Author: Lokang Jackson

Book Name: HTML and CSS

ordered list

An ordered list in HTML is a list of items that are numbered, typically using the "ol" tag. Each item in the list is represented by the "li" tag. Here is an example of an ordered list in HTML:

```
<ol>
 <li>Item 1</li>
 <li>Item 2</li>
 <li>Item 3</li>
</ol>
```

This will create a numbered list with the items "Item 1", "Item 2", and "Item 3".

CSS (Cascading Style Sheets) is used to control the presentation of the HTML elements on a web page. You can use CSS to change the appearance of the ordered list, such as the font size or color. Here is an example of how you can change the font size of the list items in an ordered list:

```
<ol>
 <li style="font-size: 20px;">Item 1</li>
 <li style="font-size: 20px;">Item 2</li>
 <li style="font-size: 20px;">Item 3</li>
</ol>
```

This will set the font size of the list items to 20px. You can also use classes or ID's to select the elements, which is more efficient and maintainable.

```
<style>
.large-font{
 font-size:20px;
}
</style>
<ol>
 <li class="large-font">Item 1</li>
 <li class="large-font">Item 2</li>
 <li class="large-font">Item 3</li>
</ol>
```

This way you can easily change the font size for all elements with class large-font in one place.

tag

HTML tags are used to create the structure and layout of a web page. There are many different types of HTML tags, each with a specific purpose. Here are a few examples:

The tag is used to create a paragraph. The text enclosed between the opening and closing tags will be displayed as a separate paragraph

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

The <img> tag is used to insert an image. The "src" attribute is used to specify the source of the image

```
<img src="image.jpg" alt="image">
```

The <a> tag is used to create a hyperlink. The "href" attribute is used to specify the destination of the link.

```
<a href="https://www.example.com">Visit Example.com</a>
```

The tag is a container for HTML elements, which can be styled with CSS.

```
<div class="container">
  <p>This is a paragraph inside a div container.</p>
  <img src="image.jpg" alt="image">
  <a href="https://www.example.com">Visit Example.com</a>
</div>
```

The <header> tag defines the header of a document, usually contains the site title or logo.

```
<header>
 <h1>Welcome to my website</h1>
</header>
```

The <nav> tag defines the navigation links of a documen

```
<nav>
 <a href="#home">Home</a> |
 <a href="#about">About</a> |
 <a href="#services">Services</a> |
 <a href="#contact">Contact</a>
</nav>
```

These are just a few examples of the many different types of HTML tags that exist. Each tag has a specific purpose and can be used in combination to create a well-structured and visually appealing web page.

class

A "div" element in HTML is a container for other HTML elements. It can be used to group elements together for styling or scripting purposes. The "class" attribute is used to assign a class name to the "div" element, which can be used to select and style the element with CSS.

Here is an example of a "div" element with a class name:

```
<div class="container">
 <p>This is some text inside the container div.</p>
</div>
```

In this example, the "div" element has a class name of "container". This class name can be used to select the div with CSS and apply styles to it:

```
.container {
 width: 50%;
 margin: 0 auto;
 background-color: #f2f2f2;
}
```

This CSS will give a width of 50% to the div and centers it on the screen, and also changes the background color of the div to #f2f2f2.

You can also assign multiple classes to a single element by separating them with a space:

```
<div class="container box">
 <p>This is some text inside the container div.</p>
</div>
```

and use CSS to style them separately

```
.container {
 width: 50%;
 margin: 0 auto;
}
.box{
 background-color: #f2f2f2;
}
```

In this way you can apply different styles to the same element, which makes it more flexible and maintainable.

It's worth noting that the div element itself does not have any specific meaning, it's mostly used to group elements together and apply styles to that group.

table

A table in HTML is used to display data in a grid format, with rows and columns. The main elements used to create a table in HTML are the "table", "tr" (table row), "th" (table header), and "td" (table data) tags. Here is an example of a simple table in HTML:

```
<table>
 <tr>
   <th>Header 1</th>
   <th>Header 2</th>
 </tr>
 <tr>
   <td>Row 1, Cell 1</td>
   <td>Row 1, Cell 2</td>
 </tr>
 <tr>
   <td>Row 2, Cell 1</td>
   <td>Row 2, Cell 2</td>
 </tr>
</table>
```

This will create a table with two columns (Header 1 and Header 2) and two rows (Row 1 and Row 2).

