
HTML & CSS
GO GET IT!
By Sachin Pisal

Note

You are free to use and distribute this book. You can alter the content, change the topics, and remove my name. Everything is fine as long as you are giving it for free.

A lot of thanks to ChatGPT, my family, and my coaches for helping and motivating me to create this e-book.I don’t take any guarantee or responsibility for any damages done to your products using the content from this book.

HTML & CSS EBook

---

1. HTML Fundamentals

---

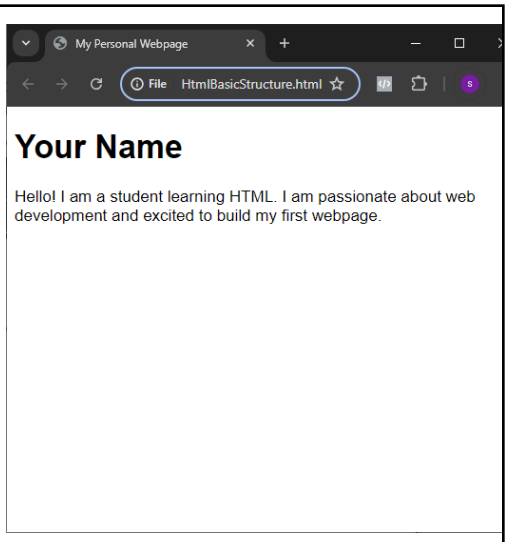
1. Basic structure of an HTML document

Scenario: Imagine you are a student who wants to create a personal webpage to introduce yourself to potential employers. Your task is to create a simple HTML page with the following requirements:

1. **Create the basic structure** of an HTML document.
2. **Add a title** for the webpage that will appear in the browser tab.
3. **Include a heading** (e.g., your name) and a paragraph that describes who you are and what you do.

<table>
<tr>
<td>

```
<!DOCTYPE html>
<html>
<head>
    <title>My Personal
Webpage</title>
</head>
<body>
    <h1>Your Name</h1>
    <p>Hello! I am a student learning
HTML. I am passionate about web
development and excited to build my
first webpage.</p>
</body>
</html>
```

</td>
<td> 
</td>
</tr>
<tr>
<td>Your Task</td>
<td>Display your name and set the heading to h2 style</td>
</tr>
</table>

By following these steps, you will have created a basic HTML document that meets the requirements and introduces you to the fundamental structure of HTML.

2. Block-level and inline elements

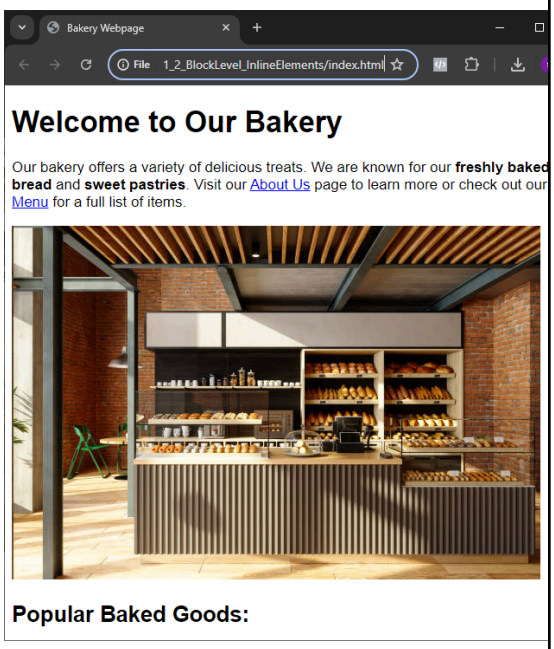
Scenario: Design a simple webpage for a small local bakery. The owner wants the webpage to have a heading, a description, an image of the bakery, and a list of popular

baked goods. Additionally, the owner wants certain text to be bolded and some links to be placed within the description.

Task:

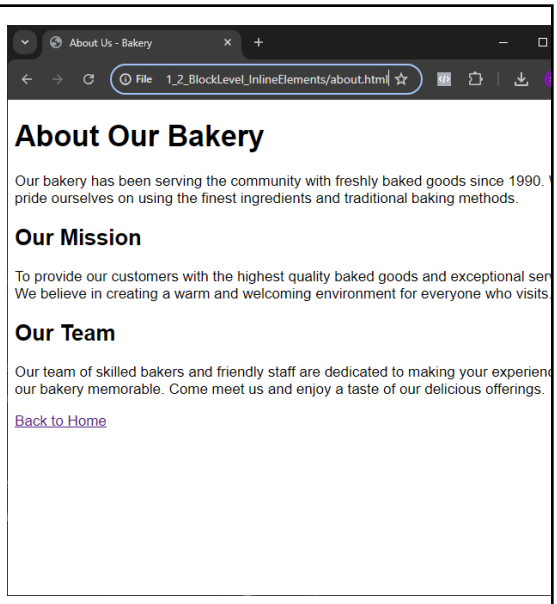
- Create an HTML document with the following requirements:
  - A heading that says "Welcome to Our Bakery".
  - A paragraph describing the bakery.
  - An image of the bakery.
  - A list of popular baked goods.
  - Certain words in the description should be bolded.
  - Include two links within the description.
- Identify and use appropriate block-level and inline elements to achieve the desired layout.

```
<!DOCTYPE html>
<html>
<head>
   <title>Bakery Webpage</title>
   <meta charset="UTF-8">
</head>
<body>
   <h1>Welcome to Our Bakery</h1>
   <p>
      Our bakery offers a variety of
delicious treats. We are known for our
      <strong>freshly baked
bread</strong> and <strong>sweet
pastries</strong>.
      Visit our <a
href="about.html">About Us</a> page to
learn more or
      check out our <a
href="menu.html">Menu</a> for a full list
of items.
   </p>
   <img src="bakery.jpg" alt="Image of
our bakery">
   <h2>Popular Baked Goods:</h2>
   <ul>
      <li>Bread</li>
      <li>Croissants</li>
      <li>Muffins</li>
      <li>Cakes</li>
   </ul>
</body>
</html>
```



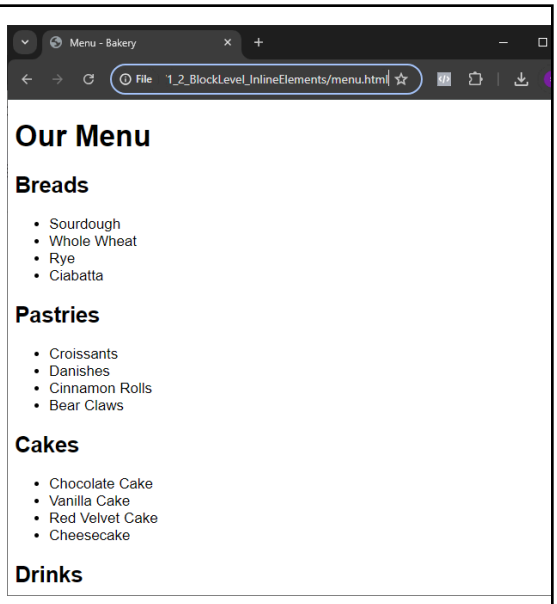
about.html

```
<!DOCTYPE html>
<html>
<head>
    <title>About Us - Bakery</title>
    <meta charset="UTF-8">
</head>
<body>
    <h1>About Our Bakery</h1>
    <p>
        Our bakery has been serving the community with freshly baked goods since 1990.
        We pride ourselves on using the finest ingredients and traditional baking methods.
    </p>
    <h2>Our Mission</h2>
    <p>
        To provide our customers with the highest quality baked goods and exceptional service.
        We believe in creating a warm and welcoming environment for everyone who visits.
    </p>
    <h2>Our Team</h2>
    <p>
        Our team of skilled bakers and friendly staff are dedicated to making your experience at our bakery memorable. Come meet us and enjoy a taste of our delicious offerings.
    </p>
    <a href="index.html">Back to Home</a>
</body>
</html>
```



menu.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Menu - Bakery</title>
    <meta charset="UTF-8">
</head>
<body>
    <h1>Our Menu</h1>
    <h2>Breads</h2>
    <ul>
        <li>Sourdough</li>
        <li>Whole Wheat</li>
        <li>Rye</li>
        <li>Ciabatta</li>
    </ul>
    <h2>Pastries</h2>
    <ul>
        <li>Croissants</li>
        <li>Danishes</li>
        <li>Cinnamon Rolls</li>
        <li>Bear Claws</li>
    </ul>
    <h2>Cakes</h2>
    <ul>
        <li>Chocolate Cake</li>
        <li>Vanilla Cake</li>
        <li>Red Velvet Cake</li>
        <li>Cheesecake</li>
    </ul>
    <h2>Drinks</h2>
    <ul>
        <li>Coffee</li>
        <li>Tea</li>
        <li>Hot Chocolate</li>
        <li>Fresh Juice</li>
    </ul>
    <a href="index.html">Back to
Home</a>
</body>
</html>
```



Your Task

- Add image for bakery.jpg
- Add a page “Contact Us” and provide a bakery address.
- Provide link to this page on index page

3. Basic HTML *entities*.

HTML entities are used to display reserved characters, special symbols, or non-printable characters in HTML. Here are some of the most commonly used basic HTML entities:

1. Reserved Characters:
    - & - Ampersand (&)
    - < - Less-than (<)
    - > - Greater-than (>)
    - " - Double quote (")
    - ' - Single quote (')
2. Special Symbols:
    - © - Copyright (©)
    - ® - Registered trademark (®)
    - ™ - Trademark (™)
    - ¢ - Cent (¢)
    - £ - Pound sterling (£)
    - ¥ - Yen (¥)
    - € - Euro (€)
3. Mathematical Symbols:
    - ± - Plus-minus (±)
    - × - Multiplication (×)
    - ÷ - Division (÷)
    - = - Equals (=)
4. Non-breaking Space:
    -   - Non-breaking space (a space that prevents line breaks at its position)
5. Miscellaneous:
    - § - Section symbol (§)
    - ¶ - Paragraph symbol (¶)
    - ° - Degree (°)

These entities ensure that characters are displayed correctly in HTML documents, particularly those that have special meanings in HTML syntax.

Scenario: Develop a webpage for a non-profit organization. The webpage needs to display special characters and symbols correctly, such as the copyright symbol, the greater-than and less-than signs, and the non-breaking space. The team also wants to ensure that any text containing reserved HTML characters is displayed correctly on the webpage.

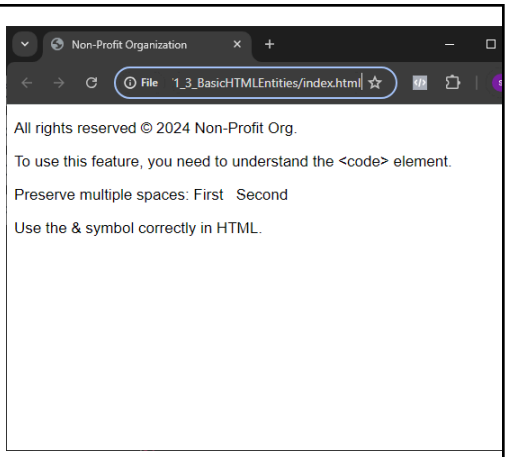
Task:

- Create an HTML document that includes:
    - A paragraph with the text: "All rights reserved © 2024 Non-Profit Org."

    - Another paragraph that uses the greater-than (>) and less-than (<) symbols correctly.
    - A line of text where multiple spaces are preserved.
    - A block of text that uses the ampersand (&) symbol correctly.
- Identify and use appropriate HTML entities to ensure these characters display as intended.

<table>
<tr>
<td>

```
<!DOCTYPE html>
<html>
<head>
    <title>Non-Profit Organization</title>
    <meta charset="UTF-8">
</head>
<body>
    <p>All rights reserved &copy; 2024 Non-Profit Org.</p>
    <p>To use this feature, you need to understand the &lt;code&gt; element.</p>
    <p>Preserve multiple spaces: First   Second</p>
    <p>Use the &amp; symbol correctly in HTML.</p>
</body>
</html>
```

</td>
<td> 
</td>
</tr>
<tr>
<td>Your Task</td>
<td>Add below sentence to above page: John said, "I paid €50 for this book, which is about <HTML> tags."</td>
</tr>
</table>

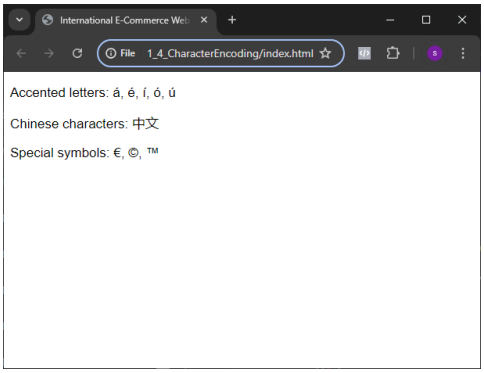
4. Character encoding

Scenario:

You are working on an international e-commerce website that caters to users from various countries, each with different languages and character sets. Some users have reported that certain characters, such as accented letters, Chinese characters, and special symbols, are not displaying correctly on the website. To fix this, you need to ensure that the HTML document is correctly set up with the appropriate character encoding.

Task:

- Create an HTML document that includes:
    - A meta tag to specify the correct character encoding.
    - A paragraph containing accented letters (e.g., á, é, í, ó, ú).
    - A paragraph containing Chinese characters (e.g., 中文).
    - A paragraph containing special symbols (e.g., €, ©, ™).
- Ensure that all characters display correctly by using the appropriate character encoding.

<table>
<tr>
<td>index.html
<!DOCTYPE html>
<html>
<head>
<title>International E-Commerce Website</title>
<meta charset="UTF-8">
</head>
<body>
<p>Accented letters: á, é, í, ó, ú</p>
<p>Chinese characters: 中文</p>
<p>Special symbols: €, ©, ™</p>
</body>
</html></td>
<td> 
</td>
</tr>
<tr>
<td>Your Task</td>
<td>Change encoding to charset="ISO-8859-1" and see the result.</td>
</tr>
</table>

5. Comments as HTML code for documentation purpose

Comments are useful for documenting your code, making notes to yourself or other developers, or temporarily disabling parts of the code without removing them entirely. Here's how you can use comments in HTML and a problem-based sample:

Syntax for HTML Comments:

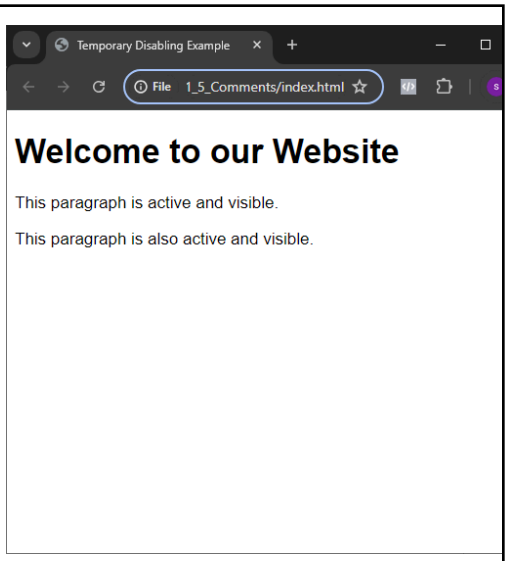
HTML comments are enclosed within <!-- and --> tags. Anything between these tags is treated as a comment and is ignored by the browser when rendering the webpage.

Problem:

You want to temporarily disable a section of HTML code without deleting it, so you can easily re-enable it later.

<table>
<tr>
<td>

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Temporary Disabling
Example</title>
</head>
<body>
    <h1>Welcome to our
Website</h1>

    <p>This paragraph is active and
visible.</p>

    <!--
    <div>
        <p>This content is temporarily
disabled.</p>
    </div>
    -->

    <p>This paragraph is also active
and visible.</p>
</body>
</html>
```

