **inline element** that can contain other text or elements such as <p>, <img>, <h1> etc.
- **href**: The href attribute specifies the target URL or destination of the hyperlink. It can be an absolute URL (e.g., "https://www.abcd.com") or a relative URL (e.g., "page.html"). The absolute URL uses full address of a webpage and the relative URL uses webpage name only. Relative URL is used if the target page is within the same website.
- **Link Text:** The text between the opening and closing <a> tags represents the visible link text that users see and can click on. In the example above, "Click me!" is the link text. Instead of link text, we can add <img>, <p>, <h1> or other html elements as well.

Additionally, hyperlinks can also be used to create internal links within the same HTML document. This is achieved by specifying an anchor name using the id attribute on an element and then referencing it in the href attribute of the <a> element.

By default, links will appear as follows in all browsers:

- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red

Following example shown the use of anchor tag to create internal links.

```html
<!DOCTYPE html>
<html>
<head>
  <title>Internal Hyperlink Example</title>
</head>
<body>
  <h1 id="top">Internal Hyperlink Example</h1>

  <p>Read the article <a href="#article">Here</a>.

  <h2>Section 1</h2>
  <p>This is the content of Section 1.</p>

  <h2>Section 2</h2>
  <p>This is the content of Section 2.</p>
  <article id="article">
    this is the place where you can place your article.
  </article>
  <a href="#top">Go to top</a>
</body>
</html>
```

In the above example, if we click on “Here”, the page is navigated to the article and again if we click on “Go to top”, the page is navigated back to top.

Executable Content Tags in HTML:

Executable content tags are used to include executable code or scripts within an HTML document. These tags allow to embed or reference code written in programming languages like JavaScript, which can be executed by the web browser.

These executable content tags allow you to enhance the functionality and interactivity of your web pages by incorporating dynamic code or styles. They provide the means to execute scripts, define styles, and create interactive elements within an HTML document.

The primary executable content tags in HTML are:

a. **<script> tag:** The <script> tag is used to embed or reference JavaScript code within an HTML document. It can be placed in the <head> section or the <body> section of the HTML document. JavaScript code within the <script> tags is executed by the browser when it encounters the tag. Here's an example of using the <script> tag:

```html
<script>
  // JavaScript code goes here
  alert("Hello, world!");
</script>
```

b. **<style> tag:** The <style> tag is used to embed or reference CSS (Cascading Style Sheets) code within an HTML document. This tag can be placed in the <head> section of a web page.CSS is used to define the visual appearance and layout of HTML elements. The CSS code within the <style> tags is interpreted by the browser to style the associated HTML content. Here's an example of using the <style> tag:

```
<!DOCTYPE html>
<html>
    <head>
        <style>
          /* CSS code goes here */
              body {
                background-color: lightblue;
              }
        </style>
    </head>
    <body>
        The content of the body element is displayed in your browser.
    </body>
</html>
```

c. **<canvas> tag:** The <canvas> tag is used to create a drawing area in an HTML document and execute JavaScript code to draw graphics, animations, or interactive elements on the canvas. It provides a powerful API for dynamically generating and manipulating graphics using JavaScript. Here's an example of using the <canvas> tag to draw a straight line:

```
<!DOCTYPE html>
<html>
<head>
  <title>Drawing a Line on Canvas</title>
</head>
<body>
  <canvas id="myCanvas" width="400" height="200"></canvas>

  <script>
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
    // Set the starting point of the line
    var startX = 50;
    var startY = 100;
    // Set the ending point of the line
    var endX = 350;
    var endY = 150;
    // Set the line color and width
    context.strokeStyle = "green";
    context.lineWidth = 5;
    // Start drawing
    context.beginPath();
    context.moveTo(startX, startY);
    context.lineTo(endX, endY);
    // Draw the line
    context.stroke();
  </script>
</body>
</html>
```

The output of above code is:

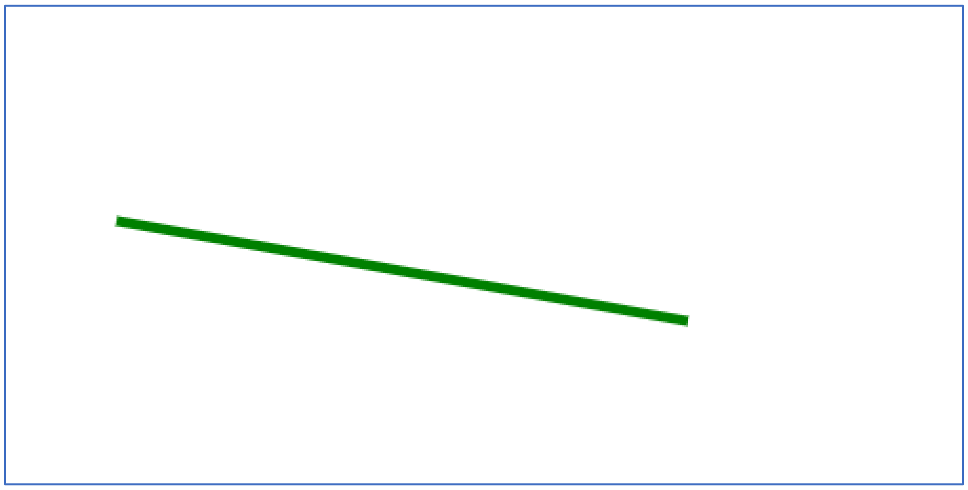


Image and Imagemaps:

Using <img> tag:

The <img> tag is used to embed an image in an HTML 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 can have the following attributes:

a. src - Specifies the path to the image.
b. alt - Specifies an alternate text for the image, if the image for some reason cannot be displayed.
c. Width – Specifies the width of an image. Its values is usually given in px or in percentage.
d. Height – To specify the height of an image. Its values is usually given in pixel or percentage.
e. Title – Specifies the title for the image. It will be displayed when the user hovers the mouse over the image.
f. Border- It specifies the border around the image. The border is always in black color. It is not recommended to use border attribute to give image border.

Except above mentioned attributes, <img> tag also supports other global attributes such as id, class etc.

Here are some examples of how to use the <img> tag:

```
<img src="image.jpg" alt="This is an image of a cat." width="100" height="100">
```

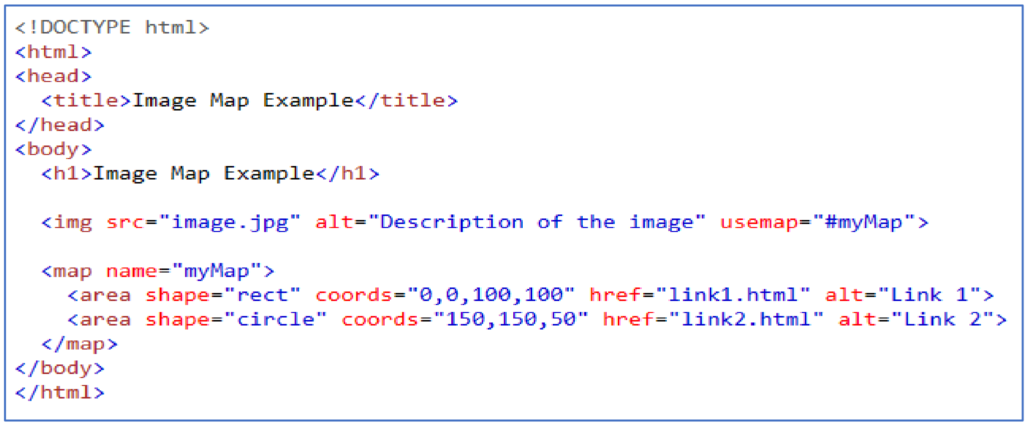
```
<img src="image.jpg" alt="This is an image of a cat." title="This is a picture of a cat">
```

While giving height and width to an image, its aspect ratio may get disturbed. In order to keep the aspect ratio intact, it is always recommended to give either height or width only but not both.

Using Imagemap:

An image map in HTML is a technique that allows specific areas of an image to be clickable and associated with different hyperlinks or actions. When a user clicks on a specific area within the image, the associated hyperlink or action is triggered.

To create an image map in HTML, we need to use the <map> and <area> tags in conjunction with an <img> tag. Here's an example:



```html
<!DOCTYPE html>
<html>
<head>
  <title>Image Map Example</title>
</head>
<body>
  <h1>Image Map Example</h1>

  <img src="image.jpg" alt="Description of the image" usemap="#myMap">

  <map name="myMap">
    <area shape="rect" coords="0,0,100,100" href="link1.html" alt="Link 1">
    <area shape="circle" coords="150,150,50" href="link2.html" alt="Link 2">
  </map>
</body>
</html>
```

Note that: The name attribute must have the same value as the <img>'s usemap attribute .

In the above example:

- The <img> tag is used to display an image on the webpage.
- The src attribute specifies the path or URL of the image file.
- The usemap attribute is added to the <img> tag and points to the corresponding <map> tag using the name attribute.
- The <map> tag is used to define the image map.
- Inside the <map> tag, the <area> tags define the clickable areas within the image.
- The shape attribute specifies the shape of the clickable area, which can be "rect" (rectangle), "circle", or "poly" (polygon).
- The coords attribute specifies the coordinates of the shape. The values depend on the shape type.
- The href attribute specifies the URL or destination of the link when the area is clicked.