CSS (Cascading Style Sheets) can be used to control the presentation of the HTML table elements, such as the font size, color, or border. Here is an example of how you can change the font size of the table data:

```
<style>
td{
 font-size:20px;
}
</style>
<table>
 <tr>
   <th>Header 1</th>
   <th>Header 2</th>
 </tr>
 <tr>
   <td>Row 1, Cell 1</td>
   <td>Row 1, Cell 2</td>
 </tr>
 <tr>
   <td>Row 2, Cell 1</td>
   <td>Row 2, Cell 2</td>
 </tr>
</table>
```

This will set the font size of the table data to 20px. You can also use CSS to control the table's border.

```
table {
 border-collapse: collapse;
}
th, td {
 border: 1px solid black;
}
```

This will create a border around the table, and around each cell, giving a grid-like structure to the table

It's also possible to style specific cells or rows by using classes or ID's to select those elements.

```
<style>
.highlight{
 background-color: yellow;
}
</style>
<table>
 <tr>
   <th>Header 1</th>
   <th>Header 2</th>
 </tr>
 <tr>
   <td class="highlight">Row 1, Cell 1</td>
   <td>Row 1, Cell 2</td>
 </tr>
 <tr>
   <td>Row 2, Cell 1</td>
   <td>Row 2, Cell 2</td>
 </tr>
</table>
```

This will highlight the first cell of the first row with a yellow background.

As you can see, you can use both HTML and CSS to create and style tables on your web pages, allowing you to display data in a structured, organized way.

grid

CSS Grid Layout, often simply referred to as Grid, provides a two-dimensional grid-based layout system that allows you to layout items into rows and columns, and it's a fantastic tool for building complex, responsive designs. This model gives more control over the various grid items and their placement in a predefined layout grid.

Basic Concepts:

Grid Container: The element on which **display: grid** or **display: inline-grid** is applied. It’s the direct parent of all the grid items.

Grid Items: The children of the grid container.

Grid Lines: The dividing lines that make up the structure of the grid. They can be horizontal or vertical ("rows" or "columns").

Grid Tracks: The space between two adjacent grid lines. Can be horizontal (rows) or vertical (columns).

Grid Cell: A single unit of the grid.

Main Properties:

Grid Container Properties:

display: Define a container as a grid container with **grid** or **inline-grid**.

grid-template-rows / **grid-template-columns**: Define the columns/rows of the grid.

grid-gap: Shorthand for **grid-row-gap** and **grid-column-gap**, specifies the size of the gap between the rows/columns.

grid-row / **grid-column**: Position and size an item in terms of row and column lines.

justify-items: Align grid items along the row (inline) axis.

align-items: Align grid items along the column (block) axis.

justify-content: Align the whole grid along the row (inline) axis.

align-content: Align the whole grid along the column (block) axis.

Grid Item Properties:

grid-column-start / **grid-column-end**: Determine where a grid item starts/ends in terms of column grid lines.

grid-row-start / **grid-row-end**: Determine where a grid item starts/ends in terms of row grid lines.

grid-area: A shorthand for **grid-row-start**, **grid-column-start**, **grid-row-end**, and **grid-column-end**.

justify-self: Align an item inside a cell along the row (inline) axis.

align-self: Align an item inside a cell along the column (block) axis.

Example:

css

```
/* Container */
.grid-container {
 display: grid;
 grid-template-rows: 100px 200px;
 grid-template-columns: 1fr 2fr;
 grid-gap: 10px;
```

```
}
/* Items */
.grid-item {
 grid-column: span 2;
}
```

html

```
<div class="grid-container">
 <div class="grid-item">Item 1</div>
 <div class="grid-item">Item 2</div>
 <div class="grid-item">Item 3</div>
</div>
```

Usage:

CSS Grid is primarily used for laying out larger scale layouts that require more control over the positioning of elements and their size, without relying on a strict flow of content. It is incredibly useful for any kind of two-dimensional layout whether that be a page or a specific pattern/component.

Browser Support:

CSS Grid is widely supported in modern browsers, making it a reliable choice for web design.

CSS Grid and Flexbox can work together to create a robust layout system for your web projects. Grid is excellent for two-dimensional layouts, while Flexbox is superb for one-dimensional layouts. Both provide more flexibility and control in web design, without having to resort to frameworks or additional coding.

flexbox

Flexbox, or the Flexible Box Layout, is a layout model in CSS that allows you to design complex layout structures with a more efficient and predictable way than traditional models, especially when you want to distribute space and align items in complex layouts and when the size of your items is unknown or dynamic.

Basic Concepts:

Flex Container: The parent element in which **display: flex** or **display: inline-flex** is applied. It becomes a flex container and its children become flex items.

Flex Items: Direct children of the flex container. They can be laid out, aligned, and justified in many ways, quickly and with predictable results, even in complex layouts.

Main Properties:

Flex Container Properties:

display: Specifies the type of rendering box. Use **flex** (renders as a block) or **inline-flex** (renders as inline).

flex-direction: Defines the direction of the main axis. The values could be **row** (default), **row-reverse**, **column**, and **column-reverse**.

justify-content: Aligns items on the main axis. Values can be **flex-start**, **flex-end**, **center**, **space-between**, **space-around**, etc.

align-items: Aligns items on the cross axis. Values are **flex-start**, **flex-end**, **center**, **baseline**, and **stretch**.

flex-wrap: By default, items will try to fit on one line. You can change that with **nowrap**, **wrap**, or **wrap-reverse**.

align-content: Aligns flex lines when there is extra space in the cross-axis. It takes similar values as **align-items** and **justify-content**.