</td>
<td> 
</td>
</tr>
<tr>
<td>Your Task</td>
<td>Enable the code</td>
</tr>
</table>

2. Formatting & Structuring

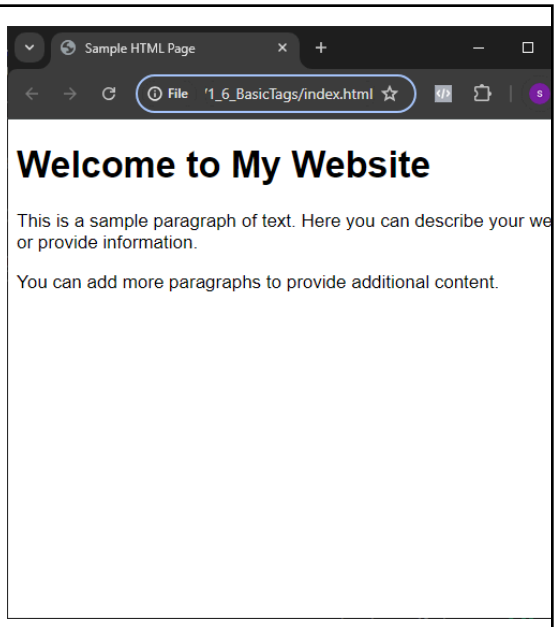
---

1. Use of basic tags (html, head, title, body)

Scenario:
Create a simple HTML page that includes a title in the browser tab and displays a heading and a paragraph of text in the main body.

index.html

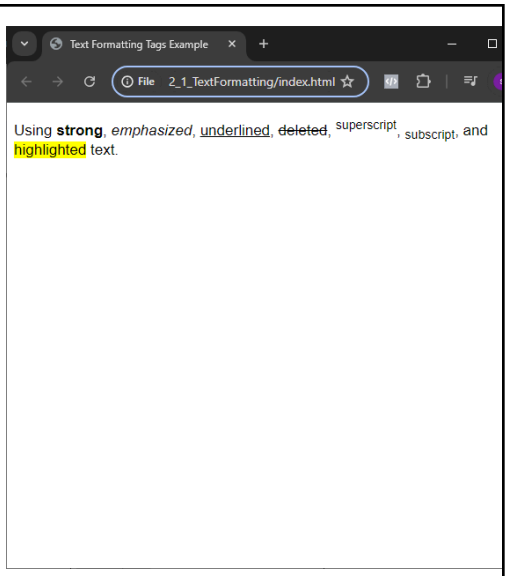
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Sample HTML Page</title>
</head>
<body>
    <h1>Welcome to My Website</h1>

    <p>This is a sample paragraph of text. Here you can describe your website or provide information.</p>

    <p>You can add more paragraphs to provide additional content.</p>
</body>
</html>
```



Your Task	**Change the title of the page, heading and add a new 2 line paragraph.**

2. Text formatting tags

Use of tags such as strong, em, u, del, sup, sub, and mark.

index.html <!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <title>Text Formatting Tags Example</title> </head> <body> Using strong, emphasized, <u>underlined</u>, ~~deleted~~, ^{superscript}, _{subscript}, and <mark>highlighted</mark> text. </body> </html>	
Your Task	Manually Type and create each HTML file. No copy, God is watching!

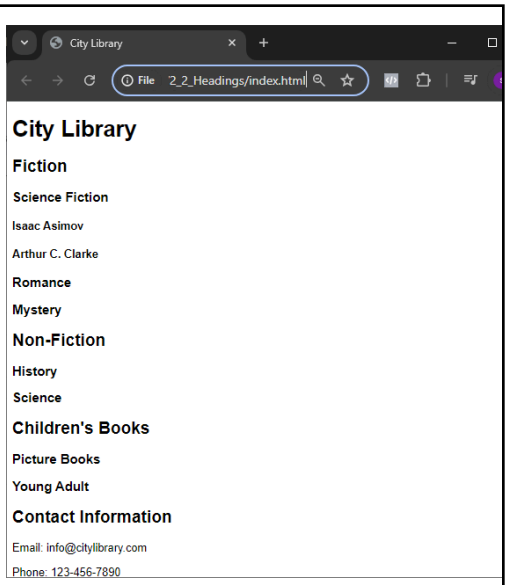
3. Headings

Headings in HTML are used to define the structure and hierarchy of the content on a webpage. They range from <h1> to <h6>, with <h1> being the most important heading and <h6> being the least important.

Scenario: Let's consider a scenario where you need to create a webpage for a local library. The library wants a page that provides information about different sections such as Fiction, Non-Fiction, Children's Books, and Contact Information. We'll use headings to structure this content effectively.

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>City Library</title>
</head>
<body>
    <h1>City Library</h1>

    <h2>Fiction</h2>
    <h3>Science Fiction</h3>
    <h4>Isaac Asimov</h4>
    <h4>Arthur C. Clarke</h4>
    <h3>Romance</h3>
    <h3>Mystery</h3>

    <h2>Non-Fiction</h2>
    <h3>History</h3>
    <h3>Science</h3>

    <h2>Children's Books</h2>
    <h3>Picture Books</h3>
    <h3>Young Adult</h3>

    <h2>Contact Information</h2>
    <p>Email:
info@citylibrary.com</p>
    <p>Phone: 123-456-7890</p>
</body>
</html>
```



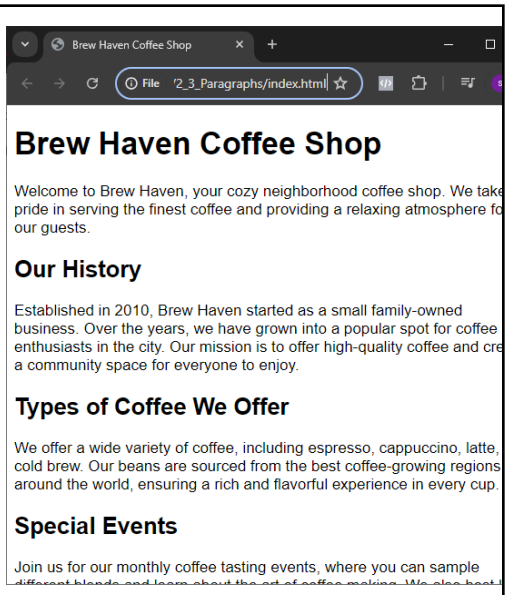
Your Task	Add category “Programming” - Add two books, “Head First Javascript” and “Javascript -Good Parts”

4. Paragraphs

Paragraphs in HTML are defined using the <p> tag. They are used to break content into readable blocks, enhancing the text flow and readability of the webpage.

Scenario: Imagine you need to create a webpage for a local coffee shop. The webpage should provide information about the shop's history, the types of coffee they offer, and details about their special events.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Brew Haven Coffee
Shop</title>
</head>
<body>
    <h1>Brew Haven Coffee
Shop</h1>

    <p>Welcome to Brew Haven, your
cozy neighborhood coffee shop. We
take pride in serving the finest coffee
and providing a relaxing atmosphere
for all our guests.</p>

    <h2>Our History</h2>
    <p>Established in 2010, Brew
Haven started as a small
family-owned business. Over the
years, we have grown into a popular
spot for coffee enthusiasts in the city.
Our mission is to offer high-quality
coffee and create a community
space for everyone to enjoy.</p>

    <h2>Types of Coffee We
Offer</h2>
    <p>We offer a wide variety of
coffee, including espresso,
cappuccino, latte, and cold brew. Our
beans are sourced from the best
coffee-growing regions around the
world, ensuring a rich and flavorful
experience in every cup.</p>

    <h2>Special Events</h2>
    <p>Join us for our monthly coffee
tasting events, where you can
sample different blends and learn
about the art of coffee making. We
```



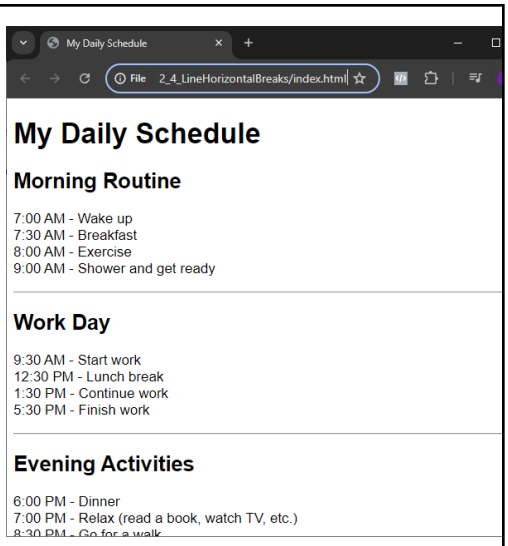
also host live music nights every Friday, featuring local artists and bands.</p> </body> </html>	
Your Task	Add one more paragraph

5. Content separation using Line breaks (br) and horizontal rules (hr)

Line breaks (
) and horizontal rules (<hr>) are useful HTML elements for controlling content separation. Line breaks are used to insert a new line without starting a new paragraph, while horizontal rules are used to create a thematic break between content sections.

Scenario: Create a simple example of a webpage that provides information about a daily schedule.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>My Daily Schedule</title>
</head>
<body>
    <h1>My Daily Schedule</h1>

    <h2>Morning Routine</h2>
    <p>
        7:00 AM - Wake up<br>
        7:30 AM - Breakfast<br>
        8:00 AM - Exercise<br>
        9:00 AM - Shower and get
ready
    </p>
    <hr>

    <h2>Work Day</h2>
    <p>
        9:30 AM - Start work<br>
        12:30 PM - Lunch break<br>
```



1:30 PM - Continue work 5:30 PM - Finish work <hr> Evening Activities 6:00 PM - Dinner 7:00 PM - Relax (read a book, watch TV, etc.) 8:30 PM - Go for a walk 9:30 PM - Prepare for bed </body> </html>	
Your Task	Add your three hobbies at the end

6. Use the blockquote, q, cite, and abbr tags

Blockquote and Cite Tags:

- <blockquote>: Defines a section that is quoted from another source.
- <q>: Denotes a short inline quotation.
- <cite>: Specifies the title of the work being quoted.

Abbreviation Tag:

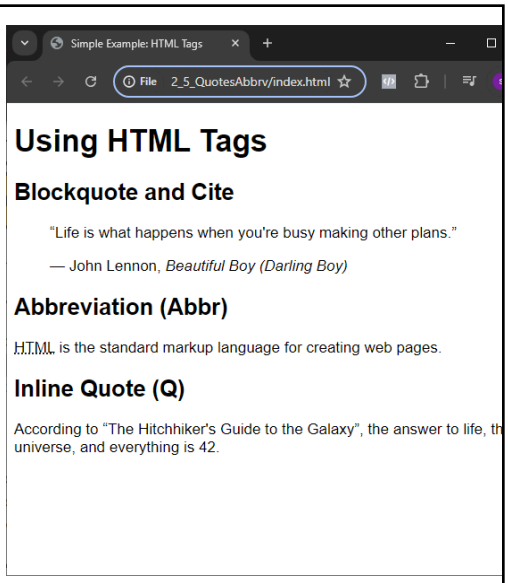
- <abbr>: Denotes an abbreviation or acronym, with the full term provided in the title attribute.

Inline Quote (Q Tag):

- <q>: Used to enclose inline quotations.