We must define the shape of the clickable area, and we can choose one of these values:

a. rect - defines a rectangular region. The coordinates values for shape= ‘rect’ can be given as:
`cords="x-coordinate of top-left point, y coordinate of top-left point, x-coordinate of bottom-right point, y-coordinate of bottom-right point”`

b. circle - defines a circular region. The coordinate values for shape= “circle” can be given as:
`cords="x-coordinate of center, y-coordinate of center, radius”.`

c. poly - defines a polygonal region. The shape="poly" contains several coordinate points, which creates a shape formed with straight lines (a polygon). This can be used to create any shape. The coordinates values for shape= “poly” can be given as:
`cords="x1,y1,x2,y2,x3,y3,…”` where x=(x1,y1), (x2,y2), (x3,y3) … are the three points.

d. default - defines the entire region. No coordinates must be given.

Client-side image maps and server-side image maps are two different approaches for creating interactive image maps in HTML. Client-side image map is more common in practice. The image map discussed above is a client-side image map.

Client-Side Imagemaps:

Client-side image maps are created using HTML and JavaScript, and the interactivity is handled on the client-side (in the user's web browser). In this approach, the image and the map coordinates are defined within the HTML code. When a user clicks on a defined area, the browser handles the action associated with that area using JavaScript.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Client-Side Image Map Example</title>
</head>
<body>
  <h1>Client-Side Image Map Example</h1>

  <img src="image.jpg" alt="Description of the image" usemap="#myMap">

  <map name="myMap">
    <area shape="rect" coords="0,0,100,100" href="link1.html" alt="Link 1">
    <area shape="circle" coords="150,150,50" href="link2.html" alt="Link 2">
    <area shape="poly" coords="200,200,250,250,200,300" href="link3.html" alt="Link 3">
  </map>

  <script>
    // JavaScript code for handling client-side image map events
    // You can add custom JavaScript code here to handle actions when an area is clicked
  </script>
</body>
</html>
```

In the above example, the client-side image map is created using the <map> and <area> tags. However, the JavaScript section is left empty, and you can add your custom JavaScript code to handle the actions when an area is clicked.

Server-Side Imagemaps:

Server-side image maps are created by defining the image and map coordinates on the server-side (usually with the help of server-side scripting languages like PHP, ASP, or CGI). When a user clicks on a specific area, the browser sends a request to the server, which processes the request and responds accordingly.

Here's a high-level overview of the server-side image map process:

- The HTML page contains an <img> tag that specifies the image source.
- The server-side script defines the map coordinates and associates them with specific actions or URLs.

- When a user clicks on an area within the image, the browser sends a request to the server with the associated information.
- The server-side script receives the request, processes it, and performs the necessary action (such as redirecting to a specific page or performing a server-side operation).

Server-side image maps require additional server-side scripting and configuration to handle the requests and responses.

Example:

index.html page:

```html
<!DOCTYPE html>
<html>
<head>
  <title>Server-Side Image Map Example</title>
</head>
<body>
  <h1>Server-Side Image Map Example</h1>

  <img src="image.jpg" alt="Description of the image" usemap="#myMap">

  <map name="myMap">
    <area shape="rect" coords="0,0,100,100" href="server-side-script.php?action=link1" alt="Link 1">
    <area shape="circle" coords="150,150,50" href="server-side-script.php?action=link2" alt="Link 2">
  </map>
</body>
</html>
```

server-side-script.php page:

```php
<?php
  // Get the action parameter from the URL query string
  $action = $_GET['action'];

  // Perform server-side actions based on the received action parameter
  if ($action === 'link1') {
    // Perform action for Link 1
    // Redirect to another page, perform database operations, or any other server-side logic
    header("Location: link1.html");
    exit();
  } elseif ($action === 'link2') {
    // Perform action for Link 2
    // Redirect to another page, perform database operations, or any other server-side logic
    header("Location: link2.html");
    exit();
  } else {
    // Invalid or unknown action
    // Handle the error accordingly
    echo "Invalid action";
    exit();
  }
?>
```

In this example, the HTML page (index.html) contains an image with a server-side image map defined using the <map> and <area> tags. The href attributes of the <area> tags point to the server-side script (server-side-script.php) with a query parameter action indicating the specific action to be performed.

The choice between client-side and server-side image maps depends on the specific requirements. Client-side image maps are simpler to implement and handle the interactivity on the client-side, while server-side image maps provide more flexibility and can perform complex server-side operations based on the user's interaction.

Using Client-Side and Server-side Imagemap Together:

Using client-side and server-side image maps together is possible and can provide additional flexibility in handling interactivity and actions. Using client-side and server-side image maps together can provide a combination of client-side interactivity and server-side processing. By combining client-side and server-side image maps, you can handle immediate interactivity on the client-side using JavaScript while also enabling server-side processing for more complex operations or interactions that require server resources.

In a combined imagemap, some area are connected with server side script and some are connected with client side script.

Example:

index.html page:

```html
<!DOCTYPE html>
<html>
<head>
  <title>Server-Side Image Map Example</title>
</head>
<body>
  <h1>Server-Side Image Map Example</h1>

  <img src="image.jpg" alt="Description of the image" usemap="#myMap">

  <map name="myMap">
    <area shape="rect" coords="0,0,100,100" href="server-side-script.php?action=link1" alt="Link 1">
    <area shape="circle" coords="150,150,50" onclick="myFunction">
  </map>

  <script>
        function myFunction()
        {
            alert("You clicked on the curcle image map");
        }
  </script>
</body>
</html>
```

server-side-script.php page:

```
<?php
  // Get the action parameter from the URL query string
  $action = $_GET['action'];

  // Perform server-side actions based on the received action parameter
  if ($action === 'link1') {
    // Perform action for Link 1
    // Redirect to another page, perform database operations, or any other
server-side logic
    header("Location: link1.html");
    exit();
  }else {
    // Invalid or unknown action
    // Handle the error accordingly
    echo "Invalid action";
    exit();
  }
?>
```

Alternative text for Imagemaps:

Alternative text (alt text) for an image map is essential for providing a textual description of the image and its interactive regions to users who may have visual impairments, use assistive technologies, or have images disabled in their browsers. Alt text should convey the purpose and content of the image map.

Following are the benefits of using alt text in imagemap:

- It can describe the overall purpose or function of the image map in a concise manner.
- It helps to clearly describe each interactive region and its associated action or destination.
- It helps to include relevant keywords that accurately represent the content or topic of the image map.
- It helps in SEO.

Example:

Here's an example of alt text for an image map with three interactive regions:

```
<img src="your-image.jpg" usemap="#your-map" alt="Description of the image">

<map name="your-map">
  <area shape="rect" coords="x1,y1,x2,y2" href="your-link.html" alt="Alt text for the first area">
  <area shape="circle" coords="x,y,radius" href="your-link.html" alt="Alt text for the second area">
  <area shape="poly" coords="x1,y1,x2,y2,x3,y3,x4,y4" href="your-link.html" alt="Alt text for the third area">
  <!-- Add more area elements if needed -->
</map>
```

In the example above, the alt attribute is added to both the img tag and each area element within the map tag. The alt attribute provides a description of the image and the areas within the image map. This description is used by screen readers and can be displayed if the image fails to load.

Tables in HTML:

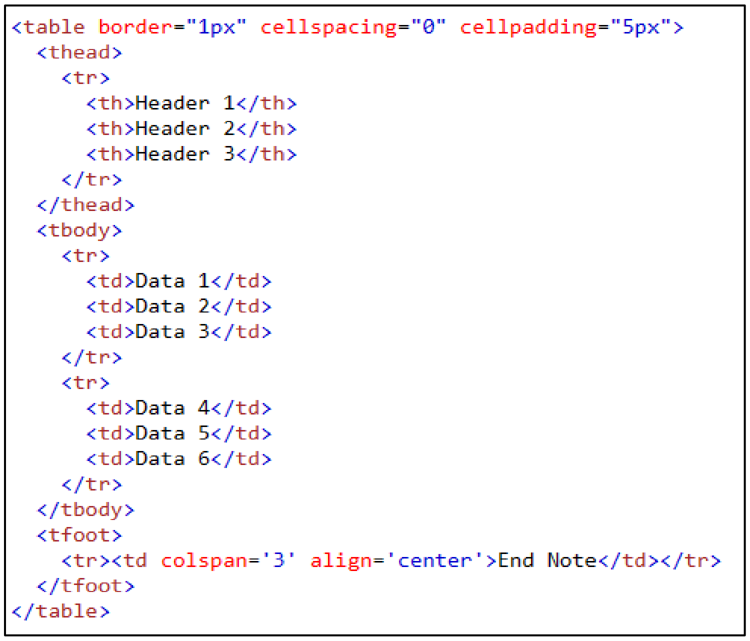
The HTML tables allow web authors to arrange data like text, images, links, other tables, etc. into rows and columns of cells. We can create a table to display data in tabular form, using <table> element, with the help of <tr> , <td>, and <th> elements. In Each table, table row is defined by <tr> tag, table header is defined by <th>, and table data is defined by <td> tags.

HTML tables can also be used to manage the layout of the page e.g. header section, navigation bar, body content, footer section etc. But it is recommended to use div tag over table to manage the layout of the page.

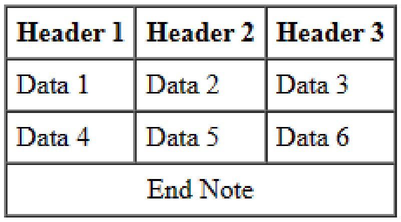
Here's a breakdown of the key components used with the <table> tag:

- **<table>:** The main container element for the table.
- **<caption>:** It is placed just after <table> tag. It contains the table caption. That displays just above the table.
- **<thead>:** This element is used to define the header section of the table. It is typically placed immediately after the opening <table> tag and contains one or more <tr> elements.
- **<tr>:** Stands for table row. It defines a single row within the table. Each row consists of one or more <td> or <th> elements.
- **<th>:** Used to define a header cell within the table. It is typically placed within a <tr> element in the <thead> section. By default, header cells are rendered in bold and centered.
- **<td>:** Represents a standard data cell within the table. It is used to define cells in the table body (<tbody>) and table footer (<tfoot>).
- **<tbody>:** This element is used to group the main content of the table, excluding the header and footer sections. It can contain one or more <tr> elements.
- **<tfoot>:** Represents the footer section of the table. It is placed after the table body and contains one or more <tr> elements.

Example:



```
<table border="1px" cellspacing="0" cellpadding="5px">
  <thead>
    <tr>
      <th>Header 1</th>
      <th>Header 2</th>
      <th>Header 3</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Data 1</td>
      <td>Data 2</td>
      <td>Data 3</td>
    </tr>
    <tr>
      <td>Data 4</td>
      <td>Data 5</td>
      <td>Data 6</td>
    </tr>
  </tbody>
  <tfoot>
    <tr><td colspan='3' align='center'>End Note</td></tr>
  </tfoot>
</table>
```



Header 1	Header 2	Header 3
Data 1	Data 2	Data 3
Data 4	Data 5	Data 6
End Note		

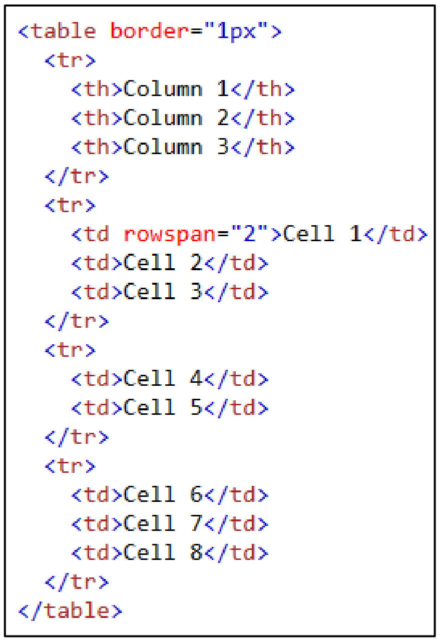
Attributes used with <table> tag:

- **border:** Used to specify the table border. Values are given in pixel.
- **summary:** The summary attribute is used in HTML to provide a summary or description of the content and purpose of a table. It helps users, especially those using assistive technologies, to understand the table's meaning and context.
  Example: `<table summary="short description about table content">`
- **cellspacing:** This attribute describes the space between table cells. Value is given in pixel.
- **cellpadding:** This attribute describes the space between cell border and cell content. Value is given in pixel.
- **width:** This attribute is used to specify the table width in pixel or percentage.
- **height:** This attribute is used to specify the table height in pixel or percentage.
- **align:** It is used to set the alignment of a table. The possible values may be “right”, “left” or “center”. Example: `<table align="right" border="1">`
- **bgcolor:** It is used to set the background color of a table. The possible values may be color name or color code. Example: `<table border="1" bgcolor="red/#ff0000">`
- **rospan:** It is used to merge the cells vertically.
- **colspan:** It is used to merge the cells horizontally.

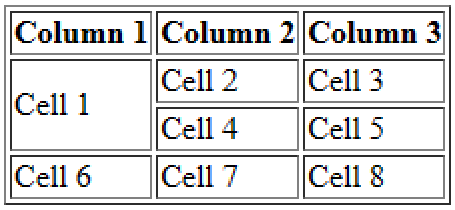
Using “rowspan”:

The rowspan attribute in HTML tables is used to define the number of rows that a cell should span vertically. It is used within the <td> or <th> tags to specify how many rows a cell should occupy.

Here's an example to illustrate the usage of rowspan including output in the alongside.



```html
<table border="1px">
  <tr>
    <th>Column 1</th>
    <th>Column 2</th>
    <th>Column 3</th>
  </tr>
  <tr>
    <td rowspan="2">Cell 1</td>
    <td>Cell 2</td>
    <td>Cell 3</td>
  </tr>
  <tr>
    <td>Cell 4</td>
    <td>Cell 5</td>
  </tr>
  <tr>
    <td>Cell 6</td>
    <td>Cell 7</td>
    <td>Cell 8</td>
  </tr>
</table>
```



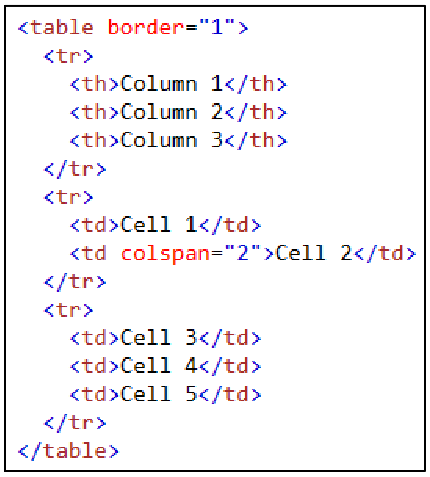
Column 1	Column 2	Column 3
Cell 1	Cell 2	Cell 3
	Cell 4	Cell 5
Cell 6	Cell 7	Cell 8

In this example, the first cell in the first row has a rowspan attribute set to "2", which means it spans two rows. As a result, the cell occupies the space of the second row as well.

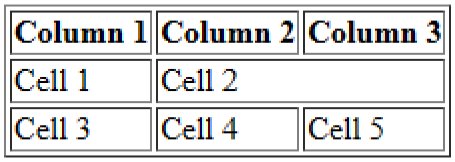
Using “colspan”:

The colspan attribute in HTML tables is used to define the number of columns that a cell should span horizontally. It is used within the <td> or <th> tags to specify how many columns a cell should occupy.

Here's an example to illustrate the usage of colspan along with output:



```html
<table border="1">
  <tr>
    <th>Column 1</th>
    <th>Column 2</th>
    <th>Column 3</th>
  </tr>
  <tr>
    <td>Cell 1</td>
    <td colspan="2">Cell 2</td>
  </tr>
  <tr>
    <td>Cell 3</td>
    <td>Cell 4</td>
    <td>Cell 5</td>
  </tr>
</table>
```



Column 1	Column 2	Column 3
Cell 1	Cell 2	
Cell 3	Cell 4	Cell 5

In this example, the second cell in the second row has a colspan attribute set to "2", which means it spans two columns. As a result, the cell occupies the space of the second and third columns.

Table sections and column properties:

In HTML, tables can be divided into different sections and columns using specific properties. These properties help organize and style the table for better structure and presentation. Let's explore the table section and column properties:

Table Sections:

- <thead>: Represents the section of the table that contains header information. It typically includes one or more <tr> elements with <th> cells for column headers.
- <tbody>: Represents the main body section of the table. It contains the data rows represented by <tr> elements, each containing one or more <td> cells with actual data.
- <tfoot>: Represents the footer section of the table. It is used to display summary information or aggregated data related to the table content. Similar to <thead>, it can contain one or more <tr> elements with <td> or <th> cells.

Table Columns:

- colspan: Specifies the number of columns a cell should span horizontally. It is used within <td> or <th> elements. For example, colspan="2" would make the cell span across two columns.
- rowspan: Specifies the number of rows a cell should span vertically. It is also used within <td> or <th> elements. For example, rowspan="3" would make the cell span across three rows.
- These properties help structure the table by grouping related content in sections and merging cells across columns and rows when needed.

Table as a design tool:

HTML tables can be used as a basic design tool to structure and organize content on a web page. While tables were primarily designed for tabular data, they can also be utilized for layout purposes. If you still choose to use HTML tables for webpage design layout, here are some steps to consider:

a. Structure the Table: Start by creating the basic structure of the table using the <table>, <tr>, and <td> elements. Think of the table as a grid system where each <td> represents a cell in the layout.

b. Define Rows and Columns: Determine the number of rows and columns you need for your layout. Use the appropriate number of <tr> elements for rows and <td> elements for columns within each row.

c. Specify Content: Place your content within the cells (<td> elements) of the table. This can include text, images, forms, or any other HTML elements.

d. Apply Styling: Use CSS to style the table and its cells to achieve your desired design. You can target the table, rows, and cells using CSS selectors and apply properties like background-color, border, padding, and width to control the appearance of the layout.

e. Responsiveness: Tables have limited responsiveness capabilities, so consider using CSS media queries to make your table layout more adaptive for different screen sizes. You may need to adjust column widths, hide or show certain columns, or modify the table structure to make it more responsive.

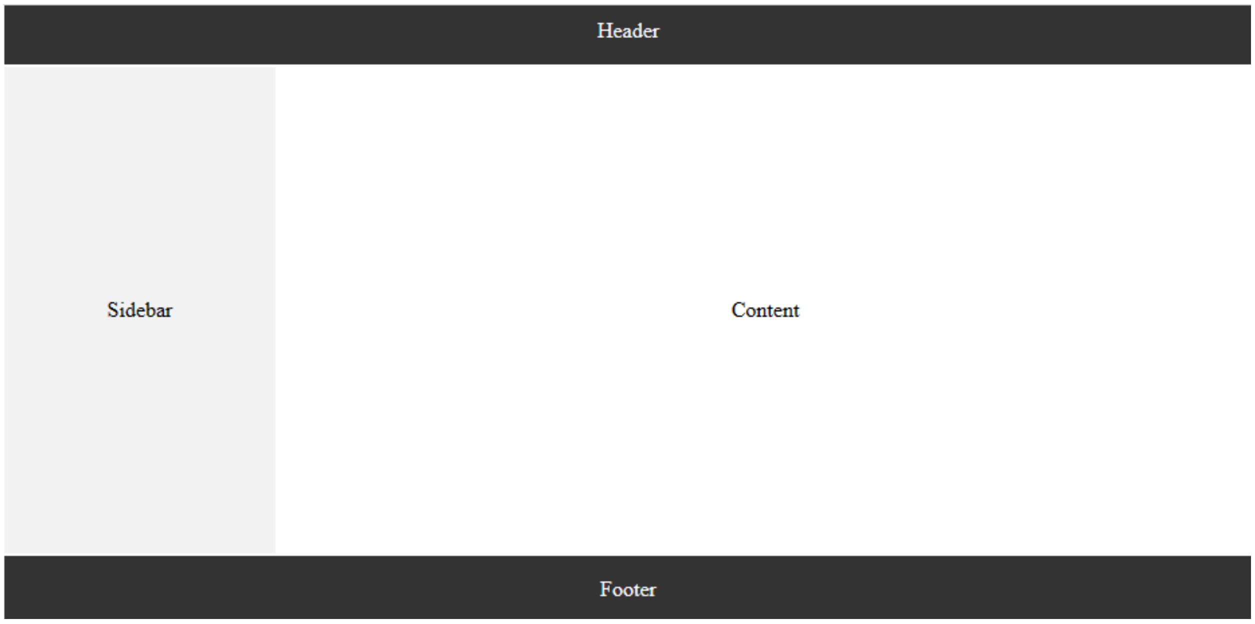
using HTML tables for layout purposes can have drawbacks, such as less flexibility, difficulty in maintaining responsive designs, and potential accessibility issues. It's generally recommended to use CSS layout techniques, such as Flexbox or CSS Grid, as they provide more control, responsiveness, and better separation of content and presentation.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    table {
      width: 100%;
      height: 100vh;
    }

    td {

      text-align: center;
    }

    .header {
      background-color: #333;
      color: #fff;
      height: 50px;
    }

    .sidebar {
      background-color: #f2f2f2;
      width: 200px;
    }

    .content {
      background-color: #fff;
    }

    .footer {
      background-color: #333;
      color: #fff;
      height: 50px;
    }
  </style>
</head>
<body>
```

```
  <table>
    <tr class="header">
      <td colspan="2">Header</td>
    </tr>
    <tr>
      <td class="sidebar">Sidebar</td>
      <td class="content">Content</td>
    </tr>
    <tr class="footer">
      <td colspan="2">Footer</td>
    </tr>
  </table>

</body>
</html>
```

Output:



Frames in HTML:

Frames in HTML are used to divide a web page into multiple sections called frames. Each frame can contain a separate HTML document or webpage and can be independently scrolled or updated. Each frame acted as a separate window within the web browser. frame is a part of a web page or browser window which displays content independent of its container, with the ability to load content independently.

HTML frame can be added in a webpage using <frameset> and <frame> tags.

a. **<frameset> tag:** This tag is used to provide the frame structure in the form of rows and columns. It divides the webpage in different sections where actual frames can be placed. It defines the area in the page where the content will be displayed from another page. It can take two attributes “row” and “cols”. The value to these attributes are the size of frames to be included given in pixel or percentage. HTML frames allow authors to present documents in multiple views, which may be independent windows or sub-window.
b. **<frame> tag:** This tag is used to provide the actual frame content. It includes the resource address as the value of src attribute.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Frames Example</title>
</head>
<frameset rows="20%, 80%">
  <frame src="header.html">
  <frame src="content.html">
</frameset>
</html>
```

Output:

The output of the above source code consists of a webpage divided into two sections horizontally. The upper section is 20% and lower section height is 80%. In the upper section, header.html is displayed and in the lower section content.html will be displayed.

Uses of frames:

- **Splitting a webpage:** Frames allowed developers to divide a webpage into multiple sections, each with its own HTML document.

- **Website navigation:** Frames allow for creating a persistent navigation menu or sidebar that remained visible across multiple pages. This way, users could navigate through different sections of a website without the need for the navigation to reload with each page.

- **Multilingual websites:** Frames were sometimes used to display different language versions of a website side by side. Each frame would contain the same content in a different language, allowing users to switch between languages easily.

- **Advertisements and banners:** Frames provided a way to display advertisements or banners that remained static while the content in the other frame(s) changed. This allowed advertisers to maintain visibility and prevent the ad from being disrupted by page navigation.

- **Complex layouts:** Frames enabled the creation of complex webpage layouts with multiple independent sections. Each frame could load a different HTML document, providing the ability to update specific parts of a webpage without affecting others.

It's important to note that while frames offered these functionalities, they also had several drawbacks, such as accessibility issues, bookmarking difficulties, and SEO challenges.

Nesting <frameset> tag:

We can include <frameset> tag within another <frameset> tag. This is possible when we use <frameset> to include an html page that contains <frameset> tag again.

Example: Consider the four html documents as follows:

"home.html" page

```
<frameset rows="20%,80%">
    <frame src="top.html">
    <frame src="main.html">
</frameset>
```

"main.html" page

```
<frameset cols="20%,80%">
    <frame src="left.html">
    <frame src="right.html">
</frameset>
```

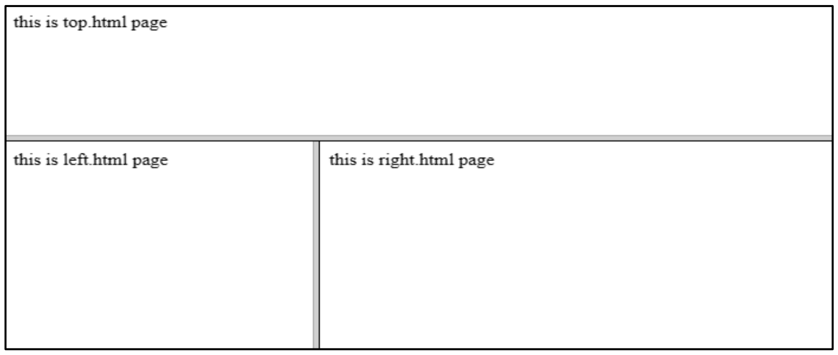
"left.html" page

```
This is left.html page
```

"right.html" page

```
This is right.html page
```

Output:



```
this is top.html page
this is left.html page
this is right.html page
```

Targeting Named Frame:

In HTML, when using the <frameset> element to divide a web page into multiple frames, we can target a specific named frame using the target attribute on links, forms, or other elements.

Here's an example of how you can target a named frame within a <frameset>.