Flex Item Properties:

flex-grow: A number that defines how much of the available space inside the flex container the item should take up.

flex-shrink: A number that defines how the item will shrink relative to the rest of the flex items in the container when there is not enough space.

flex-basis: Defines the default size of an element before the remaining space is distributed according to the flex-grow and flex-shrink factors.

flex: A shorthand for **flex-grow**, **flex-shrink**, and **flex-basis**.

align-self: Allows the default alignment to be overridden for individual flex items.

order: By default, flex items are laid out in the source order. However, **order** property can control the order in which they appear in the flex container.

Example:

css

```
/* Container */
.flex-container {
 display: flex;
 justify-content: space-between;
 align-items: center;
}
/* Items */
.flex-item {
 flex-grow: 1;
 flex-basis: auto;
}
```

Html

```
<div class="flex-container">
 <div class="flex-item">Item 1</div>
 <div class="flex-item">Item 2</div>
 <div class="flex-item">Item 3</div>
</div>
```

Usage:

Flexbox is widely used for designing components and complex layout structures, especially when striving for a responsive design. It's a powerful tool and offers solutions to many of the most common problems and tricky parts of CSS layout and beyond.

Browser Support:

Flexbox is widely supported in all modern browsers, so you can use it confidently in your projects.

These resources will assist you to understand and utilize Flexbox in your web projects more efficiently.

unordered list

An unordered list in HTML is a list of items that are not numbered, typically using the "ul" tag. Each item in the list is represented by the "li" tag. Here is an example of an unordered list in HTML:

```
<ul>
 <li>Item 1</li>
 <li>Item 2</li>
 <li>Item 3</li>
</ul>
```

This will create a bullet-pointed list with the items "Item 1", "Item 2", and "Item 3".

CSS (Cascading Style Sheets) can also be used to control the presentation of the HTML elements on a web page. You can use CSS to change the appearance of the unordered list, such as the font size or color. Here is an example of how you can change the font size of the list items in an unordered list:

```
<style>
.large-font{
 font-size:20px;
}
</style>
<ul>
 <li class="large-font">Item 1</li>
 <li class="large-font">Item 2</li>
 <li class="large-font">Item 3</li>
</ul>
```

This will set the font size of the list items to 20px. Also you can use CSS to change the bullet points to something else like images or icons.

```
ul {
 list-style-type: none;
}
ul li::before {
 content: "";
 background-image: url(path/to/image.png);
 background-repeat: no-repeat;
 display: inline-block;
 width: 20px;
 height: 20px;
 margin-right: 10px;
}
```

This will change the bullet point to the image you specified in the background-image property.

id

An "id" attribute in HTML is used to uniquely identify an element on a web page. It is similar to the "class" attribute in that it can be used to select and style the element with CSS. However, while a class can be assigned to multiple elements, an id can only be used once on a page.

Here is an example of a "div" element with an id:

```
<div id="header">
 <h1>Welcome to my website</h1>
</div>
```

In this example, the "div" element has an id of "header". This id can be used to select the div with CSS and apply styles to it:

```
#header {
 background-color: #f2f2f2;
 text-align: center;
}
```

This CSS will change the background color of the div to #f2f2f2 and center the text inside the div.

An id is used to select a unique element, therefore it's important that the id is used only once in the page, otherwise it will break the expected behavior.

As you can see, you can use both HTML and CSS to create and style elements on your web pages, and ids can be used to select and style unique elements.

audio

To create an anchor that links to an audio file in HTML, you can use the **<audio>** element along with the **src** attribute. Here is an example:

```
<audio src="audiofile.mp3" controls></audio>
```

The **controls** attribute includes the audio player controls such as play, pause, and volume.

You can also specify additional attributes to customize the audio player. For example:

```
<audio src="audiofile.mp3" controls preload="none" loop>
 Your browser does not support the audio element.
</audio>
```

The **preload** attribute controls whether or not the audio file should be loaded when the page loads. The possible values are **none**, **metadata**, and **auto**. The **loop** attribute causes the audio file to start over again when it finishes playing.

If you want to create an anchor that links to the audio file, you can use the **<a>** element as follows:

```
<a href="audiofile.mp3">Play audio</a>
```

This will create a link that, when clicked, will cause the audio file to be played.

Checkbox

A checkbox in an HTML form is used to allow a user to make multiple selections from a list of options. It is represented by the **<input type="checkbox">** element.

Here is an example of an HTML form with checkboxes:

```
<form>
 <label for="item1">Item 1</label>
 <input type="checkbox" id="item1" name="item1">
 <label for="item2">Item 2</label>
 <input type="checkbox" id="item2" name="item2">
 <label for="item3">Item 3</label>
 <input type="checkbox" id="item3" name="item3">
 <input type="submit" value="Submit">
</form>
```

In this example, the form has three checkboxes with the labels "Item 1", "Item 2", and "Item 3". Each checkbox is represented by an **<input>** element with the **type** attribute set to "checkbox". The **id** attribute is used to associate the checkbox with its corresponding label, and the **name** attribute is used to identify the checkbox when the form is submitted.