  Scenario : Create a simple quote example.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Simple Example: HTML
Tags</title>
</head>
<body>
    <h1>Using HTML Tags</h1>

    <h2>Blockquote and Cite</h2>
    <blockquote>
        <p><q>Life is what happens
when you're busy making other
plans.</q></p>
        <footer>— John Lennon,
<cite>Beautiful Boy (Darling
Boy)</cite></footer>
    </blockquote>

    <h2>Abbreviation (Abbr)</h2>
    <p>
        <abbr title="HyperText Markup
Language">HTML</abbr> is the
standard markup language for
creating web pages.
    </p>

    <h2>Inline Quote (Q)</h2>
    <p>
        According to <q>The
Hitchhiker's Guide to the Galaxy</q>,
the answer to life, the universe, and
everything is 42.
    </p>
</body>
</html>
```



Your task

Add below text:
"Be yourself; everyone else is already taken."

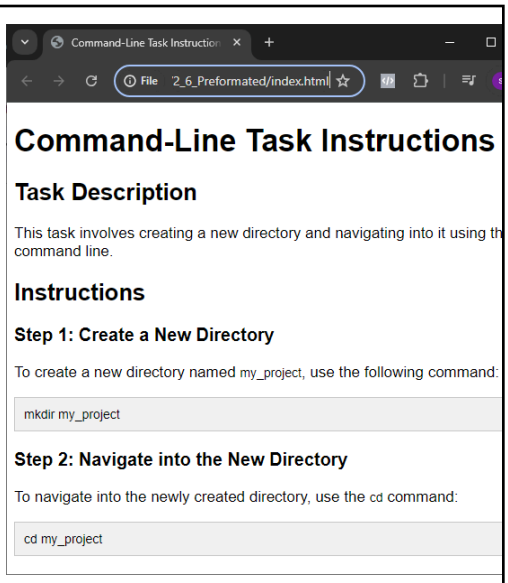
— Oscar Wilde, *The Happy Prince and Other Tales*

7. Preformatted text, format code snippets and user input.

Employ code, pre, kbd, and samp tags to display preformatted text, format code snippets and user input

Scenario: Create instruction for commandline, and display user input in a clear and formatted manner.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Command-Line Task
Instructions</title>
    <style>
        pre {
            background-color: #f0f0f0;
            padding: 10px;
            border: 1px solid #ccc;
            white-space: pre-wrap;
        }
    </style>
</head>
<body>
    <h1>Command-Line Task
Instructions</h1>

    <h2>Task Description</h2>
    <p>
        This task involves creating a
new directory and navigating into it
using the command line.
    </p>

    <h2>Instructions</h2>

    <h3>Step 1: Create a New
Directory</h3>
    <p>To create a new directory
named <code>my_project</code>,
use the following command:</p>
    <pre><code>mkdir
my_project</code></pre>

    <h3>Step 2: Navigate into the
```

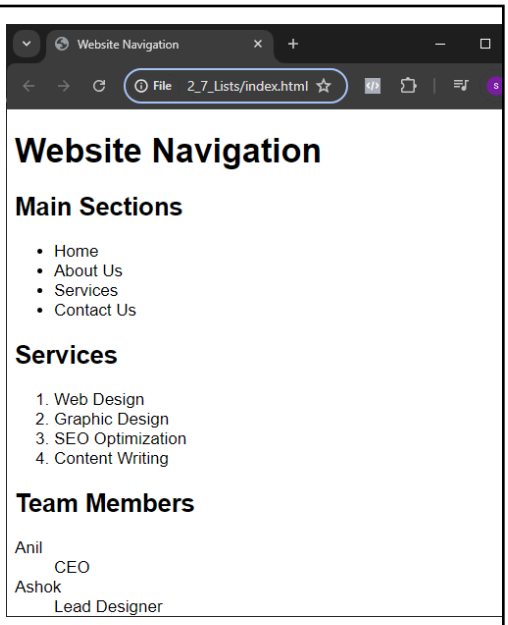


New Directory</h3> To navigate into the newly created directory, use the <kbd>cd</kbd> command: <pre><code>cd my_project</code></pre> <h3>Step 3: Verify Current Directory</h3> To verify that you are in the correct directory, use the <samp>pwd</samp> command: <pre><code>pwd</code></pre> Congratulations! You have successfully completed the task. </body> </html>	
Your Task	Add command to delete a directory

8. Ordered (ol), unordered (ul), and definition lists (dl), nesting.

Scenario : Create a website with different sections. Create ordered, unordered and nested lists for the services you provide along with team information.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Website Navigation</title>
</head>
<body>
    <h1>Website Navigation</h1>

    <h2>Main Sections</h2>
    <ul>
        <li>Home</li>
        <li>About Us</li>
        <li>Services</li>
        <li>Contact Us</li>
```



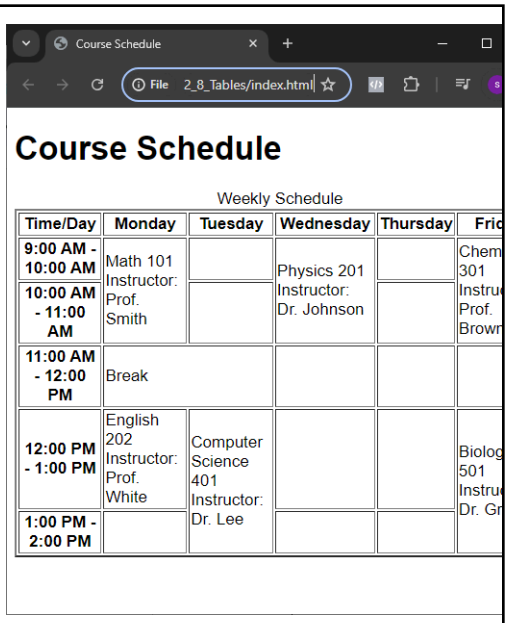
   Services - Web Design - Graphic Design - SEO Optimization - Content Writing Team Members <dl> <dt>Anil</dt> <dd>CEO</dd> <dt>Ashok</dt> <dd>Lead Designer</dd> <dt>Ashish</dt> <dd>Marketing Manager</dd> </dl> Projects - Project A - Project B - Project C - Sub-project 1 - Sub-project 2 - Project D </body> </html>	
Your Task	Add your name as CTO

9. Create and manipulate tables

Use (*table*, *tr*, *td*), focusing on headers (*th*), cell merging (*colspan*, *rowspan*), captions (*caption*), and table attributes (e.g., summary).

Scenario: Imagine you are designing a webpage for a university course schedule. You want to create a table that displays the schedule for a week, including different courses, instructors, and timings.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Course Schedule</title>
</head>
<body>
    <h1>Course Schedule</h1>

    <table summary="Weekly Course
Schedule" border="1">
        <caption>Weekly
Schedule</caption>
        <thead>
            <tr>
                <th>Time/Day</th>
                <th>Monday</th>
                <th>Tuesday</th>
                <th>Wednesday</th>
                <th>Thursday</th>
                <th>Friday</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <th>9:00 AM - 10:00
AM</th>
                <td rowspan="2">Math
101<br>Instructor: Prof. Smith</td>
                <td></td>
                <td rowspan="2">Physics
201<br>Instructor: Dr. Johnson</td>
                <td></td>
                <td
rowspan="2">Chemistry
301<br>Instructor: Prof. Brown</td>
            </tr>
            <tr>
                <th>10:00 AM - 11:00
AM</th>
                <td></td>
                <td></td>
            </tr>
            <tr>
```



<th>11:00 AM - 12:00 PM</th> <td colspan="2">Break</td> <td></td> <td></td> <td></td> </tr> <tr> <th>12:00 PM - 1:00 PM</th> <td>English 202 Instructor: Prof. White</td> <td rowspan="2">Computer Science 401 Instructor: Dr. Lee</td> <td></td> <td></td> <td rowspan="2">Biology 501 Instructor: Dr. Green</td> </tr> <tr> <th>1:00 PM - 2:00 PM</th> <td></td> <td></td> <td></td> </tr> </tbody> </table> </body> </html>	
Your Task	• Add Saturday 9:00 to 11:00 AM, Lab Practicals by Mr. Grey • Change the value of “border=0”, see what happens

Explanation

1. **Table (<table>)**: The border="1" attribute is added to display table borders.

   Table Headers (<thead>, <th>): Defines the header row of the table.

   Table Cells (<td>): Contains data cells with course names, instructors, and merged cells using rowspan and colspan attributes.

3. Multimedia and Hyperlinks

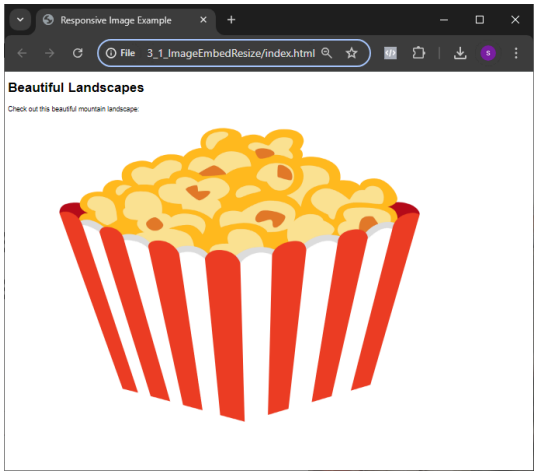
---

1. Embedding images for responsive design

Scenario:

Imagine you have a website showcasing various landscapes, and you want to ensure that the images are accessible, load efficiently, and adapt to different screen sizes.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Simple Responsive Images
Example</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
            background-color: #f0f0f0;
            padding: 20px;
            margin: 0;
        }
        .responsive-image {
            max-width: 100%;
            height: auto;
        }
    </style>
</head>
<body>
    <h1>Simple Responsive Images
Example</h1>
    <p>This example demonstrates how
to embed images for responsive design
using HTML and CSS.</p>

    <!-- Simple Responsive Image -->
```



<img src="https://via.placeholder.com/1024" alt="Example Image" class="responsive-image"> The image above adapts to different screen sizes, providing a responsive design. </body> </html>	

Explanation:

1. **src Attribute:**
    - Specifies the URL of the image. This is the fallback image that will be displayed if none of the images in the srcset match the conditions.
2. **alt Attribute:**
    - Provides alternative text for the image. This is crucial for accessibility, allowing screen readers to describe the image to visually impaired users. It also displays when the image fails to load.
3. **srcset Attribute:**
    - Contains a list of image URLs along with their respective widths. This helps the browser choose the best image to load based on the device's screen size and resolution.
    - Example: images/mountain-small.jpg 480w means that mountain-small.jpg is 480 pixels wide.
4. **sizes Attribute:**
    - Specifies the sizes of the image slots for different viewport widths. This guides the browser in selecting the appropriate image from the srcset.
    - Example: (max-width: 600px) 480px means that if the viewport width is 600 pixels or less, the image will take up 480 pixels.
5. **width and height Attributes:**
    - Define the intrinsic dimensions of the image. These help the browser allocate space for the image before it is fully loaded, preventing layout shifts.

How it Works:

- On a small screen (up to 600px wide), the browser will use the 480px wide image.
- On a medium screen (up to 900px wide), the browser will use the 800px wide image.

- On a large screen (more than 900px wide), the browser will use the 1200px wide image.

This setup ensures that the image is responsive, loading an appropriate version based on the device's screen size and resolution, thus optimizing loading times and enhancing the user experience.

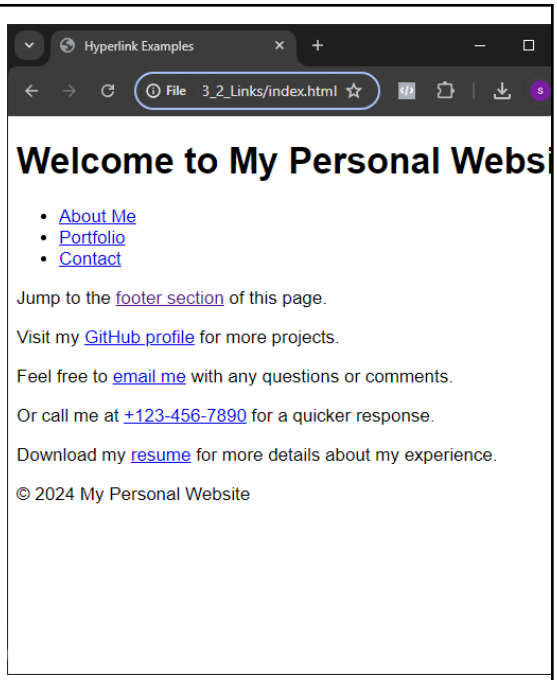
2. Create hyperlinks

Using the *a* tag, covering internal and anchor links, external links, email and telephone links, and link-specific attributes (*href*, *target*, *download*, *rel*, *title*).

Scenario:

Create a personal website, add you information, portfolio, contact information along with profile, email and phone hyperlinks

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Hyperlink Examples</title>
</head>
<body>
    <h1>Welcome to My Personal
Website</h1>

    <!-- Internal Link -->
    <nav>
        <ul>
            <li><a href="about.html"
title="Learn more about me">About
Me</a></li>
            <li><a href="portfolio.html"
title="Check out my
work">Portfolio</a></li>
            <li><a href="contact.html"
title="Get in touch with
me">Contact</a></li>
        </ul>
    </nav>

    <!-- Anchor Link -->
```



Jump to the footer section of this page. <!-- External Link --> Visit my GitHub profile for more projects. <!-- Email Link --> Feel free to email me with any questions or comments. <!-- Telephone Link --> Or call me at +123-456-7890 for a quicker response. <!-- Download Link --> Download my resume for more details about my experience. <!-- Footer Section for Anchor Link --> <footer id="footer"> © 2024 My Personal Website </footer> </body> </html>	
Your Task	Create missing pages Create resume.pdf

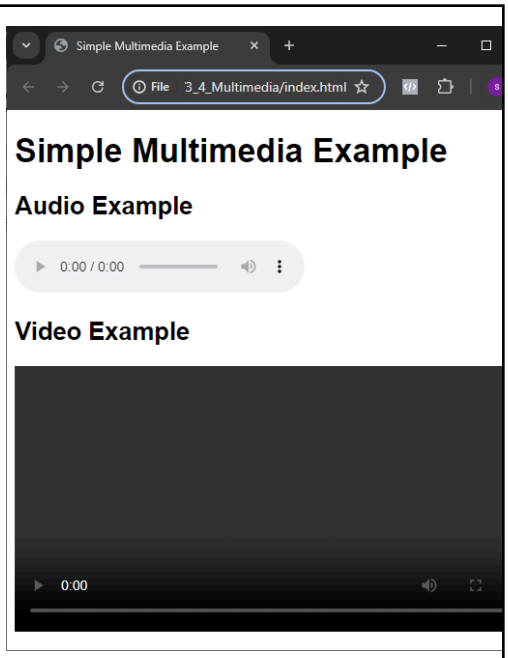
3. Embed multimedia

Content using *audio* and *video* elements (*src*, *controls*, *width*, *height*, *autoplay*, *loop*, *preload*), including considerations for cross-browser compatibility (the *source* element) and responsiveness (*poster*, *media*).

Scenario: Imagine you are creating a webpage for a music band and want to showcase their latest audio track and a promotional video.