```
<!DOCTYPE html>
<html>
<head>
  <title>Targeting Named Frames in Frameset</title>
</head>
<frameset cols="50%, 50%">
  <frame src="frame1.html" name="frame1">
  <frame src="frame2.html" name="frame2">
</frameset>
</html>
```

In the example above, we have a <frameset> element that divides the page into two frames, named "frame1" and "frame2". Each frame is defined using the <frame> element and has its own src attribute pointing to the respective HTML files.

To target a specific frame, we can use the target attribute on links or forms. For example, if we want to target "frame1" when a link is clicked, we can use the target attribute like this:

```
<a href="page.html" target="frame1">Click me to target frame1</a>
```

When the link is clicked, the content of "page.html" will load within the frame named "frame1".

Creating Floating Frames:

Inline Frame or an iframe allows us to open new pages inside a page. It is similar to frameset tag. Inline frames are also referred to as Floating frames. We use <iframe> tag to create an inline frame.

The basic syntax for creating iframe is as follows:

```
<iframe height=“height” width=“width” src="url_of_the_document"></iframe>
```

Where;

- The src attribute specifies the URL or path to the document you want to embed.
- The height and width attributes define the height and width of the frame.
- The frameborder attribute defines the thickness of the frame boarder. The value must be given in pixel.

Uses:

- The <iframe> is used to embed other webpage within the specified section of a webpage.
- It can be used to share the content of one webpage to another. For example a YouTube video can be shared to other webpages using iframe.

Using Hidden Frames:

Hidden frames, also known as invisible frames or zero-sized frames, are frames that are visually hidden on a web page. They are created using the <frame> or <iframe> tag and have a width and height of zero or are positioned off-screen.

Following are some of the uses of hidden frames:

- To submit data without page reload.
- To load and manipulate external content from other webpages.
- To communicate between different web pages.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Hidden Frame</title>
</head>
<body>
  <h1>Hidden Frame Example</h1>

  <iframe src="about:blank" style="display: none;"></iframe>

  <p>This is the rest of my web page.</p>
</body>
</html>
```

In the above example:

- The <iframe> tag is used to create a hidden frame.
- The src attribute is set to "about:blank" to create an empty frame.
- The style attribute with display: none; is added to hide the frame from being visible on the page.

HTML Forms

Forms in HTML are used to collect user input and send it to a server for processing. They allow users to enter data, make selections, and submit the form to trigger an action or send the data to a server-side script. A form contains controls such as text fields, password fields, checkboxes, radio buttons, submit button, menus etc. HTML forms are required if you want to collect some data from of the site visitor. For example: If a user want to purchase some items on internet, he/she must fill the form such as shipping address and credit/debit card details so that item can be sent to the given address.

We use the following tags to create a HTML form:

Tag	Description
<form>	The <form> tag is used to create a form. It encloses all the form elements and defines the form's behavior and destination.
<input>	It is Used for various types of input fields such as text, password, email, checkbox, radio buttons, etc.
<textarea>	It defines a multi-line input control.
<label>	It defines a label for an input element. It is used for better usability and accessibility.
<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.

<input> tag:

The <input> tag in HTML is used to create various types of input elements that allow users to enter data or make selections. It is a self-closing tag and does not have a closing tag.

The <input> tag has several attributes that control its behavior, appearance, and the type of input it represents. Here are some commonly used attributes for the <input> tag:

- **type:** Specifies the type of input element to be created. For example, type="text" creates a text input field, type="password" creates a password input field, and so on. The type attribute is required for the <input> tag.
  Following are different types of input types in html.

a. **Text Input (<input type="text">):** This is the most basic input type used to accept single-line text input from the user. It can be used for various purposes like capturing names, addresses, or any other short text data.

b. **Password Input (<input type="password">):** This input type is specifically designed to capture sensitive information like passwords. It masks the entered characters to protect the user's privacy.

c. **Email Input (<input type="email">):** This input type is used to capture email addresses. It provides built-in validation to ensure that the entered value follows the standard email format.

d. **Number Input (<input type="number">):** This input type is used to accept numeric input from the user. It can be used for capturing quantities, ages, or any other numeric data. It may include additional attributes like min, max, and step to define the range and increment/decrement values.

e. **Checkbox Input (<input type="checkbox">):** Checkboxes allow the user to select multiple options from a predefined set. Each checkbox represents a single value, and the selected values are sent to the server upon form submission.

f. **Radio Button Input (<input type="radio">):** Radio buttons are used when the user can select only one option from a predefined set of mutually exclusive options. Each radio button represents a single value, and only one option can be selected at a time.

g. **File Input (<input type="file">):** This input type is used for uploading files from the user's device. It allows the user to browse their computer and select one or more files to be uploaded.

h. **Date Input (<input type="date">):** This input type is used for capturing dates. It provides a date picker interface, allowing the user to select a date from a calendar.

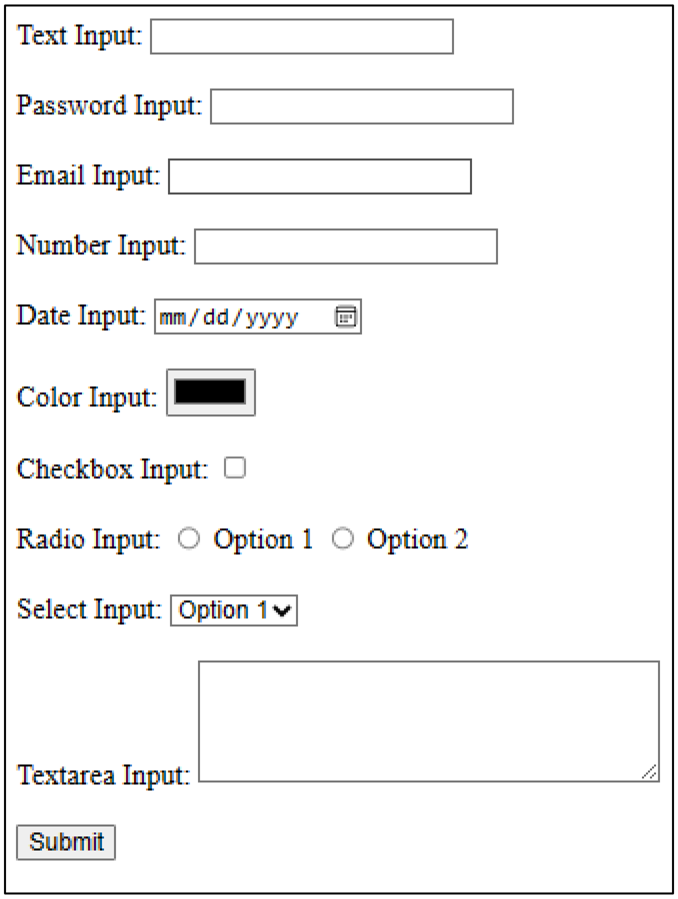
i. **Time Input (<input type="time">):** This input type is used for capturing time values. It provides a time picker interface, allowing the user to select a time from a dropdown or input the time manually.

j. **Color Input (<input type="color">):** This input type allows the user to select a color from a color picker. It displays a color palette or a color wheel, depending on the browser support.

Following example shows different types of input fields in a form.

```
<!DOCTYPE html>
<html>
<head>
  <title>Form with Different Input Types</title>
</head>
<body>
  <form>
    <label>Text Input:</label>
    <input type="text"><br><br>

    <label>Password Input:</label>
    <input type="password"><br><br>

    <label>Email Input:</label>
    <input type="email"><br><br>

    <label>Number Input:</label>
    <input type="number"><br><br>

    <label>Date Input:</label>
    <input type="date"><br><br>

    <label>Color Input:</label>
    <input type="color"><br><br>

    <label>Checkbox Input:</label>
    <input type="checkbox"><br><br>

    <label>Radio Input:</label>
    <input type="radio"> Option 1
    <input type="radio"> Option 2<br><br>

    <label>Select Input:</label>
    <select>
      <option value="option1">Option 1</option>
      <option value="option2">Option 2</option>
      <option value="option3">Option 3</option>
    </select><br><br>

    <label>Textarea Input:</label>
    <textarea rows="4" cols="30"></textarea><br><br>

    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

Output:

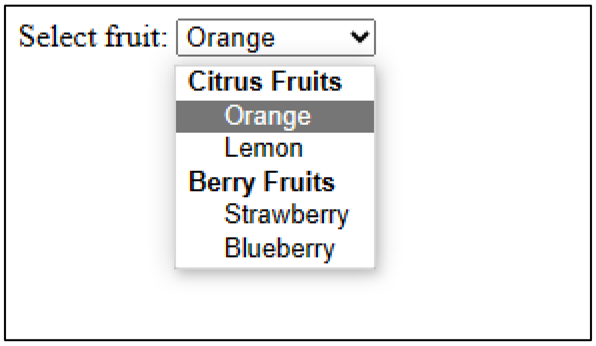


- **name:** Specifies the name of the input element. It is used to identify the input field when the form is submitted to the server.
- **value:** Sets the initial value of the input field. This attribute is optional but can be useful for pre-filling the input field with default values.
- **placeholder:** Specifies a short hint that describes the expected value of the input field. It is typically displayed as a grayed-out text within the input field.
- **required:** Indicates that the input field must be filled out before the form can be submitted. If the required attribute is present, the browser will prevent form submission if the field is empty.
- **disabled:** Disables the input field, preventing the user from interacting with it or entering data.
- **readonly:** Makes the input field read-only, allowing the user to see the value but not modify it.
- **maxlength:** Sets the maximum number of characters allowed in the input field.
- **min and max:** Used with certain input types like number or date to specify the minimum and maximum values allowed for the input.

Example:

```
<form>
  <label for="fruit">Select fruit:</label>
  <select id="fruit" name="fruit">
    <optgroup label="Citrus Fruits">
      <option value="orange">Orange</option>
      <option value="lemon">Lemon</option>
    </optgroup>
    <optgroup label="Berry Fruits">
      <option value="strawberry">Strawberry</option>
      <option value="blueberry">Blueberry</option>
    </optgroup>
  </select>
</form>
```

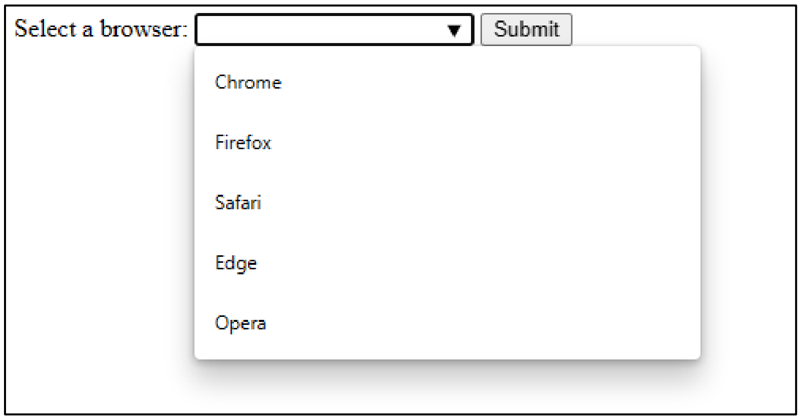
Outpt:



Example:

```
<form>
  <label>Select a browser:</label>
  <input type="text" id="browser" list="browsers">

  <datalist id="browsers">
    <option value="Chrome">
    <option value="Firefox">
    <option value="Safari">
    <option value="Edge">
    <option value="Opera">
  </datalist>

  <input type="submit" value="Submit">
</form>
```

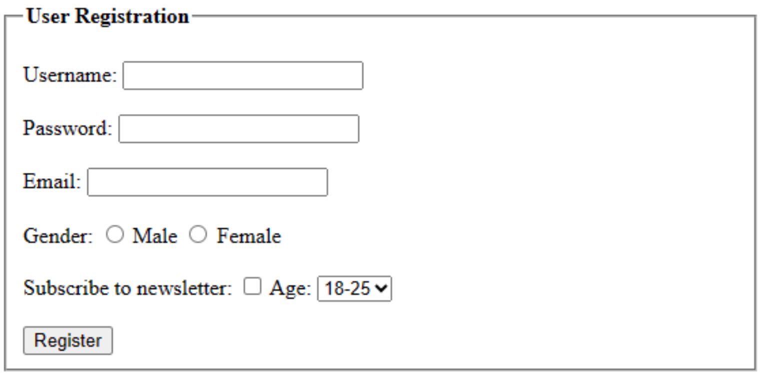
Output:



Example:

```
<form>
  <fieldset>
    <legend><b>User Registration</b></legend>
    <br>

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

    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required>
    <br><br>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>
    <br><br>

    <label for="gender">Gender:</label>
    <input type="radio" id="male" name="gender" value="male">
    <label for="male">Male</label>
    <input type="radio" id="female" name="gender" value="female">
    <label for="female">Female</label>
    <br><br>

    <label for="newsletter">Subscribe to newsletter:</label>
    <input type="checkbox" id="newsletter" name="newsletter" value="subscribe">

    <label for="age">Age:</label>
    <select id="age" name="age">
      <option value="18-25">18-25</option>
      <option value="26-35">26-35</option>
      <option value="36-45">36-45</option>
      <option value="46+">46+</option>
    </select>
    <br><br>

    <input type="submit" value="Register">
  </fieldset>
</form>
```

Output:



Named Input Fields:

Named input fields in HTML are used to assign a name to an input element so that its value can be identified and accessed when the form is submitted. The name attribute in HTML form input elements is essential because it identifies the data sent to the server when the form is submitted. When a form is submitted, the browser collects the values entered in the input fields and sends them to the server as part of an HTTP request.

The name attribute serves as the key that associates the input value with its corresponding data when the form is processed on the server side. It allows us to reference and access the submitted values in server-side scripts or when handling the form data.

For example,

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

The browser will send the value entered in the field as part of the form data, using the name attribute as the key. On the server side, you can retrieve the value of the username field using the appropriate method depending on the server-side language/framework being used. In PHP, we can access it using $_POST['username'], while in ASP.NET, we can access it through Request.Form["username"].

In summary, the name attribute is crucial for identifying and accessing form input values on the server side, allowing us to process the form data and perform necessary actions based on the user's input.

Multiple Line Text (<TEXTAREA> Tag):

The <textarea> element in HTML is used to create a multi-line text input field where users can enter and edit text. We use the following syntax to create a multiple line text input field using <textarea> tag.

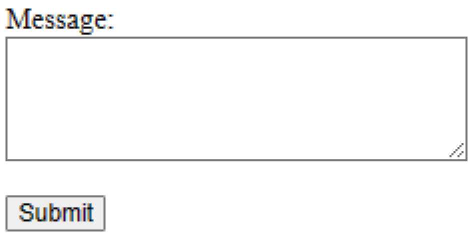
```
<textarea row='no. of rows' cols='no_of_columns'></textarea>
```

Example:

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

```
  <title>Multiple Lines Text Input</title>
</head>
<body>
  <label for="message-input">Message:</label><br>
  <textarea id="message-input" name="message" rows="4"
cols="30"></textarea><br><br>

  <input type="submit" value="Submit">
</body>
</html>
```

Output:



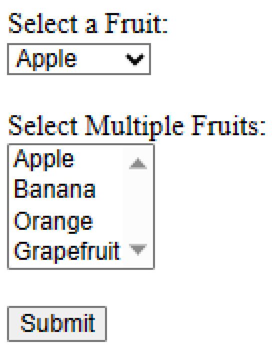
In this example, the <textarea> element is used to create a multi-line text input field. The rows attribute specifies the number of visible lines in the textarea, and the cols attribute determines the number of visible characters per line. You can adjust these values to fit your desired dimensions.

Dropdown and List Boxes:

We use <select> and <option> tags to create a dropdown an list box in html. Here's an example of creating a dropdown menu and a list box using the <select> element in HTML:

```
<!DOCTYPE html>
<html>
<head>
  <title>Dropdown and List Box Example</title>
</head>
<body>
  <label for="dropdown">Select a Fruit:</label><br>
  <select id="dropdown">
    <option value="apple">Apple</option>
    <option value="banana">Banana</option>
    <option value="orange">Orange</option>
    <option value="grapefruit">Grapefruit</option>
  </select><br><br>

  <label for="listbox">Select Multiple Fruits:</label><br>
  <select id="listbox" multiple>
    <option value="apple">Apple</option>
    <option value="banana">Banana</option>
    <option value="orange">Orange</option>
    <option value="grapefruit">Grapefruit</option>
  </select><br><br>

  <input type="submit" value="Submit">
</body>
</html>
```

Output:



Dropdown menu:

- The <select> element with the ID "dropdown" and the name "fruit" creates a dropdown menu.
- Each option is defined using the <option> element within the <select> element.
- The value attribute specifies the value associated with each option.
- Users can select a single option from the dropdown menu.

List box:

- The <select> element with the ID "listbox" and the name "fruits[]" creates a list box.
- The multiple attribute allows users to select multiple options from the list.
- Each option is defined using the <option> element, similar to the dropdown menu.
- Users can select multiple options from the list box.

When the form is submitted, the selected value from the dropdown menu will be sent as the value of the "fruit" field. For the list box, since multiple options can be selected, the values will be sent as an array.

Hidden Fields:

Hidden fields in HTML forms are used to store data that is not directly visible or editable by the user. They are useful for passing information between different pages or for storing values that need to be sent along with the form submission. The <input> elements with the type="hidden" attribute create hidden fields.

Each hidden field has a name attribute to identify the field when the form is submitted, and a value attribute to specify the value associated with the field.

Here's an example of using hidden fields in an HTML form:

```
<!DOCTYPE html>
<html>
<head>
  <title>Hidden Fields Example</title>
</head>
<body>
  <form action="process-form.php" method="POST">
    <input type="hidden" name="hidden-field1" value="Value 1">
    <input type="hidden" name="hidden-field2" value="Value 2">

    <label for="name-input">Name:</label>
    <input type="text" id="name-input" name="name"><br><br>

    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

When the form is submitted, the hidden fields and the visible input field will be sent as part of the form data. In this example, the hidden fields hidden-field1 and hidden-field2 will be sent with their respective values alongside the user-entered name.

Hidden fields are useful for carrying additional data that we want to include with the form submission without displaying it to the user. Keep in mind that hidden fields can be manipulated by savvy users, so avoid storing sensitive information or relying on them for security measures.

File Upload

To enable file uploads in an HTML form, you need to use the <form> element with the enctype attribute set to "multipart/form-data". Additionally, you'll need to include an <input> element of type "file" within the form. Here's an example:

```
<form action="upload.php" method="POST" enctype="multipart/form-data">

  <label for="file">Select a file:</label>

  <input type="file" id="file" name="file">

  <input type="submit" value="Upload">

</form>
```

In this example, the action attribute of the <form> element specifies the URL or script that will handle the form submission. In this case, it points to upload.php. The method attribute is set to "POST" to send the form data in the HTTP request body.

The <input> element of type "file" creates a file input field. When the user selects a file using the file picker dialog, the chosen file will be associated with the input field.