Audio

An audio file in HTML can be embedded on a webpage using the **<audio>** element. The **src** attribute is used to specify the source of the audio file, which can be a URL or a relative file path.

Here is an example of how to embed an audio file on a webpage:

```
<audio controls>
 <source src="audio/example.mp3" type="audio/mpeg">
 <source src="audio/example.ogg" type="audio/ogg">
```

```
 Your browser does not support the audio element.
</audio>
```

In this example, the **<audio>** element is used to embed the audio file, with the **controls** attribute to display the audio controls (play, pause, volume, etc). The **<source>** elements are used to specify the source of the audio file. In this case, two sources are provided, one in mp3 format and another in ogg format. The **type** attribute is used to specify the audio file format. The text between the **<audio>** tags is displayed in case the browser doesn't support the audio element.

You can also use **autoplay** attribute to automatically play the audio when the page loads.

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

Additionally, you can use **loop** attribute to play the audio in loop.

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

It's important to note that, not all browsers support the same audio file formats. MP3 and Ogg are the most widely supported formats.

video

A video file in HTML can be embedded on a webpage using the **<video>** element. The **src** attribute is used to specify the source of the video file, which can be a URL or a relative file path.

Here is an example of how to embed a video file on a webpage:

```
<video width="320" height="240" controls>
 <source src="video/example.mp4" type="video/mp4">
 <source src="video/example.webm" type="video/webm">
 Your browser does not support the video element.
</video>
```

In this example, the **<video>** element is used to embed the video file, with the **width** and **height** attributes to specify the size of the video player. The **controls** attribute is used to display the video controls (play, pause, volume, etc). The **<source>** elements are used to specify the source of the video file. In this case, two sources are provided, one in mp4 format and another in webm format. The **type** attribute is used to specify the video file format. The text between the **<video>** tags is displayed in case the browser doesn't support the video element.

drop-down

A dropdown, also known as a select box or drop-down menu, is a way to present a list of options to a user in an HTML form. The user can select one option from the list. The **<select>** element is used to create a dropdown menu in HTML.

Here is an example of an HTML form with a dropdown menu:

```
<form>
 <label for="color">Select a color:</label>
 <select id="color" name="color">
   <option value="red">Red</option>
   <option value="green">Green</option>
   <option value="blue">Blue</option>
 </select>
 <input type="submit" value="Submit">
</form>
```

In this example, the form has a dropdown menu with the label "Select a color". The dropdown menu is represented by a **<select>** element with the **id** attribute set to "color" and the **name** attribute set to "color". Inside the select element, there are three **<option>** elements, each representing a color choice. The **value** attribute is used to identify the option when the form is submitted.

form

An HTML form is a way to collect user input on a web page. Forms are created using the **<form>** element, and various form elements such as text fields, checkboxes, radio buttons, and dropdown menus can be added to the form using various **<input>** elements. When a user submits the form, the data entered into the form is sent to the server for processing.

Here is an example of a simple HTML form:

```
<form action="submit-form.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">
 <label for="password">Password:</label>
 <input type="password" id="password" name="password">
 <input type="submit" value="Submit">
</form>
```

In this example, the form has a text field for the user's name, an email field, and a password field. Each field is represented by an **<input>** element with the appropriate **type** attribute. The **id** attribute is used to associate the field with its corresponding label, and the **name** attribute is used to identify the field when the form is submitted. The form's **action** attribute is set to "submit-form.php" which tells the browser where to send the form data when the form is submitted. The **method** attribute is set to "post" which tells the browser to send the form data as a post request.

When the user submits the form, the data entered into the fields is sent to the server-side script

specified in the form's **action** attribute (submit-form.php in this case) for processing. The server-side script can then use a language like PHP to access the form data and do things like insert it into a database, send an email, or perform some other type of processing.

It's important to note that form validation should be done before submitting the form to the server-side script. It can be done by using JavaScript on the client-side or by using a server-side programming language like PHP or Python.

RadioButton

A radio button in an HTML form is used to allow a user to make a single selection from a list of mutually exclusive options. It is represented by the **<input type="radio">** element.

Here is an example of an HTML form with radio buttons:

```
<form>
 <label for="gender_male">Male</label>
 <input type="radio" id="gender_male" name="gender" value="male">
 <label for="gender_female">Female</label>
 <input type="radio" id="gender_female" name="gender" value="female">
 <label for="gender_other">Other</label>
 <input type="radio" id="gender_other" name="gender" value="other">
 <input type="submit" value="Submit">
</form>
```

In this example, the form has three radio buttons with the labels "Male", "Female", and "Other". Each radio button is represented by an **<input>** element with the **type** attribute set to "radio". The **id** attribute is used to associate the radio button with its corresponding label, and the **name** attribute is used to group the radio buttons together so that only one can be selected at a time. The **value** attribute is used to identify the selected option when the form is submitted.