<table>
<tr>
<td>

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Simple Multimedia
Example</title>
</head>
<body>
    <h1>Simple Multimedia
Example</h1>

    <section>
        <h2>Audio Example</h2>
        <audio controls>
            <source
src="happybirthday.mp4"
type="audio/mpeg">
            Your browser does not
support the audio element.
        </audio>
    </section>

    <section>
        <h2>Video Example</h2>
        <video controls width="100%">
            <source src="sample.mp4"
type="video/mp4">
            Your browser does not
support the video element.
        </video>
    </section>
</body>
</html>
```

</td>
<td> 
</td>
</tr>
<tr>
<td>Your Task</td>
<td>The video file is not present. Adjust the src URL to (https://www.example.com/video/sample.mp4) to point to actual video files you want to embed.</td>
</tr>
</table>

The <audio> element embeds an audio file with basic controls (controls attribute) for playback.

The <video> element embeds a video file with basic controls (controls attribute) and adjusts its width to 100% of the container's width for responsiveness.

Both elements include a <source> element to specify the source file (src) and its MIME type (type), ensuring compatibility across different browsers.

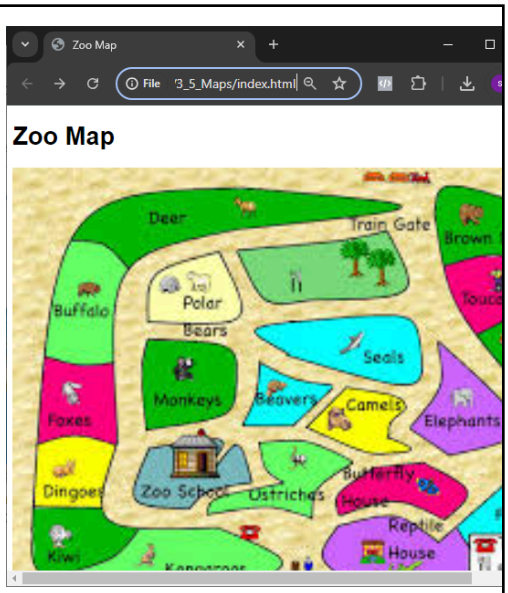
4. Maps

Design interactive image maps with map and area tags.

Scenario: Imagine you are designing a webpage for a zoo map where users can click on different areas representing animal enclosures to get more information about the animals.

<table>
<tr>
<td>

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Zoo Map</title>
</head>
<body>
    <h1>Zoo Map</h1>

    <img src="zoo-map.jpeg" alt="Zoo
Map" usemap="#zoomap"
width="800" height="600" />

    <map name="zoomap">
        <area shape="rect"
coords="50,50,250,250" alt="Lion
Enclosure" href="lion.html">
        <area shape="circle"
coords="400,200,100" alt="Elephant
Enclosure" href="elephant.html">
        <area shape="poly"
coords="600,100,700,150,650,250"
alt="Monkey Enclosure"
href="monkey.html">
    </map>
</body>
</html>
```

</td>
<td>


</td>
</tr>
<tr>
<td>Your Task</td>
<td>Correct for lion and monkey. Add</td>
</tr>
</table>

	respective page on click event

Explanation

- **Image (<img>)**:
    - Displays the zoo map image (zoo-map.png) with specified dimensions (width="800" and height="600").
    - Uses usemap="#zoomap" attribute to associate with the <map> element named "zoomap".

Map (<map>):

- Defines the image map named "zoomap" that contains clickable areas (<area> tags).
- name="zoomap" associates the <map> with the <img> element via usemap="#zoomap".

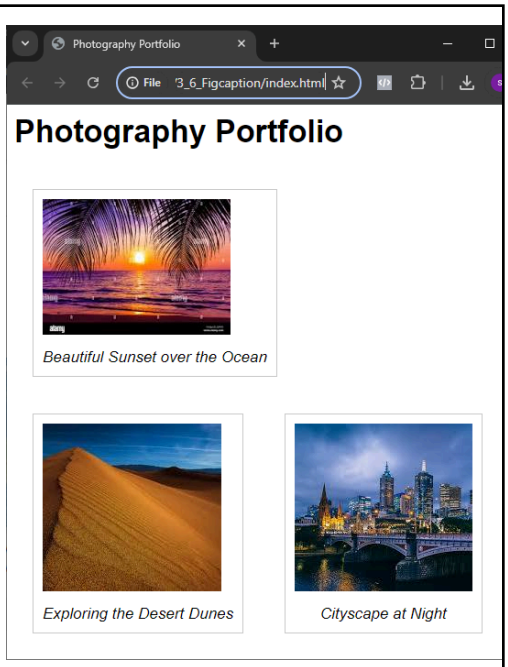
Areas (<area>):

- Defines clickable areas on the image (shape, coords, alt, href attributes).
- shape="rect" specifies rectangular shape; coords define coordinates (left, top, right, bottom).
- shape="circle" specifies circular shape; coords define center (x, y) and radius.
- shape="poly" specifies polygonal shape; coords define vertices (x, y pairs).
- alt provides alternative text for accessibility; href links to different pages for each animal enclosure.

5. figure and figcaption.

Scenario : Imagine you are designing a webpage for a photography portfolio where each image is accompanied by a caption describing the scene or location captured in the photo.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Photography Portfolio</title>
    <style>
        /* Optional: CSS for styling the
figure and figcaption */
        figure {
            margin: 20px;
            border: 1px solid #ccc;
            padding: 10px;
            display: inline-block;
        }
        figcaption {
            text-align: center;
            font-style: italic;
            margin-top: 10px;
        }
    </style>
</head>
<body>
    <h1>Photography Portfolio</h1>

    <figure>
        <img src="photo1.jpeg"
alt="Photo 1">
        <figcaption>Beautiful Sunset
over the Ocean</figcaption>
    </figure>

    <figure>
        <img src="photo2.jpeg"
alt="Photo 2">
        <figcaption>Exploring the
Desert Dunes</figcaption>
    </figure>

    <figure>
        <img src="photo3.jpeg"
alt="Photo 3">
        <figcaption>Cityscape at
Night</figcaption>
    </figure>
</body>
</html>
```



Your Task	Add beautiful beach photo

Figure (<figure>):

- Represents a standalone image with an optional caption (<figcaption>).
- Contains an <img> element as its content.

Image (<img>):

- Displays the photo (photo1.jpeg) with specified alt attribute for accessibility.

Figcaption (<figcaption>):

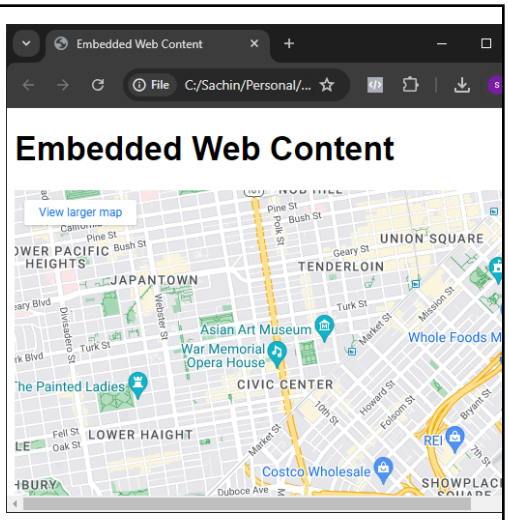
- Provides a caption for the image (Beautiful Sunset over the Ocean).

6. Embed external web content

Embed external web content with iframe, including making iframes responsive.

Scenario: Imagine you are creating a webpage that includes a section with a responsive iframe displaying a Google Maps location.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Embedded Web
Content</title>
</head>
<body>
    <h1>Embedded Web
Content</h1>

    <iframe
src="https://www.google.com/maps/e
mbed?pb=!1m18!1m12!1m3!1d1218
9.694869225707!2d-122.417937098
5738!3d37.77563540584937!2m3!1f
0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3
m3!1m2!1s0x80858070f27b9a9d%3
A0xd80938e0f057882!2sGolden%20
Gate%20Bridge!5e0!3m2!1sen!2sus!
```



4v1624906173835!5m2!1sen!2sus" width="600" height="450" frameborder="0" style="border:0;" allowfullscreen="" aria-hidden="false" tabindex="0"></iframe> Embedding Google Maps location of the Golden Gate Bridge in San Francisco. </body> </html>	
Your Task	Try to show your city

HTML Structure:

- The <iframe> element is used to embed external web content.
- The src attribute specifies the URL of the external content to be embedded (in this case, a Google Maps location).

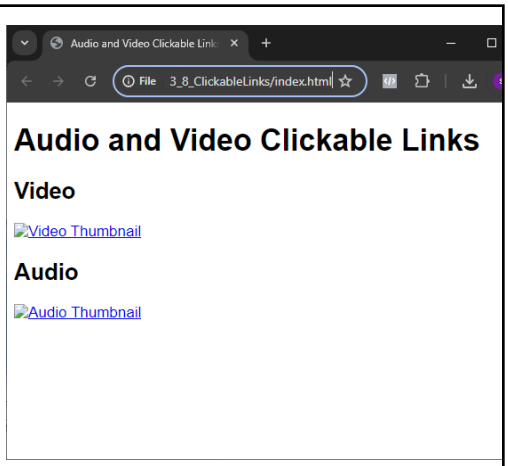
External Content (Google Maps):

- The <iframe> embeds a Google Maps location of the Golden Gate Bridge in San Francisco.
- Adjust the src attribute URL to embed different content as needed

7. clickable links

Integrate multimedia elements as clickable links.

Scenario: Integrate a single audio and video element as clickable links in HTML:

<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Audio and Video Clickable Links</title> </head> <body> Audio and Video Clickable Links Video <img src="video-thumbnail.jpg" alt="Video Thumbnail" width="300" height="200"> Audio <img src="audio-thumbnail.jpg" alt="Audio Thumbnail" width="300" height="200"> </body> </html>	
Your Task	Thumbnails and audio, video files you need to add.

HTML Structure:

- Uses <a> (anchor) tags to create clickable links.
- Each <a> tag surrounds an <img> tag, creating an image that serves as a thumbnail for the multimedia content.

Video:

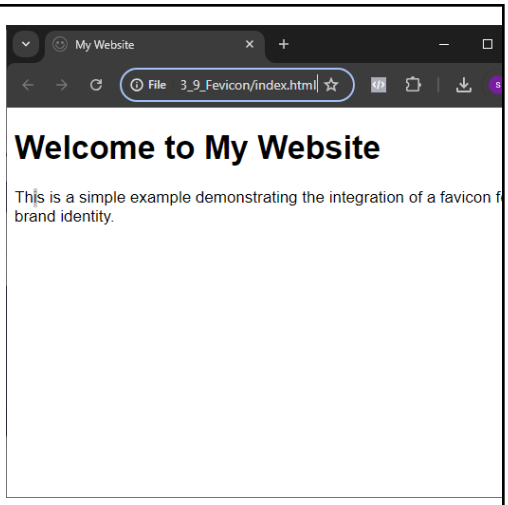
- Clicking on the video thumbnail (<img> tag) opens the video file (example-video.mp4) in a new tab (target="_blank").

Audio:

- Similarly, clicking on the audio thumbnail (<img> tag) opens the audio file (example-audio.mp3) in a new tab.

8. favicons

Scenario: Create your website and add a favicon

<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>My Website</title> <!-- Replace 'favicon.ico' with your actual favicon file --> <link rel="shortcut icon" type="image/x-icon" href="favicon.ico"> </head> <body> Welcome to My Website This is a simple example demonstrating the integration of a favicon for brand identity. </body> </html>	
Your Task	Not working, Try if you can get it working

4. Forms and Styling

---

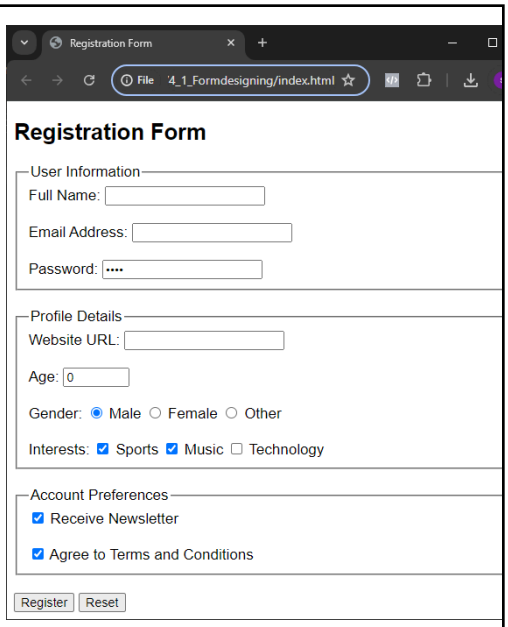
1. Design forms

Design (*form*, *label*) with a variety of input types (*input*, *type*, *name*, *id*, *text*, *email*, *url*, *number*, *password*, *checkbox*, *radio*, *submit*, *reset*, *textarea*) and understand their specific use cases.