Once the form is submitted, the selected file will be included in the request payload. You can handle the file upload on the server-side using the appropriate programming language and process the uploaded file as needed.

Note that the server-side code (upload.php in this example) is responsible for handling the file upload and saving the file to a specified location or performing any additional processing required.

It's important to ensure that you validate and sanitize the uploaded files on the server-side to prevent any security vulnerabilities or malicious file uploads.

Button

The <button> tag is used to define a clickable button within an HTML form or any other context where a button is needed. Here's an example of how to use the <button> tag:

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

In this example, the <button> tag creates a button element with the label "Click me". The type="button" attribute specifies that it is a button element. The type attribute is optional, and if not specified, it defaults to "submit". However, using type="button" explicitly ensures that the button does not submit a form if it is placed inside one.

Radio button

A radio button is a form element in HTML that allows users to select only one option from a list of predefined options. It is represented by a small circular button that can be either selected (checked) or deselected (unchecked). Here's an example of how to create radio buttons in HTML:

```
<form>
  <input type="radio" id="option1" name="radioOptions" value="option1">
  <label for="option1">Option 1</label><br>
  <input type="radio" id="option2" name="radioOptions" value="option2">
  <label for="option2">Option 2</label><br>
  <input type="radio" id="option3" name="radioOptions" value="option3">
  <label for="option3">Option 3</label><br>
</form>
```

In the example above, we have three radio buttons with the names "radioOptions". The name attribute groups the radio buttons together, ensuring that only one option can be selected within the same group. The id attribute uniquely identifies each radio button, and the value attribute specifies the value that gets submitted when the form is submitted.

The <label> element is associated with each radio button using the for attribute, which matches the id of the corresponding radio button. This allows users to click on the label text to select the associated radio button.

When the user submits the form, the selected radio button's value will be sent to the server for further processing.

Checkbox

A checkbox is a form element in HTML that allows users to select one or more options from a list of choices. It is represented by a small square box that can be either checked or unchecked. Here's an example of how to create checkboxes in HTML:

```
<form>
  <input type="checkbox" id="option1" name="somename" value="option1">
  <label for="option1">Option 1</label><br>
  <input type="checkbox" id="option2" name=" somename" value="option2">
  <label for="option2">Option 2</label><br>
  <input type="checkbox" id="option3" name=" somename"  value="option3">
  <label for="option3">Option 3</label><br>
</form>
```

In the example above, we have three checkboxes with the names "checkboxOptions". Each checkbox operates independently, allowing users to select multiple options at once.

The id attribute uniquely identifies each checkbox, and the value attribute specifies the value that gets submitted when the form is submitted. The name attribute groups the checkboxes together, allowing multiple values to be submitted under the same name.

The <label> element is associated with each checkbox using the for attribute, which matches the id of the corresponding checkbox. This allows users to click on the label text to select or deselect the associated checkbox.

When the user submits the form, the values of the selected checkboxes will be sent to the server for further processing. On the server side, you can retrieve the selected values based on the name attribute to handle the form data appropriately.

Select Option

The <select> element in HTML is used to create a dropdown list or a selection box. It allows users to choose one option from a list of predefined options. Here's an example of how to create a select element with options in HTML:

```
<select id="fruit" name="fruit">
  <option value="apple">Apple</option>
  <option value="banana">Banana</option>
  <option value="orange">Orange</option>
  <option value="grape">Grape</option>
</select>
```

In the example above, we have a select element with the id "fruit" and the name "fruit". It contains four options: Apple, Banana, Orange, and Grape.

Each <option> element represents an option within the select element. The value attribute specifies the value associated with each option, which gets submitted when the form is submitted. The text between the opening and closing <option> tags is the visible text that appears in the dropdown list.

When the user selects an option from the dropdown list, the selected option's value will be sent to the server for further processing. On the server side, you can retrieve the selected value based on the name attribute to handle the form data appropriately.

Action Buttons

In HTML forms, you can use <button> elements as action buttons to perform specific actions related to the form. These buttons can submit the form, reset form fields, or trigger custom JavaScript functions. Here are examples of different types of action buttons within an HTML form:

a. **Submit Button:**

   The type="submit" attribute on the button indicates that it is a submit button. When clicked, it will submit the form to the specified action URL or file.

```
<form action="submit.php" method="post">
  <!-- form fields -->
  <button type="submit">Submit</button>
</form>
```

b. **Reset Button:**

   The type="reset" attribute on the button creates a reset button. When clicked, it will reset the form fields to their default values, clearing any user-entered data.

```
<form>
  <!-- form fields -->
  <button type="reset">Reset</button>
</form>
```

c. Custom Action Button:

To create a custom action button in HTML that triggers a JavaScript function, you can use the <button> element with the type="button" attribute and define the function you want to execute. Here's an example:

```
<button type="button" onclick="myFunction()">
     Custom Action
</button>

<script>
  function myFunction() {
    // Perform the desired action here
    alert("Custom action executed!");
  }
</script>
```

In the example above, we have a button with the label "Custom Action". The type="button" attribute indicates that it is a generic button without any default behavior.

The onclick attribute specifies the JavaScript function that will be executed when the button is clicked. In this case, the myFunction() function will be called.

Inside the myFunction() function, you can perform any desired actions. In this example, an alert message will be displayed when the button is clicked, indicating that the custom action has been executed.

You can customize the myFunction() function to perform the specific action you need. It could include manipulating the form data, interacting with other elements on the page, or making AJAX requests, among other possibilities.

Labeling Input Fields

In HTML, the <label> element is used to associate a label with a form element, such as an input field or a checkbox. The <label> element improves accessibility by providing a textual description or caption for the associated form element. Labeling input fields using the <label> element improves accessibility, as it provides a clear association between the label and the input field. It helps users understand the purpose of each input field, especially for users with assistive technologies or those who prefer keyboard navigation.

Here are a few ways to use the <label> element in HTML:

a. Labeling an input field using the for attribute:

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

In this example, the for attribute in the <label> element matches the id attribute of the input field. It establishes a connection between the label and the input field. When users click on the label, the associated input field receives focus.

b. Enclosing an input field within the <label> element:

```
<label>Email:
  <input type="email" name="email">
</label>
```

In this case, the input field is placed within the <label> element. The label text "Email:" is positioned alongside the input field. This method is suitable when the label is a simple text and closely associated with the input field.

c. Using the placeholder:

In HTML, the placeholder attribute is used to provide a temporary hint or example text within an input field. While the placeholder attribute can give visual cues to users, it is not a substitute for a proper label. However, you can combine the placeholder attribute with other methods to improve form usability.

Here's an example:

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

In the example above, the <label> element is used to provide a label for the input field. The for attribute in the label matches the id attribute of the input field, establishing the association. This allows users to click on the label, which sets focus on the corresponding input field.

The input field itself has the placeholder attribute, which displays the temporary text "Enter your username" as a hint or example inside the input field. It can provide additional guidance to users about the expected input.

Using the placeholder attribute alone is not sufficient for accessibility purposes, as it does not provide a permanent label. It is important to include a proper <label> element as well. This ensures that assistive technologies and users with disabilities can access the form correctly.

Grouping Related Fields

In HTML, the <fieldset> element is used to group related form fields together, creating a logical and visual container. It helps in organizing and structuring forms by grouping related inputs. The <fieldset> element establishes a boundary around the grouped form fields.

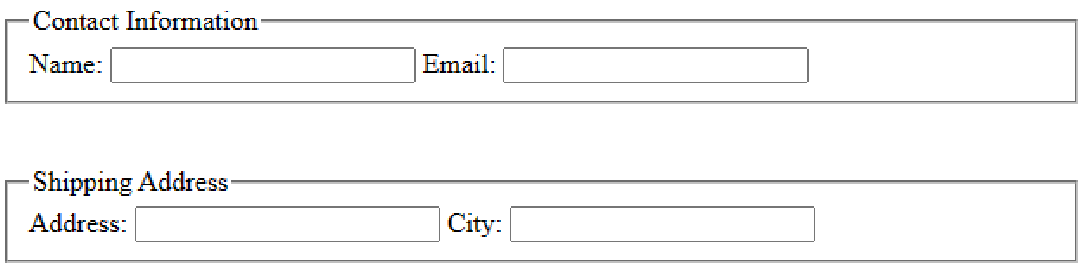
The <legend> element is used inside the <fieldset> element to provide a caption or title for the group of form fields. It acts as a heading or descriptive label for the <fieldset>, providing a clear indication of the purpose or category of the grouped form fields.

Together, <fieldset> and <legend> create a visual and semantic relationship between form fields. The <fieldset> element sets up the grouping, while the <legend> element provides a visible and accessible label for the group. This combination improves the accessibility and usability of forms, making it easier for users to understand the context and organization of the form fields.