Also, you can make a radio button pre-selected by adding **checked** attribute to the desired radio button element.

```
<input type="radio" id="gender_male" name="gender" value="male" checked>
```

TextArea

A textarea in an HTML form is used to allow a user to enter multiple lines of text. It is represented by the **<textarea>** element.

Here is an example of an HTML form with a textarea:

```
<form>
 <label for="message">Leave a message:</label>
 <textarea id="message" name="message" rows="4" cols="50"></textarea>
 <input type="submit" value="Submit">
</form>
```

In this example, the form has a textarea element with the label "Leave a message", represented by

the **<textarea>** element. The **id** attribute is used to associate the textarea with its corresponding label, and the **name** attribute is used to identify the textarea when the form is submitted. The **rows** and **cols** attributes are used to specify the number of rows and columns in the textarea.

image

To include an image in an HTML document, you can use the **<img>** element. The **src** attribute specifies the path to the image file, and the **alt** attribute provides alternative text for the image:

```
<img src="image.jpg" alt="A description of the image">
```

You can also specify the width and height of the image using the **width** and **height** attributes:

```
<img src="image.jpg" alt="A description of the image" width="100"
height="100">
```

If you want to create a link to the image, you can use the **<a>** element:

```
<a href="image.jpg"><img src="image.jpg" alt="A description of the
image"></a>
```

This will create a clickable image that, when clicked, will take the user to the image file.

You can also use the **<figure>** and **<figcaption>** elements to add a caption to the image:

```
<figure>
 <img src="image.jpg" alt="A description of the image">
 <figcaption>Caption for the image</figcaption>
</figure>
```

text

Anchor text is the visible, clickable text in a hyperlink. It is the text that is displayed to the user and is usually underlined to indicate that it is a link.

To create a hyperlink in HTML, you can use the **<a>** element and the **href** attribute. The **href** attribute specifies the destination of the link, and the text between the **<a>** and **** tags is the anchor text.

Here is an example of a hyperlink with anchor text:

```
<a href="https://www.example.com">Example Website</a>
```

In this example, "Example Website" is the anchor text. When the user clicks on the anchor text, they will be taken to the URL specified in the **href** attribute (https://www.example.com).

You can also use the **title** attribute to add a tooltip to the anchor text:

```
<a href="https://www.example.com" title="Visit the Example
Website">Example Website</a>
```

When the user hovers over the anchor text, the tooltip "Visit the Example Website" will be displayed.

video

To include a video in an HTML document, you can use the **<video>** element along with the **src** attribute. Here is an example:

```
<video src="video.mp4" controls></video>
```

The **controls** attribute includes the video player controls such as play, pause, and volume.

You can also specify additional attributes to customize the video player. For example:

```
<video src="video.mp4" controls preload="none" loop>
 Your browser does not support the video element.
</video>
```

The **preload** attribute controls whether or not the video should be loaded when the page loads. The possible values are **none**, **metadata**, and **auto**. The **loop** attribute causes the video to start over again when it finishes playing.

You can also use the **<source>** element to specify multiple video sources, in case the browser does not support the first video format:

```
<video controls>
 <source src="video.mp4" type="video/mp4">
 <source src="video.webm" type="video/webm">
 Your browser does not support the video element.
</video>
```

In this example, the browser will try to play the MP4 video first. If it is not supported, it will try to play the WebM video. If neither format is supported, the text "Your browser does not support the video element." will be displayed.

2D

To draw 2D shapes in HTML and CSS, you can use the **div** element and apply styles to it using CSS.

Here is an example of how to draw a square using HTML and CSS:

```
<div style="width: 100px; height: 100px; background-color: red;"></div>
```

This will create a div element with a width of 100 pixels and a height of 100 pixels, and apply a red background color to it. This will create a red square on the web page.

You can also use CSS properties such as **border-radius** to create rounded corners on the square:

```
<div style="width: 100px; height: 100px; background-color: red; border-
radius: 20px;"></div>
```

To draw other shapes, you can use the **border** property to create the desired shape. For example, to draw a triangle, you can use the following CSS:

```
.triangle {  width: 0;  height: 0;  border-left: 50px solid transparent;
border-right: 50px solid transparent;  border-bottom: 100px solid red; }
```

Then, you can apply this class to a **div** element in your HTML:

```
<div class="triangle"></div>
```

This will create a red triangle on the web page.