Scenario: You are a web developer tasked with creating a registration form for a new online community platform. Your form should include:

1. **User Information:**
    - Full Name
    - Email Address
    - Password
2. **Profile Details:**
    - Website URL
    - Age
    - Gender (Male, Female, Other)
    - Interests (multiple options)
3. **Account Preferences:**
    - Receive Newsletter (Yes/No)
    - Agree to Terms and Conditions (Checkbox)
4. **Form Actions:**
    - Submit Button
    - Reset Button

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Registration Form</title>
</head>
<body>
    <h2>Registration Form</h2>
    <form
action="/submit_registration"
method="post">
        <!-- User Information -->
        <fieldset>
            <legend>User
Information</legend>
            <label for="full-name">Full
Name:</label>
            <input type="text"
id="full-name" name="fullName"
required><br><br>

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

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

        <!-- Profile Details -->
        <fieldset>
            <legend>Profile
Details</legend>
            <label for="website">Website
URL:</label>
            <input type="url" id="website"
name="website"><br><br>

            <label
for="age">Age:</label>
            <input type="number"
```



```
id="age" name="age" min="0"
max="120"><br><br>

        <label>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>
        <input type="radio" id="other"
name="gender" value="other">
        <label
for="other">Other</label><br><br>

        <label>Interests:</label>
        <input type="checkbox"
id="sports" name="interests"
value="sports">
        <label
for="sports">Sports</label>
        <input type="checkbox"
id="music" name="interests"
value="music">
        <label
for="music">Music</label>
        <input type="checkbox"
id="tech" name="interests"
value="tech">
        <label
for="tech">Technology</label>
    </fieldset>
    <br>

    <!-- Account Preferences -->
    <fieldset>
        <legend>Account
Preferences</legend>
        <input type="checkbox"
id="newsletter" name="newsletter">
        <label
for="newsletter">Receive
Newsletter</label><br><br>

        <input type="checkbox"
id="terms" name="terms" required>
        <label for="terms">Agree to
Terms and Conditions</label>
```

</fieldset> <!-- Form Actions --> <input type="submit" value="Register"> <input type="reset" value="Reset"> </form> </body> </html>	
Your Task	Add Company Name and add two radio options “Offline” or “Online”

2. GET vs. POST

Scenario :

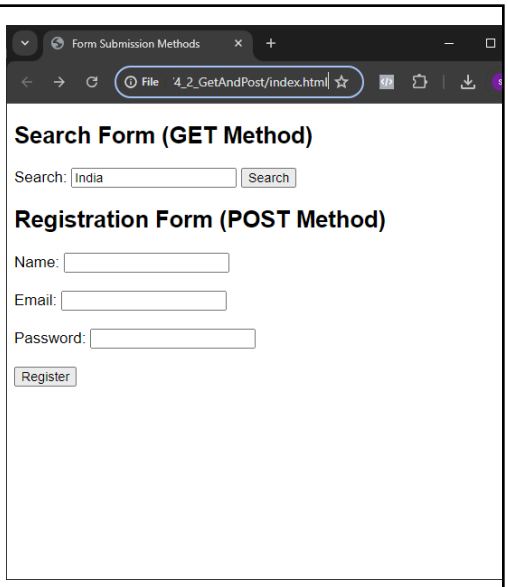
Search Form:

- Should allow users to search for content on the website.
- Search queries should be visible in the URL so users can bookmark or share the search results.

Registration Form:

- Should collect sensitive user information such as name, email, and password.
- Data should be submitted securely and not be visible in the URL.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Form Submission
Methods</title>
</head>
<body>
    <h2>Search Form (GET
Method)</h2>
    <form action="/search"
method="get">
        <label
for="query">Search:</label>
        <input type="text" id="query"
name="query" required>
        <input type="submit"
value="Search">
    </form>

    <h2>Registration Form (POST
Method)</h2>
    <form action="/register"
method="post">
        <label
for="name">Name:</label>
        <input type="text" id="name"
name="name" required><br><br>

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

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

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



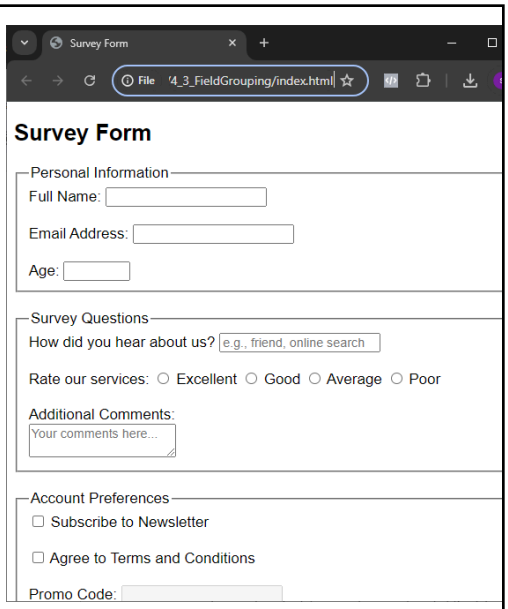
Your Task	Take a break, no task!

3. Field Grouping

Implement field grouping with *fieldset* and *legend* tags for enhanced form usability and use form-specific attributes to customize form behavior and appearance (*value*, *placeholder*, *disabled*, *readonly*).

Scenario : You are a web developer creating a survey form for an online study. The form should be user-friendly and organized into distinct sections. Additionally, some fields should have predefined values, placeholders, or be read-only/disabled to guide user input and improve the form's usability.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Survey Form</title>
</head>
<body>
    <h2>Survey Form</h2>
    <form action="/submit_survey"
method="post">
        <!-- Personal Information
Section -->
        <fieldset>
            <legend>Personal
Information</legend>
            <label for="full-name">Full
Name:</label>
            <input type="text"
id="full-name" name="fullName"
required><br><br>

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

            <label
for="age">Age:</label>
            <input type="number"
```



```
id="age" name="age" min="0"
max="120" required>
        </fieldset>
        <br>

        <!-- Survey Questions Section
-->
        <fieldset>
            <legend>Survey
Questions</legend>
            <label for="referrer">How did
you hear about us?</label>
            <input type="text"
id="referrer" name="referrer"
placeholder="e.g., friend, online
search"><br><br>

            <label>Rate our
services:</label>
            <input type="radio"
id="excellent" name="rating"
value="Excellent">
            <label
for="excellent">Excellent</label>
            <input type="radio" id="good"
name="rating" value="Good">
            <label
for="good">Good</label>
            <input type="radio"
id="average" name="rating"
value="Average">
            <label
for="average">Average</label>
            <input type="radio" id="poor"
name="rating" value="Poor">
            <label
for="poor">Poor</label><br><br>

            <label
for="comments">Additional
Comments:</label><br>
            <textarea id="comments"
name="comments"
placeholder="Your comments
here..."></textarea>
        </fieldset>
        <br>

        <!-- Account Preferences
Section -->
        <fieldset>
```

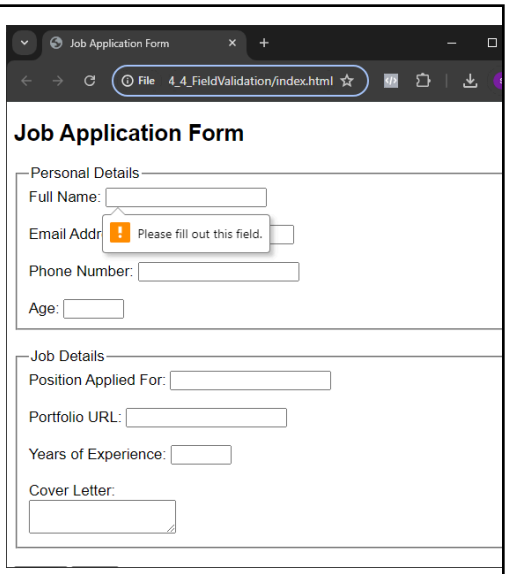
<legend>Account Preferences</legend> <input type="checkbox" id="newsletter" name="newsletter"> <label for="newsletter">Subscribe to Newsletter</label> <input type="checkbox" id="terms" name="terms" required> <label for="terms">Agree to Terms and Conditions</label> <label for="promo-code">Promo Code:</label> <input type="text" id="promo-code" name="promoCode" disabled> </fieldset> <!-- Form Actions --> <input type="submit" value="Submit"> <input type="reset" value="Reset"> </form> </body> </html>	
Your Task	Add one more section “Hobbies”. Add 3 text boxes.

4. Validation techniques

Use validation techniques with attributes (e.g., required, min, max, maxlength, autocomplete, etc.).

Scenario: Create a job application form for a company's career page. The form should validate user inputs to ensure all necessary information is provided correctly.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Job Application Form</title>
</head>
<body>
    <h2>Job Application Form</h2>
    <form action="/submit_application"
method="post" autocomplete="on">
        <!-- Personal Details Section -->
        <fieldset>
            <legend>Personal
Details</legend>
            <label for="full-name">Full
Name:</label>
            <input type="text"
id="full-name" name="fullName"
required><br><br>

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

            <label for="phone">Phone
Number:</label>
            <input type="tel" id="phone"
name="phone" pattern="[0-9]{10}"
required><br><br>

            <label
for="age">Age:</label>
            <input type="number"
id="age" name="age" min="18"
max="65" required>
        </fieldset>
        <br>

        <!-- Job Details Section -->
        <fieldset>
            <legend>Job
Details</legend>
            <label for="position">Position
Applied For:</label>
            <input type="text"
id="position" name="position"
```



required> <label for="portfolio">Portfolio URL:</label> <input type="url" id="portfolio" name="portfolio"> <label for="experience">Years of Experience:</label> <input type="number" id="experience" name="experience" min="0" max="50" required> <label for="cover-letter">Cover Letter:</label> <textarea id="cover-letter" name="coverLetter" maxlength="500" required></textarea> </fieldset> <!-- Form Actions --> <input type="submit" value="Submit"> <input type="reset" value="Reset"> </form> </body> </html>	
Your Task	Add current salary and expected salary. The salary should not be negative and must be filled in.

Validation Techniques and Attributes:

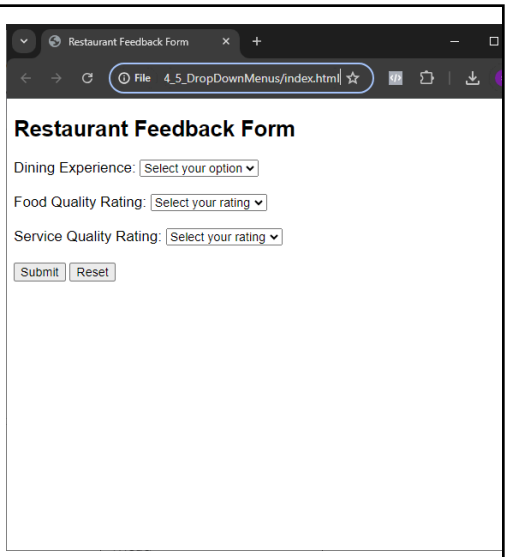
- required: Ensures the field must be filled out before form submission (used in most fields).
- min and max: Specifies the minimum and maximum values for numerical inputs (used for Age and Years of Experience).
- maxlength: Sets the maximum number of characters allowed in an input field (used for Cover Letter).
- pattern: Specifies a regular expression the input field's value must match (used for Phone Number to ensure it's exactly 10 digits).

- autocomplete: Suggests previously entered values to the user based on their browser history, enhancing the user experience (enabled for the form).

5. Create dropdown menus with select and option tags.

Scenario: Create a feedback form for a restaurant's website. The form should include a dropdown menu where users can select their dining experience type, a dropdown for rating the food quality, and a dropdown for rating the service quality.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Restaurant Feedback
Form</title>
</head>
<body>
    <h2>Restaurant Feedback
Form</h2>
    <form action="/submit_feedback"
method="post">
        <!-- Dining Experience
Dropdown -->
        <label
for="dining-experience">Dining
Experience:</label>
        <select id="dining-experience"
name="diningExperience" required>
            <option value="" disabled
selected>Select your option</option>
            <option
value="dine-in">Dine-In</option>
            <option
value="takeout">Takeout</option>
            <option
value="delivery">Delivery</option>
        </select>
        <br><br>

        <!-- Food Quality Rating
Dropdown -->
        <label for="food-quality">Food
```



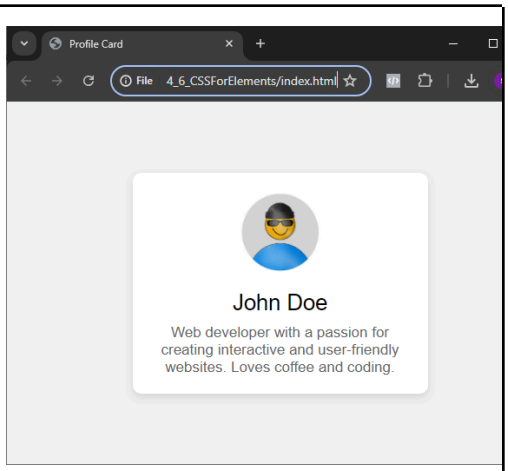
Quality Rating:</label> <select id="food-quality" name="foodQuality" required> <option value="" disabled selected>Select your rating</option> <option value="excellent">Excellent</option> <option value="good">Good</option> <option value="average">Average</option> <option value="poor">Poor</option> </select> <!-- Service Quality Rating Dropdown --> <label for="service-quality">Service Quality Rating:</label> <select id="service-quality" name="serviceQuality" required> <option value="" disabled selected>Select your rating</option> <option value="excellent">Excellent</option> <option value="good">Good</option> <option value="average">Average</option> <option value="poor">Poor</option> </select> <!-- Form Actions --> <input type="submit" value="Submit"> <input type="reset" value="Reset"> </form> </body> </html>	
Your Task	

6. Apply CSS for styling HTML elements

Apply CSS for styling HTML elements, including the use of inline styles (the style attribute) and internal stylesheets (the style element).

Scenario : Create a simple profile card for a user on a social networking site. The profile card should display the user's name, a brief bio, and a profile picture. You need to style the profile card using both inline styles and an internal stylesheet to make it visually appealing.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Profile Card</title>
    <style>
        /* Internal Stylesheet */
        body {
            font-family: Arial, sans-serif;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
            background-color: #f0f0f0;
        }
        .profile-card {
            background-color: white;
            padding: 20px;
            margin: 10px;
            border-radius: 10px;
            box-shadow: 0 4px 8px
rgba(0, 0, 0, 0.1);
            text-align: center;
            width: 300px;
        }
        .profile-name {
            font-size: 1.5em;
            margin: 10px 0;
        }
        .profile-bio {
            color: #666;
        }
    </style>
</head>
<body>
```

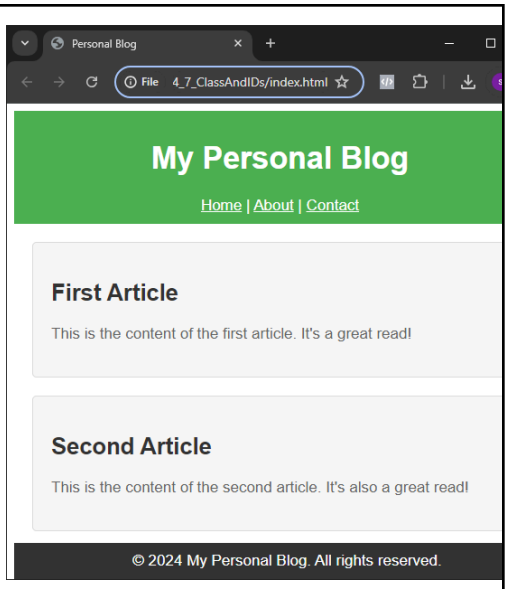


 <!-- Inline Style for Profile Picture --> <img src="profile-picture.jpeg" alt="Profile Picture" style="width: 100px; height: 100px; border-radius: 50%;"> John Doe Web developer with a passion for creating interactive and user-friendly websites. Loves coffee and coding. </body> </html>	
Your Task	Change image and background color of the profile

7. CSS classes (class) and IDs (id)

Scenario : Create a personal blog using class and id attribute.Style the header, footer and article content.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Personal Blog</title>
    <style>
        /* Styling using IDs */
        #header {
            background-color: #4CAF50;
            color: white;
            padding: 10px 0;
            text-align: center;
        }
        #footer {
            background-color: #333;
```



```
            color: white;
            text-align: center;
            padding: 10px 0;
            position: fixed;
            width: 100%;
            bottom: 0;
        }

        /* Styling using Classes */
        .article {
            background-color: #f9f9f9;
            border: 1px solid #ddd;
            margin: 20px;
            padding: 20px;
            border-radius: 5px;
        }
        .article-title {
            font-size: 1.5em;
            color: #333;
        }
        .article-content {
            color: #666;
        }
    </style>
</head>
<body>
    <!-- Header with ID -->
    <div id="header">
        <h1>My Personal Blog</h1>
        <nav>
            <a href="#home" style="color:
white;">Home</a> |
            <a href="#about" style="color:
white;">About</a> |
            <a href="#contact"
style="color: white;">Contact</a>
        </nav>
    </div>

    <!-- Main Content with Articles -->
    <div class="article">
        <h2 class="article-title">First
Article</h2>
        <p class="article-content">This
is the content of the first article. It's a
great read!</p>
    </div>
    <div class="article">
        <h2 class="article-title">Second
Article</h2>
        <p class="article-content">This
```

is the content of the second article. It's also a great read!</p> <!-- Footer with ID --> © 2024 My Personal Blog. All rights reserved. </body> </html>	
Your Task	Add your third article, Add “Jobs” option in Header.