Example:

```
<form>
  <fieldset>
    <legend>Contact Information</legend>

    <label for="name">Name:</label>
    <input type="text" id="name" name="name">

    <label for="email">Email:</label>
    <input type="email" id="email" name="email">

    <!-- More form fields related to contact information -->
  </fieldset>

  <fieldset>
    <legend>Shipping Address</legend>

    <label for="address">Address:</label>
    <input type="text" id="address" name="address">

    <label for="city">City:</label>
    <input type="text" id="city" name="city">

    <!-- More form fields related to shipping address -->
  </fieldset>

  <!-- Submit button and other form elements -->
</form>
```

Output:



In the example above, we use the <fieldset> element to group related form fields together. The <legend> element is placed inside the <fieldset> element and provides a title or description for the group.

The <label> elements are used to associate the text labels with their respective form fields using the for attribute. When users click on the labels, focus is automatically set to the associated form fields.

By utilizing <fieldset> and <legend>, you can visually and semantically group related form fields together, making the form more structured and easier for users to understand.

Additionally, CSS can be applied to <fieldset> and <legend> elements to style and customize their appearance, further enhancing the visual presentation of the form.

In summary, <fieldset> and <legend> are HTML elements that work together to group related form fields and provide a descriptive label for the grouping. They play a crucial role in structuring forms and improving user experience by visually and semantically organizing form elements.

Disabled and Read Only Field

Disabled Field: The disabled attribute is used to disable an input field, preventing users from interacting with or modifying its value. Disabled fields are not submitted when the form is submitted. Here's an example:

```
<input type="text" name="username" value="John Doe" disabled>
```

In this example, the text input field with the name "username" is disabled. The disabled attribute is added to the input element, preventing users from editing the value or interacting with the field.

Readonly Field: The readonly attribute is used to make an input field read-only, allowing users to view the value but not modify it. Read-only fields are submitted when the form is submitted. Here's an example:

```
<input type="text" name="email" value="john@example.com" readonly>
```

In this example, the text input field with the name "email" is set to read-only. The readonly attribute is added to the input element, preventing users from editing the value but still allowing them to view it.

It's important to note that disabled fields cannot be edited or focused, while read-only fields can still be focused and copied but not modified.

By using the disabled or readonly attribute, you can control the user interaction and the behavior of form fields, depending on your requirements.

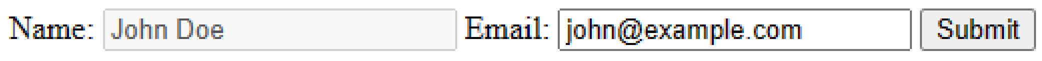
Example:

```
<form>
  <center>Example of disabled and readonly fields</center>
  <label for="name">Name:</label>
```

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

    <label for="email">Email:</label>
    <input type="text" value="john@example.com" readonly>
    <input type="submit" value="Submit">
  </form>
```

Output:

Example of disabled and readonly fields



In this example, we have two form fields: one that is disabled and another that is set to readonly.

The first input field for the name is disabled using the disabled attribute. Users cannot modify or interact with the field.

The second input field for the email is set to readonly using the readonly attribute. Users can view the value but cannot modify it.

When the form is submitted, both the disabled and readonly fields will be included in the form data.

Form Field Event Handlers

In HTML, event handlers are used to add interactivity and perform actions in response to user interactions with form elements. Event handlers are JavaScript functions that are triggered when a specific event occurs on a form element, such as clicking a button, submitting a form, or changing the value of an input field.

By utilizing event handlers, you can create dynamic and responsive forms that provide enhanced user experience. With JavaScript functions associated with event handlers, you can perform validation, manipulate form data, update the DOM, or interact with the server based on user input.

It's important to note that event handlers can be assigned directly to HTML elements using attributes such as onclick or onsubmit, or they can be attached dynamically using JavaScript to provide more flexibility and control over event handling.

Onclick event handler: The onclick event handler in HTML is used to execute a JavaScript function when an element, such as a button or a link, is clicked. It allows you to add interactivity and perform actions in response to user clicks. Here's an example of using the onclick event handler:

```
<button onclick="myFunction()">Click me</button>

<script>
  function myFunction() {
    // Perform the desired action here
    alert("Button clicked!");
  }
</script>
```

In the example above, we have a button with the label "Click me". The onclick attribute is used to associate the button with the myFunction() JavaScript function. When the button is clicked, the myFunction() function is executed.

Inside the myFunction() function, you can perform any desired actions or logic. In this example, an alert message will be displayed when the button is clicked, indicating that the button has been clicked.

You can customize the myFunction() function to perform specific actions based on your requirements. This could include manipulating the DOM, updating form data, making AJAX requests, or performing other interactions with the page.

The onclick event handler can be applied to various HTML elements, such as buttons, links (<a>), checkboxes, or radio buttons, allowing you to trigger custom actions in response to user clicks.

Similarly, there are numbers of event handlers in html. Following list shows the different event handlers available in html and their use.

Form Events:

onblur	Fires the moment that the element loses focus
onchange	Fires the moment when the value of the element is changed
oncontextmenu	Script to be run when right clicked
onfocus	Fires the moment when the element gets focus
oninput	Script to be run when an element gets user input

onsearch	Fires when the user writes something in a search field (for <input="search">)
onselect	Fires after some text has been selected in an element
onsubmit	Fires when a form is submitted

Keyboard Events:

Attribute	Description
onkeydown	Fires when a user is pressing a key
onkeypress	Fires when a user presses a key
onkeyup	Fires when a user releases a key

Mouse Events:

Attribute	Description
onclick	Fires on a mouse click on the element
ondblclick	Fires on a mouse double-click on the element
onmousedown	Fires when a mouse button is pressed down on an element
onmousemove	Fires when the mouse pointer is moving while it is over an element
onmouseout	Fires when the mouse pointer moves out of an element

onmouseover	Fires when the mouse pointer moves over an element
onmouseup	Fires when a mouse button is released over an element
onwheel	Fires when the mouse wheel rolls up or down over an element

Passing Form Data

In HTML, forms serve as a crucial means of user interaction on the web. When a user submits a form, it's essential to pass the data to the server for processing or storage. HTML provides the method attribute, which allows us to specify how form data should be transmitted. In this article, we'll explore the different values of the method attribute and their implications for passing form data.

a. **Get Method:** The method="get" attribute value signifies that form data should be appended to the URL as query parameters. When a user submits a form using the GET method, the resulting URL will contain the form data. This method is suitable for retrieving data from the server or when the data is not sensitive. However, it's worth noting that the data becomes visible to users and is stored in browser history. The server can access the data from the query parameters, making it accessible for processing.

   Points on Get Method:

   - Appends form-data into the URL in name/value pairs
   - The length of a URL is limited (about 3000 characters)
   - Never use GET to send sensitive data! (will be visible in the URL)
   - Useful for form submissions where a user wants to bookmark the result

   Example:

```
<form action="process-data.php" method="get">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name">

  <label for="email">Email:</label>
  <input type="email" id="email" name="email">

  <input type="submit" value="Submit">
</form>
```

   In this example, the form uses the GET method with the method="get" attribute. When the form is submitted, the form data is appended to the URL as query parameters. For instance, if the user enters "John" as the name and "john@example.com" as the email, the resulting URL will be something like: process-data.php?name=John&email=john@example.com. The server-side script specified in the action attribute can access the data using the query parameters.

b. Post Method: Using method="post" indicates that form data should be sent in the body of the HTTP request. Unlike the GET method, the form data is not directly visible in the URL or browser history. This makes it more secure and suitable for sensitive data or when large amounts of information need to be transmitted. The form data is accessible on the server-side by parsing the request body. The POST method is commonly used when creating new records, updating existing data, or submitting sensitive information such as passwords or credit card details.

Points on Post Method:

- Appends form-data inside the body of the HTTP request (data is not shown in URL)
- Has no size limitations
- Form submissions with POST cannot be bookmarked
- Suitable to submit confidential data such as password.

Example:

```
<form action="process-data.php" method="post">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name">

  <label for="email">Email:</label>
  <input type="email" id="email" name="email">

  <input type="submit" value="Submit">
</form>
```

In this example, the form uses the POST method with the method="post" attribute. When the form is submitted, the form data is sent in the body of the HTTP request. The data is not directly visible in the URL. The server-side script specified in the action attribute can access the form data from the request body. This method is suitable for sensitive data or when large amounts of information need to be transmitted.

Selecting the appropriate method depends on the specific requirements of your form. If you need to retrieve or search for data, the GET method is suitable. It's also ideal for bookmarkable links or when you want to share the URL with others. On the other hand, if you are submitting data that modifies server-side resources or contains sensitive information, the POST method is recommended for its security and data privacy benefits.

The method attribute in HTML provides control over how form data is passed from the client to the server. The GET method appends form data to the URL, allowing easy retrieval but exposing the data to users and storing it in browser history. The POST method sends form data in the body of the HTTP request, providing enhanced security and privacy. By understanding the implications of each method and their use cases, you can make informed decisions when designing and implementing HTML forms for passing form data.

Cascading Style Sheet (CSS)

Introduction

Importance

Different Approaches to CSS

Using Multiple Approaches

Linking to Style Information in Separate File

Setting up Style Information

Using the <LINK> Tag

Embedding Style Information

Using <STYLE> Tag

Inline Style Information