You can also use the **clip-path** property to create more complex shapes. For example, to create a star shape, you can use the following CSS:

```
.star {  width: 200px;  height: 200px;  background-color: yellow;  clip-
path: polygon(50% 0%, 61% 35%, 98% 35%, 68% 57%, 79% 91%, 50% 70%, 21%
91%, 32% 57%, 2% 35%, 39% 35%); }
```

Then, you can apply this class to a **div** element in your HTML:

```
<div class="star"></div>
```

This will create a yellow star on the web page.<!DOCTYPE html>

```
Complete code
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>2D shapes</title>
  <style>
    .triangle {
      width: 0;
```

```
      height: 0;
      border-left: 50px solid transparent;
      border-right: 50px solid transparent;
      border-bottom: 100px solid red;
    }

    .star {
      width: 200px;
      height: 200px;
      background-color: yellow;
      clip-path: polygon(50% 0%, 61% 35%, 98% 35%, 68% 57%, 79% 91%, 50%
70%, 21% 91%, 32% 57%, 2% 35%, 39% 35%);
    }

  </style>
</head>
<body>
<div style="width: 100px; height: 100px; background-color: red;"></div>
<br>
<div style="width: 100px; height: 100px; background-color: red; border-
radius: 20px;"></div>
<br>
<div class="triangle"></div>
<br>
<div class="star"></div>
</body>
</html>
```

3d

HTML and CSS are primarily used for creating and styling two-dimensional content on the web. However, it is possible to create the illusion of three-dimensional shapes using CSS.

Here is an example of how you can use CSS to create a 3D cube:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    /* Define the container for the cube */
    .cube {
      width: 200px;
      height: 200px;
      position: relative;
      transform-style: preserve-3d;
      transform: rotateX(45deg) rotateY(45deg);
      margin: auto;
      margin-top: 100px;
    }
```

```
    /* Define the individual faces of the cube */
    .face {
      position: absolute;
      width: 200px;
      height: 200px;
      border: 1px solid black;
      /* Set the background color of each face */
      background-color: #eee;
    }

    /* Position the faces */
    .front {
      transform: translateZ(100px);
    }
    .back {
      transform: rotateY(180deg) translateZ(100px);
    }
    .left {
      transform: rotateY(-90deg) translateZ(100px);
    }
    .right {
      transform: rotateY(90deg) translateZ(100px);
    }
    .top {
      transform: rotateX(90deg) translateZ(100px);
    }
    .bottom {
      transform: rotateX(-90deg) translateZ(100px);
    }
  </style>
</head>
<body>
<!-- Create the cube container -->
<div class="cube">
  <!-- Add the individual faces -->
  <div class="face front">Front</div>
  <div class="face back">Back</div>
  <div class="face left">Left</div>
  <div class="face right">Right</div>
  <div class="face top">Top</div>
  <div class="face bottom">Bottom</div>
</div>
</body>
</html>
```

This code creates a container element with the class "cube" and six child elements with the class "face". The "cube" element is transformed to give the illusion of three-dimensional space, and the "face" elements are positioned and rotated within this space to form a cube. The background color of each face can be customized using the "background-color" property.

Note that this is just one way to create a 3D effect using CSS. There are many other techniques that

you can use, such as using the "perspective" property, applying transforms to individual elements, and using CSS animation. Consult the documentation for more information.

Comments in html and css

Comments are essential in both CSS and HTML for documenting your code and making it more readable for yourself and others who might work on it in the future. Here's how you can add comments in both:

HTML Comments

In HTML, comments are added between <!-- and -->. Anything between these tags will be ignored by the browser.

```
<!-- This is a single-line comment in HTML -->
<!--
   This is a multi-line comment in HTML.
   It can span multiple lines.
-->
```

CSS Comments

In CSS, comments are added between /* and */. Similarly, anything between these symbols will be ignored by the browser.

```
/* This is a single-line comment in CSS */
/*
   This is a multi-line comment in CSS.
   It can span multiple lines.
*/
```

Example of Comments in HTML and CSS

Here is a simple example that shows how to use comments in both HTML and CSS:

HTML Example

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
   <!-- Link to external CSS file -->
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <!-- Main container for the page content -->
   <div class="container">
       <!-- Heading for the page -->
```

```
        <h1>Welcome to My Website</h1>
        <!-- Paragraph with some text -->
        <p>This is a sample paragraph.</p>
    </div>
</body>
</html>
```

CSS Example

```
/* Reset some default styles */
body {
    margin: 0;
    padding: 0;
    font-family: Arial, sans-serif; /* Set default font */
}
/* Main container styling */
.container {
    padding: 20px;
    background-color: #f0f0f0; /* Light gray background */
}
/* Heading style */
h1 {
    color: #333; /* Dark gray text color */
}
/* Paragraph style */
p {
    font-size: 16px; /* Set font size */
    color: #666; /* Medium gray text color */
}
```

Comments are a good practice in coding as they help clarify the purpose of the code, especially for complex sections or for team projects.

text and font properties

font-family

The **font-family** property in CSS specifies the font family for the text. It allows you to choose a specific font for your text content, rather than relying on the default font for the web browser.

For example:

```
p {
 font-family: Arial, sans-serif;
}
```

In the example above, the font-family is set to "Arial", and if "Arial" is not available, it will fall back to a generic sans-serif font.

You can specify multiple font families, separated by commas, in case the first choice is not available.

This allows you to provide a backup font in case the first choice is not installed on the user's device.

It's important to remember that not all fonts are available on all devices, so it's a good practice to specify multiple fonts to ensure that the text will be displayed as intended on all devices.

font-size

The **font-size** property in CSS is used to specify the size of the font. The size can be specified in several ways, including pixels, ems, and percentages.