Styling using IDs:

- IDs are used to style unique elements on the page.
- The #header ID styles the header section with a green background, white text, and center alignment.
- The #footer ID styles the footer section with a dark background, white text, center alignment, and fixes it to the bottom of the page.

Styling using Classes:

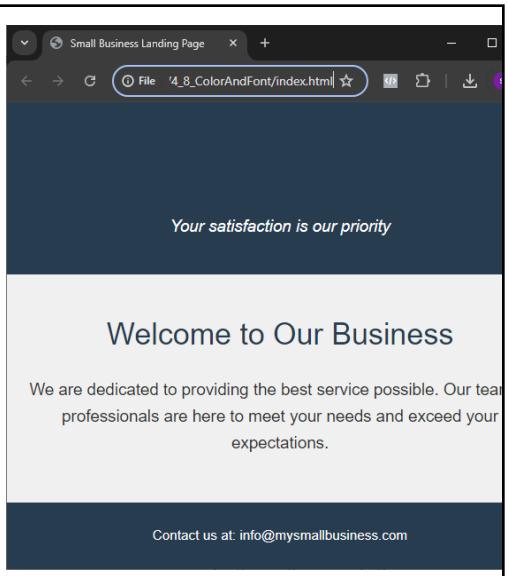
- Classes are used to style multiple elements similarly.
- The .article class styles the article containers with a light background, border, margin, padding, and rounded corners.
- The .article-title class styles the titles of the articles with a larger font size and darker color.
- The .article-content class styles the content of the articles with a lighter color.

8. Color and Font properties

Incorporate color (color, background-color) and font properties (font-weight, font-size, font-family, font-style) for consistent website styling.

Scenario : Create a simple landing page for a small business. The page should include a header, a main content section with some text, and a footer. You need to use CSS to apply consistent color and font styling throughout the page.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Small Business Landing
Page</title>
    <style>
        /* General Styling */
        body {
            font-family: 'Arial', sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f4f4f4;
            color: #333;
        }
        h1, h2 {
            color: #2c3e50;
        }
        p {
            font-size: 1.1em;
            line-height: 1.6;
        }

        /* Header Styling */
        #header {
            background-color: #2c3e50;
            color: white;
            padding: 20px 0;
            text-align: center;
        }
        #header h1 {
            font-size: 2.5em;
            font-weight: bold;
        }
        #header p {
            font-style: italic;
            margin-top: 10px;
        }

        /* Main Content Styling */
        #main-content {
            padding: 20px;
            text-align: center;
        }
        #main-content h2 {
            font-size: 2em;
            font-weight: normal;
        }
```



/* Footer Styling */
#footer {
background-color: #2c3e50;
color: white;
text-align: center;
padding: 10px 0;
position: fixed;
width: 100%;
bottom: 0;
}
#footer p {
font-size: 0.9em;
}
</style>
</head>
<body>
<!-- Header -->
 	
 My Small Business
 Your satisfaction is our priority
 	
<!-- Main Content -->
 	
 Welcome to Our Business
 We are dedicated to providing the best service possible. Our team of professionals are here to meet your needs and exceed your expectations.
 	
<!-- Footer -->
 	
 Contact us at: info@mysmallbusiness.com
 	
</body>
</html>	

Your Task	Change the background color, font type, font size and check the effects.

General Styling:

- The body element is styled with a default font family (Arial, sans-serif), background color, and text color to ensure a consistent look across the page.
- Headings (h1, h2) and paragraphs (p) are given specific colors and font sizes to enhance readability.

Header Styling:

- The #header ID styles the header section with a dark background color, white text color, and center alignment.
- The h1 element within the header has a large font size and bold weight to make the business name stand out.
- The p element within the header uses an italic font style for the tagline.

Main Content Styling:

- The #main-content ID styles the main content section with padding and center alignment.
- The h2 element within the main content has a font size and normal weight to distinguish it from the header.

Footer Styling:

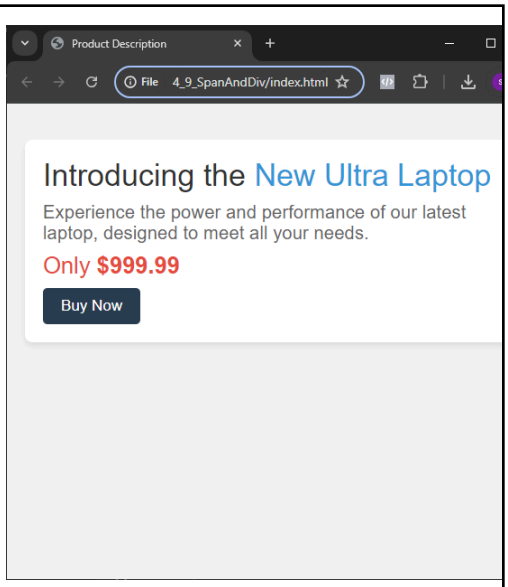
- The #footer ID styles the footer section with a dark background color, white text color, and center alignment.
- The p element within the footer has a smaller font size for contact information.

9. Span and div

Use of span and div for styling and layout.

Scenario : You are a web developer tasked with creating a simple product description section for an e-commerce website. The section should include a product title, a brief description, a price, and a "Buy Now" button. You need to use div for layout purposes and span for styling inline elements.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Product Description</title>
    <style>
        /* General Styling */
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 20px;
            background-color: #f4f4f4;
        }

        /* Product Section Styling */
        .product-section {
            background-color: white;
            padding: 20px;
            margin: 20px auto;
            border-radius: 8px;
            box-shadow: 0 4px 8px
rgba(0, 0, 0, 0.1);
            max-width: 600px;
        }
        .product-title {
            font-size: 2em;
            color: #333;
        }
        .product-description {
            font-size: 1.2em;
            color: #666;
            margin: 10px 0;
        }
        .product-price {
            font-size: 1.5em;
            color: #e74c3c;
            margin: 10px 0;
        }
        .buy-now {
            display: inline-block;
            background-color: #2c3e50;
            color: white;
            padding: 10px 20px;
            text-decoration: none;
            border-radius: 5px;
        }
    </style>
```



</head> <body> <!-- Product Section --> <!-- Product Title --> Introducing the New Ultra Laptop <!-- Product Description --> Experience the power and performance of our latest laptop, designed to meet all your needs. <!-- Product Price --> Only $999.99 <!-- Buy Now Button --> Buy Now </body> </html>	
Your Task	Add image for the laptop

Using div for Layout:

- div elements are used to create distinct sections for the product title, description, price, and button.
- div is a block-level element that helps structure the layout by creating separations between different parts of the content.

Using span for Inline Styling:

- span elements are used within div elements to style specific parts of the text.

- In the product title, a span with a specific color is used to highlight the product name ("New Ultra Laptop").
- In the product price, a span with bold font weight is used to emphasize the price.

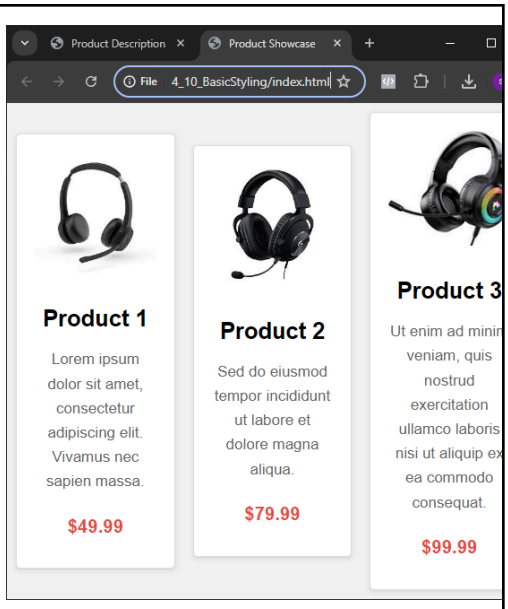
Styling with CSS:

- The general styling for the body sets a background color and applies a default font.
- The .product-section class styles the product section with a white background, padding, margin, border-radius, and box-shadow for a card-like appearance.
- The .product-title, .product-description, and .product-price classes style the respective sections with specific font sizes and colors.
- The .buy-now class styles the "Buy Now" button with a background color, text color, padding, text decoration, and border-radius.

10. border, padding, margin

Incorporate basic styling attributes to manipulate the appearance of containers (border, padding, margin).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Product Showcase</title>
    <style>
        /* General Styling */
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
            display: flex;
            justify-content: center;
            align-items: center;
            min-height: 100vh;
            margin: 0;
        }

        /* Product Card Styling */
```



```
        .product-card {
            background-color: white;
            border: 1px solid #ddd;
            border-radius: 5px;
            padding: 20px;
            margin: 10px;
            width: 300px;
            text-align: center;
            box-shadow: 0 2px 4px
rgba(0, 0, 0, 0.1);
        }
        .product-card img {
            width: 100%;
            border-radius: 5px;
            margin-bottom: 10px;
        }
        .product-card h2 {
            font-size: 1.5em;
            margin-bottom: 10px;
        }
        .product-card p {
            color: #666;
            line-height: 1.6;
            margin-bottom: 10px;
        }
        .product-card .price {
            font-size: 1.2em;
            color: #e74c3c;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <div class="product-card">
        <img src="product1.jpg"
alt="Product 1">
        <h2>Product 1</h2>
        <p>Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Vivamus
nec sapien massa.</p>
        <p class="price">$49.99</p>
    </div>

    <div class="product-card">
        <img src="product2.jpg"
alt="Product 2">
        <h2>Product 2</h2>
        <p>Sed do eiusmod tempor
incididunt ut labore et dolore magna
aliqua.</p>
        <p class="price">$79.99</p>
```

    <img src="product3.jpg" alt="Product 3"> Product 3 Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. $99.99 </body> </html>	
Your Task	Add product4

Using border:

- Each .product-card class applies a border with 1px solid #ddd to outline each product card, giving them a distinct boundary.

Using padding:

- The .product-card class uses padding: 20px to create space inside each card, ensuring the content (image, title, description, price) is not too close to the edges.

Using margin:

- The .product-card class uses margin: 10px to create space between each product card, providing visual separation and improving readability.

Styling Details:

- Images (img) within each product card are styled to be responsive (width: 100%) and have rounded corners (border-radius: 5px).
- Titles (h2), descriptions (p), and prices (p.price) are styled with appropriate font sizes, colors, and margins to improve readability and hierarchy.

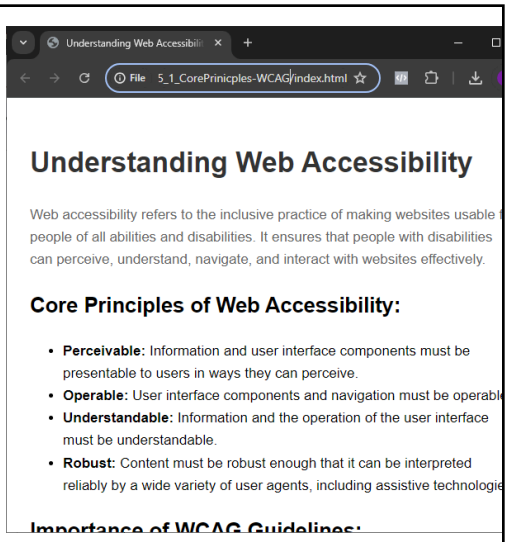
5. Best Practices and Modern HTML

1. Core Principles

The core principles of web accessibility and the importance of following WCAG guidelines.

Scenario : You are a web developer working on a project for a public service website that provides important information to citizens.Create a webpage that adheres to web accessibility principles to ensure it can be accessed and used by all individuals, including those with disabilities.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Understanding Web
Accessibility</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
            padding: 20px;
        }
        .container {
            max-width: 800px;
            margin: 0 auto;
        }
        h1 {
            color: #333;
        }
        p {
            color: #666;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Understanding Web
```



Accessibility</h1> Web accessibility refers to the inclusive practice of making websites usable for people of all abilities and disabilities. It ensures that people with disabilities can perceive, understand, navigate, and interact with websites effectively. <h2>Core Principles of Web Accessibility:</h2> - Perceivable: Information and user interface components must be presentable to users in ways they can perceive. - Operable: User interface components and navigation must be operable. - Understandable: Information and the operation of the user interface must be understandable. - Robust: Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies. <h2>Importance of WCAG Guidelines:</h2> The Web Content Accessibility Guidelines (WCAG) provide detailed guidelines for making web content accessible. Following WCAG ensures that websites are usable by people with a wide range of disabilities, including visual, auditory, physical, speech, cognitive, language, learning, and neurological disabilities. Benefits of following WCAG guidelines include: - Enhancing usability for all users, including those with	

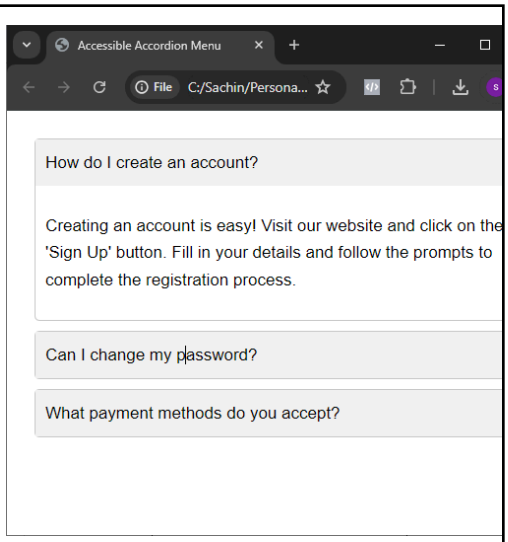
disabilities. - Increasing audience reach and user engagement. - Meeting legal requirements and avoiding potential lawsuits related to accessibility. - Improving overall website quality and user satisfaction. By adhering to web accessibility principles and WCAG guidelines, web developers contribute to a more inclusive and accessible online environment for everyone. </body> </html>	
Your Task	Read the HTML page carefully.

2. ARIA

Utilize ARIA roles (role), states, and properties (aria-expanded, aria-checked, aria-hidden) to make web content more accessible.

Scenario : You are developing an interactive accordion menu for a website that displays frequently asked questions (FAQs). Each question can be expanded to reveal its answer. Your task is to use ARIA roles, states, and properties to ensure the accordion menu is accessible to all users, including those who rely on assistive technologies.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Accessible Accordion
Menu</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
            padding: 20px;
        }
        .accordion-item {
            border: 1px solid #ccc;
            border-radius: 5px;
            margin-bottom: 10px;
        }
        .accordion-question {
            background-color: #f0f0f0;
            padding: 10px;
            cursor: pointer;
        }
        .accordion-answer {
            padding: 10px;
            display: none;
        }
        .accordion-answer.active {
            display: block;
        }
    </style>
</head>
<body>
    <div class="accordion-item"
role="presentation">
        <div class="accordion-question"
role="button" aria-expanded="false"
aria-controls="answer1"
onclick="toggleAccordion(this)">
            How do I create an account?
        </div>
        <div class="accordion-answer"
id="answer1" aria-hidden="true">
            <p>Creating an account is
easy! Visit our website and click on
the 'Sign Up' button. Fill in your
details and follow the prompts to
complete the registration
```



```
process.</p>
        </div>
    </div>

    <div class="accordion-item"
role="presentation">
        <div class="accordion-question"
role="button" aria-expanded="false"
aria-controls="answer2"
onclick="toggleAccordion(this)">
            Can I change my password?
        </div>
        <div class="accordion-answer"
id="answer2" aria-hidden="true">
            <p>Yes, you can change your
password anytime by logging into
your account, navigating to the
settings or profile section, and
selecting the option to change your
password.</p>
        </div>
    </div>

    <div class="accordion-item"
role="presentation">
        <div class="accordion-question"
role="button" aria-expanded="false"
aria-controls="answer3"
onclick="toggleAccordion(this)">
            What payment methods do
you accept?
        </div>
        <div class="accordion-answer"
id="answer3" aria-hidden="true">
            <p>We accept major credit
cards, PayPal, and bank transfers.
You can choose your preferred
payment method during the checkout
process.</p>
        </div>
    </div>

    <script>
        function
toggleAccordion(element) {
            var answerId =
element.getAttribute('aria-controls');
            var answer =
document.getElementById(answerId)
;
```

if (element.getAttribute('aria-expanded') === 'true') {
 	
element.setAttribute('aria-expanded', 'false');
 	
answer.setAttribute('aria-hidden', 'true');
 	
answer.classList.remove('active');
} else {
 	
element.setAttribute('aria-expanded', 'true');
 	
answer.setAttribute('aria-hidden', 'false');
 	
answer.classList.add('active');
}
}
</script>
</body>
</html>	

Your Task	Add a new Question “Whether the sessions will be online or offline”. Provide an answer for that.

Accordion Menu Structure:

- Each .accordion-item represents a FAQ item consisting of a question (accordion-question) and an answer (accordion-answer).

ARIA Roles and Attributes:

- role="presentation" on .accordion-item indicates it's a structural container.
- role="button" on .accordion-question makes it accessible as a button.
- aria-expanded="false" initially on each question indicates the answer is hidden.
- aria-controls attribute links each question to its corresponding answer for screen readers (id="answer1", id="answer2", id="answer3").
- aria-hidden="true" on each answer initially hides the answer content from screen readers until expanded.

JavaScript Function:

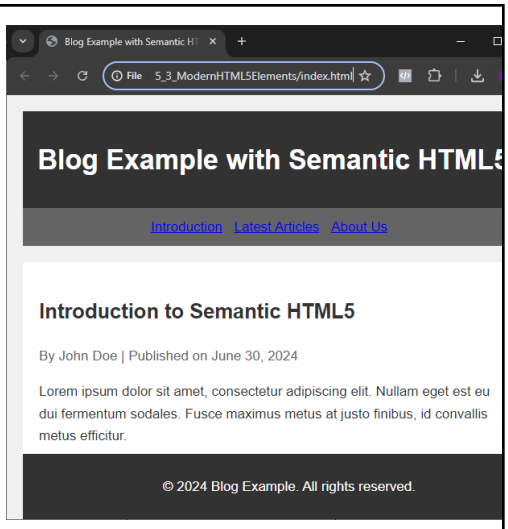
- toggleAccordion() function toggles the visibility of the answer (accordion-answer) when the question (accordion-question) is clicked.
- It updates aria-expanded and aria-hidden attributes dynamically based on the current state (true for expanded, false for collapsed).
- CSS (display: none; and display: block;) controls the visibility of the answer content based on its active class.

3. header, nav, main, article, section, aside, footer

Implement semantic markup and integrate modern HTML5 structural elements for document organization (header, nav, main, article, section, aside, footer).

Scenario : You are tasked with creating a webpage for a blog that features articles on various topics. Your goal is to implement semantic HTML5 elements to structure the document effectively, ensuring it is well-organized and accessible.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Blog Example with Semantic
HTML5</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
            background-color: #f0f0f0;
            margin: 0;
            padding: 20px;
        }
        header {
            background-color: #333;
            color: white;
            text-align: center;
            padding: 10px 0;
        }
        nav {
            background-color: #666;
            padding: 10px;
        }
        nav ul {
```



```
            list-style-type: none;
            padding: 0;
            margin: 0;
            text-align: center;
        }
        nav ul li {
            display: inline;
            margin-right: 10px;
        }
        main {
            background-color: white;
            padding: 20px;
            border-radius: 5px;
            margin-top: 20px;
        }
        article {
            margin-bottom: 20px;
            border-bottom: 1px solid
#ccc;
            padding-bottom: 20px;
        }
        article h2 {
            color: #333;
        }
        article .meta {
            color: #666;
            margin-bottom: 10px;
        }
        article p {
            color: #333;
        }
        footer {
            text-align: center;
            background-color: #333;
            color: white;
            padding: 10px 0;
            position: fixed;
            bottom: 0;
            width: 100%;
        }
    </style>
</head>
<body>
    <header>
        <h1>Blog Example with
Semantic HTML5</h1>
    </header>

    <nav>
        <ul>
            <li><a
```

```
href="#section1">Introduction</a></li
>
            <li><a
href="#section2">Latest
Articles</a></li>
            <li><a
href="#section3">About Us</a></li>
        </ul>
    </nav>

    <main>
        <section id="section1">
            <article>
                <h2>Introduction to
Semantic HTML5</h2>
                <p class="meta">By John
Doe | Published on June 30,
2024</p>
                <p>Lorem ipsum dolor sit
amet, consectetur adipiscing elit.
Nullam eget est eu dui fermentum
sodales. Fusce maximus metus at
justo finibus, id convallis metus
efficitur.</p>
            </article>
        </section>

        <section id="section2">
            <article>
                <h2>Exploring Modern
Web Development Techniques</h2>
                <p class="meta">By Jane
Smith | Published on July 1,
2024</p>
                <p>Vestibulum facilisis
diam a dui rutrum, sed eleifend
magna lacinia. Nunc vel ultrices ex.
Etiam at massa vitae ligula pretium
gravida.</p>
            </article>

            <article>
                <h2>Understanding CSS
Grid Layout</h2>
                <p class="meta">By Emily
Johnson | Published on July 2,
2024</p>
                <p>Curabitur efficitur
blandit mi, sed vehicula nisi. Proin ac
nulla eget orci suscipit eleifend vel et
odio.</p>
```

</article> </section> <aside> <section id="section3"> About Us Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam eget est eu dui fermentum sodales. Fusce maximus metus at justo finibus, id convallis metus efficitur. </section> </aside> </main> <footer> © 2024 Blog Example. All rights reserved. </footer> </body> </html>	
Your Task	Change color of the webpage, Add Resource in header navigation.Add support email in footer

- **Semantic HTML5 Elements:**
  - **header**: Contains the main heading of the blog.
  - **nav**: Contains navigation links (ul and li) for easy access to different sections.
  - **main**: Wraps the main content of the blog.
  - **section**: Groups related articles (article) within the main content.
  - **article**: Represents each blog post with a title (h2), author meta information (p.meta), and content (p).
- **Styling Details:**
  - Various CSS styles (header, nav, main, article, footer) are applied to enhance the visual presentation and layout of the webpage.
  - footer is fixed at the bottom of the page using position: fixed; bottom: 0; to ensure it stays visible.

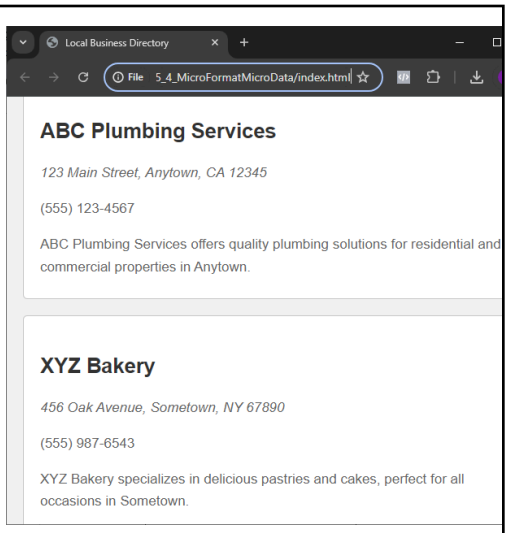
By implementing semantic HTML5 elements (header, nav, main, article, section, aside, footer), you improve the document structure and accessibility of the webpage. This approach not only enhances SEO but also makes it easier for assistive technologies to navigate and interpret the content, providing a better user experience overall.

4. Microformats & microdata

Apply microformats (hCard, vCard), microdata, and schema.org (itemprop, itemscope) for enriched content semantics.

Scenario :Develop a webpage for a local business directory that lists various businesses along with their contact information. Your task is to implement microformats (hCard, vCard), microdata, and schema.org markup to enhance the semantics of the business listings for improved search engine visibility and structured data understanding.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Local Business
Directory</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
            background-color: #f0f0f0;
            padding: 20px;
            margin: 0;
        }
        .business {
            background-color: white;
            border: 1px solid #ccc;
            border-radius: 5px;
            padding: 20px;
            margin-bottom: 20px;
        }
        .business h2 {
            color: #333;
            margin-bottom: 10px;
        }
        .business p {
            color: #666;
            margin-bottom: 5px;
        }
        .business .address {
            font-style: italic;
        }
```



```
    </style>
</head>
<body>
    <div class="business" itemscope
itemtype="http://schema.org/LocalBu
siness">
        <h2 itemprop="name">ABC
Plumbing Services</h2>
        <p class="address"
itemprop="address" itemscope
itemtype="http://schema.org/PostalA
ddress">
            <span
itemprop="streetAddress">123 Main
Street</span>,
            <span
itemprop="addressLocality">Anytown
</span>,
            <span
itemprop="addressRegion">CA</spa
n>
            <span
itemprop="postalCode">12345</spa
n>
        </p>
        <p itemprop="telephone">(555)
123-4567</p>
        <p itemprop="description">ABC
Plumbing Services offers quality
plumbing solutions for residential and
commercial properties in
Anytown.</p>
    </div>