For example:

```
p {
 font-size: 16px;
}
```

In the example above, the font size is set to 16 pixels.

Using ems allows you to specify the size relative to the parent element's font size. For example:

```
p {
 font-size: 1em;
}
```

In the example above, the font size is set to 1 times the size of the parent element's font size.

Percentages can also be used to specify the font size as a percentage of the parent element's font size. For example:

```
p {
 font-size: 100%;
}
```

In the example above, the font size is set to 100% of the parent element's font size.

It's important to note that different browsers have different default font sizes, so it's a good practice to specify the font size in your CSS to ensure that the text will be displayed consistently across all devices.

font-style

The **font-style** property in CSS specifies the style of the font. The possible values for this property are **normal**, **italic**, and **oblique**.

For example:

```
p {
```

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

In the example above, the font style is set to italic.

The **normal** value sets the font to a normal, non-italic style. The **italic** value sets the font to an italic style, and the **oblique** value sets the font to an oblique style, which is similar to italic but with less slant.

It's important to keep in mind that not all fonts have both italic and oblique styles, so it's a good practice to specify multiple fonts for the **font-family** property in case the desired style is not available.

font-weight

The **font-weight** property in CSS specifies the weight (or thickness) of the font. The possible values for this property range from **100** to **900**, with **400** being equivalent to the normal weight, and **700** being equivalent to bold.

For example:

```
p {
 font-weight: bold;
}
```

In the example above, the font weight is set to bold.

The values for **font-weight** can also be specified using keywords, such as **normal**, **bold**, **lighter**, and **bolder**.

For example:

```
p {
 font-weight: lighter;
}
```

In the example above, the font weight is set to lighter.

It's important to keep in mind that not all fonts have the same range of weights, so it's a good practice to specify multiple fonts for the **font-family** property in case the desired weight is not available.

color

The **color** property in CSS is used to specify the color of the text. The color can be specified in several ways, including using color names (e.g. **red**, **blue**, etc.), hexadecimal values (e.g. **#ff0000**

for red), RGB values (e.g. **rgb(255, 0, 0)** for red), and RGBA values (e.g. **rgba(255, 0, 0, 1)** for red).

For example:

```
p {
 color: blue;
}
```

In the example above, the text color is set to blue.

For example:

```
p {
 color: #ff0000;
}
```

In the example above, the text color is set to red.

It's important to keep in mind that different devices may display colors differently, so it's a good practice to specify the color in your CSS to ensure that the text will be displayed consistently across all devices.

text-align

The **text-align** property in CSS is used to specify the horizontal alignment of text within an element. The possible values for this property are **left**, **right**, **center**, and **justify**.

For example:

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

In the example above, the text within the **p** element is centered.

The **left** value sets the text to be aligned to the left, the **right** value sets the text to be aligned to the right, and the **center** value centers the text. The **justify** value spreads the text evenly across the entire width of the element.

It's important to note that the **text-align** property only affects the horizontal alignment of the text and does not affect the vertical alignment. To control the vertical alignment of text, you can use the **vertical-align** property.

text-decoration

The **text-decoration** property in CSS is used to add decorations to text. The possible values for this

property are **none**, **underline**, **overline**, **line-through**, and **blink**.

For example:

```
a {
 text-decoration: underline;
}
```

In the example above, the text within **a** elements (i.e. links) will be underlined.

The **none** value removes any existing text decorations, the **underline** value adds an underline to the text, the **overline** value adds an overline to the text, the **line-through** value adds a line through the text, and the **blink** value causes the text to blink.

It's important to keep in mind that the **blink** value is not widely supported by modern browsers and should be used with caution.

text-transform

The **text-transform** property in CSS is used to control the capitalization of text. The possible values for this property are **none**, **uppercase**, **lowercase**, and **capitalize**.

For example:

```
p {
 text-transform: uppercase;
}
```

In the example above, the text within the **p** element will be displayed in uppercase letters.

The **none** value leaves the text as is, the **uppercase** value converts all letters to uppercase, the **lowercase** value converts all letters to lowercase, and the **capitalize** value capitalizes the first letter of each word.

It's important to keep in mind that the **text-transform** property only affects the capitalization of the text and does not affect the font size, weight, or style. To control these aspects of the text, you can use the **font-size**, **font-weight**, and **font-style** properties.

line-height

The **line-height** property in CSS is used to specify the height of a line of text. The value for this property can be a length (e.g. **20px**), a percentage (e.g. **150%**), or a number (e.g. **1.5**).

For example:

```
p {
 line-height: 1.5;
}
```

In the example above, the line height of the text within the **p** element is set to **1.5**, meaning that the line height will be 1.5 times the size of the font.

The **line-height** property controls the amount of space above and below the text within an element. A larger line height will result in more space above and below the text, while a smaller line height will result in less space.

It's important to note that the value of the **line-height** property is not the actual height of the text, but rather the height of the line box that contains the text. The actual height of the text is determined by the font size, the font family, and other factors.

letter-spacing

The **letter-spacing** property in CSS is used to control the spacing between characters in a block of text. The value for this property can be a length (e.g. **2px**) or a percentage (e.g. **150%**).

For example:

```
p {
 letter-spacing: 2px;
}
```

In the example above, the spacing between characters within the **p** element is set to **2px**.

The **letter-spacing** property allows you to increase or decrease the space between characters in a block of text. A positive value for **letter-spacing** will increase the space between characters, while a negative value will decrease the space.

It's important to note that the **letter-spacing** property only affects the spacing between characters and does not affect the overall font size or weight. To control these aspects of the text, you can use the **font-size** and **font-weight** properties.

word-spacing

The **word-spacing** property in CSS is used to control the spacing between words in a block of text. The value for this property can be a length (e.g. **10px**) or a percentage (e.g. **150%**).

For example:

```
p {
```

```
 word-spacing: 10px;
}
```

In the example above, the spacing between words within the **p** element is set to **10px**.

The **word-spacing** property allows you to increase or decrease the space between words in a block of text. A positive value for **word-spacing** will increase the space between words, while a negative value will decrease the space.

It's important to note that the **word-spacing** property only affects the spacing between words and does not affect the overall font size or weight. To control these aspects of the text, you can use the **font-size** and **font-weight** properties.

text-indent

The **text-indent** property in CSS is used to control the indentation of the first line of a block of text. The value for this property can be a length (e.g. **20px**), a percentage (e.g. **10%**), or an absolute length (e.g. **5mm**).

For example:

```
p {
 text-indent: 20px;
}
```

In the example above, the first line of text within the **p** element will be indented **20px** from the left margin.

The **text-indent** property is commonly used to indent the first line of a paragraph, but it can be used on any block-level element. A positive value for **text-indent** will move the first line of text to the right, while a negative value will move it to the left.

It's important to note that the **text-indent** property only affects the first line of text in an element and does not affect subsequent lines. To control the indentation of subsequent lines, you can use the **padding** or **margin** properties.

text-shadow

The **text-shadow** property in CSS is used to add a shadow to text. The value for this property is a set of values that define the shadow's position, size, and color.

For example:

```
p {
```

```
 text-shadow: 2px 2px 3px gray;
}
```

In the example above, a gray shadow is added to the text within the **p** element. The shadow is positioned **2px** to the right and **2px** down from the text, and has a blur radius of **3px**.

The syntax for the **text-shadow** property is as follows:

```
text-shadow: [horizontal offset] [vertical offset] [blur radius] [color];
```

The horizontal and vertical offsets determine the position of the shadow relative to the text. A positive horizontal offset moves the shadow to the right, while a negative offset moves it to the left. A positive vertical offset moves the shadow down, while a negative offset moves it up.

The blur radius determines the spread of the shadow, with larger values resulting in a more blurred shadow and smaller values resulting in a more defined shadow.

The color value determines the color of the shadow. You can specify the color using any valid CSS color value, such as a named color (e.g. **gray**), a hexadecimal value (e.g. **#808080**), or an RGB value (e.g. **rgb(128, 128, 128)**).

It's important to note that the **text-shadow** property only affects the text within an element and does not affect the background or other elements on the page.

@font-face

The **@font-face** rule in CSS allows you to specify a custom font to be used in your web page, rather than relying on the user's device to have a particular font installed.

Here's an example of using **@font-face** to embed a font file:

```
@font-face {
 font-family: "My Custom Font";
 src: url("path/to/font.ttf") format("truetype");
}
```

In this example, you specify the **font-family** property with a unique name for the font. This name is used to reference the font in your CSS. The **src** property specifies the location and format of the font file. The format of the font file is specified in parentheses after the **url()** function. In this case, the font is in TrueType format (**.ttf**).

You can then use the custom font in your CSS like this:

```
p {
 font-family: "My Custom Font", sans-serif;
}
```

In this example, the **font-family** property is set to the custom font you specified in the **@font-face** rule, followed by a fallback font (in this case, **sans-serif**).

The **@font-face** rule works in most modern browsers and is a powerful way to add unique typography to your web page. However, it's important to keep in mind that not all browsers support all font formats, so you should specify multiple sources for different font formats if possible to ensure maximum compatibility.

font-variant

The **font-variant** property in CSS is used to specify the font variation of a text. This property is used to display small capital letters, or to turn off the display of special characters (such as ligatures) in a font.

There are two main values for the **font-variant** property: **normal** and **small-caps**.

- The **normal** value specifies that the text should be displayed in its normal variation.
- The **small-caps** value specifies that the text should be displayed in small capital letters.

Here's an example of using the **font-variant** property to display text in small capital letters:

```
p {
 font-variant: small-caps;
}
```

In this example, the text within the **p** element will be displayed in small capital letters.

It's worth noting that not all fonts support small capital letters, so using this property may result in unexpected behavior in some cases. Additionally, the appearance of small capital letters can vary greatly between fonts, so you may need to experiment to find a font that gives you the desired result.