    <div class="business" itemscope
itemtype="http://schema.org/LocalBu
siness">
        <h2 itemprop="name">XYZ
Bakery</h2>
        <p class="address"
itemprop="address" itemscope
itemtype="http://schema.org/PostalA
ddress">
            <span
itemprop="streetAddress">456 Oak
Avenue</span>,
            <span
itemprop="addressLocality">Someto
wn</span>,
            <span
itemprop="addressRegion">NY</spa
```

n> 67890 (555) 987-6543 XYZ Bakery specializes in delicious pastries and cakes, perfect for all occasions in Sometown. </body> </html>	
Your Task	Add your Computer/laptop services business in the listing

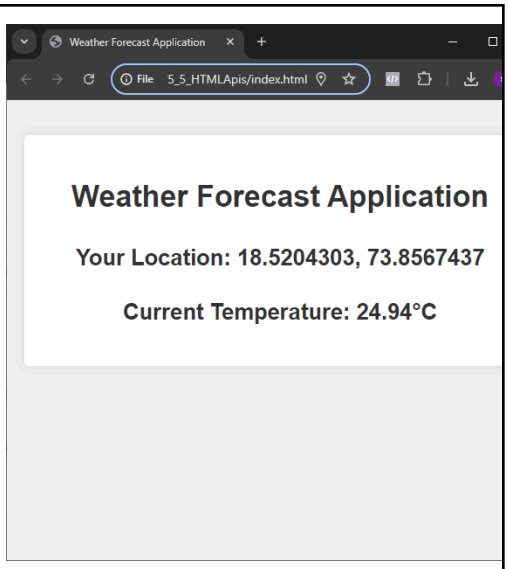
- **HTML Markup:**
  - Each .business div represents a business listing.
  - itemscope indicates the scope of the structured data (each business).
  - itemtype="http://schema.org/LocalBusiness" specifies the type of schema being used (local business).
- **Microformats and Schema.org Markup:**
  - **itemprop**: Specifies properties of the schema (e.g., name, address, telephone, description).
  - **itemscope**: Defines a section that contains properties about an item (here, each business).
  - **itemtype**: Specifies the type of schema being used (http://schema.org/LocalBusiness).
- **Styling Details:**
  - CSS styles (body, .business, h2, p) are applied for basic layout and visual presentation of each business listing.

By implementing microformats (hCard, vCard), microdata, and schema.org markup in this example, you enhance the semantic meaning of the business listings. This structured data helps search engines better understand and display relevant information about local businesses, improving visibility and accessibility of the content.

5. Basics of HTML APIs

Understand and apply the basics of HTML APIs (Geolocation, Web Storage) in web applications.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Weather Forecast
Application</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
            background-color: #f0f0f0;
            padding: 20px;
            margin: 0;
        }
        .container {
            max-width: 600px;
            margin: 20px auto;
            background-color: #fff;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px
rgba(0,0,0,0.1);
        }
        h1, h2 {
            text-align: center;
            color: #333;
        }
        .location {
            text-align: center;
            margin-bottom: 20px;
        }
        .weather {
            text-align: center;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Weather Forecast
Application</h1>

        <div class="location">
            <h2 id="location">Detecting
Location...</h2>
        </div>
```



```
        <div class="weather">
            <h2
id="temperature">Loading...</h2>
        </div>
    </div>

    <script>
        // Get user's location using
Geolocation API

navigator.geolocation.getCurrentPosi
tion(function(position) {
            var latitude =
position.coords.latitude;
            var longitude =
position.coords.longitude;

            // Display user's location
            var locationElement =
document.getElementById('location');
            locationElement.textContent
= 'Your Location: ' + latitude + ', ' +
longitude;

            // Fetch weather data based
on coordinates (using
OpenWeatherMap API as example)
            var apiKey =
'YOUR_OPENWEATHERMAP_API_
KEY';
            var apiUrl =
'https://api.openweathermap.org/data
/2.5/weather?lat=' + latitude + '&lon='
+ longitude + '&appid=' + apiKey +
'&units=metric';

            fetch(apiUrl)
                .then(response =>
response.json())
                .then(data => {
                    var temperatureElement
=
document.getElementById('temperat
ure');

temperatureElement.textContent =
'Current Temperature: ' +
data.main.temp + '°C';
                })
                .catch(error => {
                    console.log('Error
```

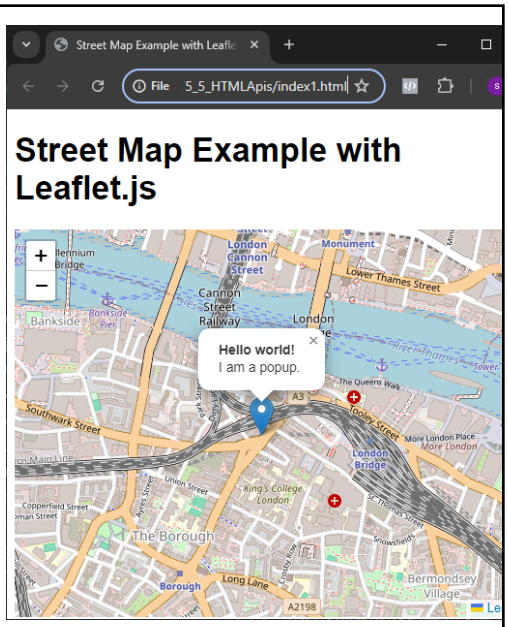
fetching weather data:', error);
var temperatureElement =
document.getElementById('temperature');
 	
temperatureElement.textContent =
'Error fetching weather data';
});
}, function(error) {
console.log('Error getting user location:', error);
var locationElement =
document.getElementById('location');
locationElement.textContent = 'Unable to retrieve your location';
});
</script>
</body>
</html>	

Your Task	Go to the site "https://openweathermap.org/" create a key and use it instead of YOUR_OPENWEATHERMAP_API_KEY in html

Scenario :

You are developing a travel blog website where users can share their travel experiences. Your task is to integrate HTML APIs such as Geolocation and Web Storage to enhance user experience and functionality on the website.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Street Map Example with
Leaflet.js</title>
    <!-- Include Leaflet CSS -->
    <link rel="stylesheet"
href="https://unpkg.com/leaflet/dist/le
aflet.css" />

    <style>
        #map {
            height: 400px;
        }
    </style>
</head>
<body>
    <h1>Street Map Example with
Leaflet.js</h1>

    <div id="map"></div>

    <!-- Include Leaflet JavaScript -->
    <script
src="https://unpkg.com/leaflet/dist/lea
flet.js"></script>

    <script>
        // Initialize the map
        var map =
L.map('map').setView([51.505,
-0.09], 13); // London coordinates

        // Add a tile layer (base map)

L.tileLayer('https://{s}.tile.openstreet
map.org/{z}/{x}/{y}.png', {
            maxZoom: 19,
        }).addTo(map);

        // Add a marker at a specific
location
        var marker = L.marker([51.505,
-0.09]).addTo(map)
            .bindPopup('Marker
Location');
```



// Add a popup to the marker marker.bindPopup("Hello world! I am a popup.").openPopup(); </script> </body> </html>	
Your Task	**Customization:** Explore Leaflet.js documentation to customize markers, popups, map styles, and layers further. **Interaction:** Implement additional features such as user interaction, overlays (polygons, circles), and integrating with external data sources.

Geolocation API:

- The Geolocation API (navigator.geolocation.getCurrentPosition) retrieves the user's current latitude and longitude coordinates.
- These coordinates are displayed on the webpage (#location) and stored in Web Storage (localStorage) for future use.

Web Storage:

- localStorage is used to store the user's current location (localStorage.setItem('currentLocation', JSON.stringify({latitude, longitude}))).

Interactive Map:

- The Leaflet.js library is used to display an interactive map (#map) centered on the user's detected location.
- A marker is placed on the map to indicate the user's current location, enhancing visual feedback.

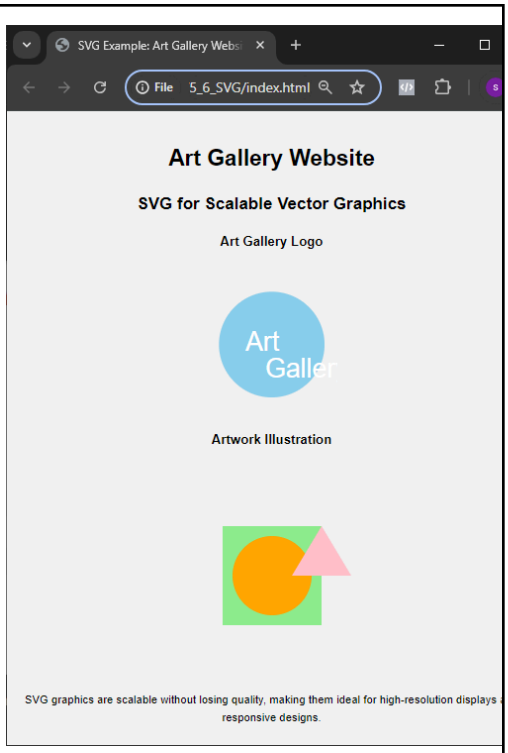
6. SVG

Utilize SVG for scalable vector graphics, focusing on its benefits and basic syntax.

Scenario :

You are developing a website for an art gallery that showcases various artworks, including logos and illustrations. Your task is to utilize SVG for creating scalable vector graphics to ensure high-quality and resizable images across different devices.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>SVG Example: Art Gallery
Website</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
            background-color: #f0f0f0;
            padding: 20px;
            margin: 0;
            text-align: center;
        }
        .logo-svg {
            width: 200px;
            height: auto;
            margin-top: 20px;
        }
        .artwork-svg {
            width: 300px;
            height: auto;
            margin-top: 20px;
        }
    </style>
</head>
<body>
    <h1>Art Gallery Website</h1>

    <h2>SVG for Scalable Vector
Graphics</h2>

    <h3>Art Gallery Logo</h3>
    <svg class="logo-svg"
```



xmlns="http://www.w3.org/2000/svg" viewBox="0 0 100 100"> <circle cx="50" cy="50" r="40" fill="skyblue" /> <text x="30" y="55" font-family="Arial" font-size="20" fill="white">Art</text> <text x="45" y="75" font-family="Arial" font-size="20" fill="white">Gallery</text> </svg> Artwork Illustration <svg class="artwork-svg" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 200 200"> <rect x="50" y="50" width="100" height="100" fill="lightgreen" /> <circle cx="100" cy="100" r="40" fill="orange" /> <polygon points="150,50 180,100 120,100" fill="pink" /> </svg> SVG graphics are scalable without losing quality, making them ideal for high-resolution displays and responsive designs. </body> </html>	
Your Task	Add your own more SVG Element

- **SVG Basics:**
  - **Inline SVG:** SVG elements (<svg>, <circle>, <rect>, <text>, <polygon>) are used directly within the HTML document.
  - **Attributes:** Attributes like xmlns, viewBox, and shape attributes (cx, cy, r for circles; x, y, width, height for rectangles) are used to define and style SVG elements.
- **Scalability and Benefits:**
  - **Scalable:** SVG graphics (<svg>) are scalable vector graphics, meaning they can be resized without losing quality.
  - **Interactivity:** SVG supports interactivity and animation, enhancing user engagement.
  - **Accessibility:** SVG can be accessible with proper text alternatives (<title>, <desc>) for screen readers.

Additional Considerations

- **Advanced Features:** Explore advanced SVG features like gradients, filters, animations (<animate>, <animateTransform>), and SVG scripting (<script>).
- **Integration:** Use SVG in conjunction with CSS for styling (fill, stroke, stroke-width) and JavaScript for dynamic behavior.

By leveraging SVG in your web development projects, especially in contexts like art galleries or any visual-heavy websites, you can ensure that graphics maintain quality across various screen resolutions and devices, providing a better user experience.

7. HTML5 development best practices

HTML5 development best practices focusing on code readability and performance:

- **Semantic Markup:** Use appropriate HTML5 tags (<header>, <nav>, <section>, <article>, <footer>) for clearer structure.
- **Clean Code:** Maintain consistent indentation, proper spacing, and meaningful naming conventions for elements and attributes.
- **Accessibility:** Ensure use of ARIA roles (role), labels (aria-label, aria-labelledby) for screen readers and assistive technologies.
- **Optimized Loading:** Minimize HTTP requests, use asynchronous loading (async, defer attributes), and optimize images (<img> tags) for faster page load times.
- **CSS and JavaScript:** Externalize styles (<link> tag) and scripts (<script> tag) for cacheability and maintainability.
- **Mobile Responsiveness:** Implement responsive design using media queries (@media) for varied screen sizes.
- **Validation:** Validate HTML markup using tools like W3C Validator to ensure compliance with standards.
- **SEO-Friendly:** Use proper <meta> tags (description, keywords) and structured data (microdata, schema.org) for search engine optimization.
- **Testing and Debugging:** Regularly test across browsers (Chrome, Firefox, Safari, Edge) and devices for consistent performance and functionality.
- **Documentation:** Document code structure, functionalities, and dependencies for future maintenance and collaboration.

8. Testing web accessibility

Introduction to Web Accessibility

Web accessibility ensures that people with disabilities can perceive, understand, navigate, and interact with the web. It involves designing websites and applications that are inclusive and accessible to all users, regardless of their abilities or impairments.

Why Test for Web Accessibility?

Testing for web accessibility is crucial to:

- **Ensure Inclusivity:** Make sure all users, including those with disabilities, can access and use your website.

- **Legal Compliance:** Many countries have laws mandating accessibility for public sector websites and services.
- **Enhance Usability:** Improve overall usability and user experience for everyone.

Principles of Web Accessibility

Understanding the core principles of web accessibility helps guide testing efforts:

- **Perceivable:** Information and user interface components must be presentable to users in ways they can perceive.
- **Operable:** Users must be able to operate the interface (e.g., navigate forms and buttons) successfully.
- **Understandable:** Content and operation of the interface must be understandable.
- **Robust:** Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.

Basic Testing Techniques

1. **Keyboard Navigation:**
    - Ensure all functionality is operable via keyboard alone (Tab key navigation, Enter key for buttons).
2. **Screen Reader Compatibility:**
    - Test using screen readers (e.g., NVDA, VoiceOver) to verify content is properly announced and navigable.
3. **Color Contrast:**
    - Check color contrast ratios (use tools like WebAIM's Color Contrast Checker) to ensure readability for users with low vision.
4. **Semantic HTML:**
    - Use semantic HTML (e.g., <button>, <form>, <table>) to ensure proper interpretation by assistive technologies.
5. **Alternative Text for Images:**
    - Ensure all images have descriptive alternative text (alt attribute) to convey meaning to users who cannot see them.
6. **Form Accessibility:**
    - Validate form fields have accessible labels (<label> tags) and provide error messages that are programmatically associated with inputs.
7. **Focus Management:**
    - Verify focus indicators (:focus styles) are visible and correctly styled to indicate where keyboard focus is.

Tools for Accessibility Testing

- **WAVE Toolbar:** Browser extension to evaluate web accessibility directly within your browser.
- **Lighthouse:** Built into Chrome's DevTools, it includes an accessibility audit feature.
- **AXE Accessibility Checker:** Browser extension or API for testing against WCAG (Web Content Accessibility Guidelines) standards.

Testing Methodology

1. **Automated Testing:**
    - Use tools for quick checks and to catch basic accessibility issues.
2. **Manual Testing:**
    - Conduct thorough manual checks to ensure usability and compliance with WCAG standards.
3. **User Testing:**
    - Involve users with disabilities in testing to gather firsthand feedback on accessibility barriers.

Continuous Improvement

- **Feedback Loop:** Gather feedback from users and accessibility experts to continually improve accessibility features.
- **Regular Audits:** Conduct periodic accessibility audits to ensure ongoing compliance with standards and guidelines.

Conclusion

Understanding and applying the basics of testing web accessibility is essential for creating inclusive web experiences. By adhering to accessibility principles, conducting comprehensive testing, and using appropriate tools, developers can ensure their websites are accessible to everyone, regardless of their abilities or disabilities. This commitment not only enhances usability but also meets legal requirements and demonstrates social responsibility in web development practices.