HTML, CSS and JavaScript Exercises

HTML Exercises

1. Create a Page with a Clear Heading Structure

Create a webpage with one <h1> for the main topic and <h2> headings for its subtopics. Keep heading levels in a logical order.

2. Add a Paragraph and a Div

Add a <p> describing your favorite hobby and a that groups related content about why you enjoy it.

3. Use Strong Emphasis

Write a sentence about your favorite food and use to mark the food name as important. Notice that semantic emphasis is more meaningful than changing appearance alone.

4. Create Bulleted and Numbered Lists

Make a webpage with a <ul> of three favorite movies and an <ol> of three favorite books.

5. Separate Two Sections

Use a thematic break <hr> between two related sections. Style it in CSS with a thicker red border.

6. Build an Accessible Table

Create a <table> with a short <caption>, column headings in <th> cells, and two data rows. List flower names and sizes, such as Rose - Small and Tulip - Large.

7. Link to Another Page

Add an <a> link to a trustworthy page about your favorite animal. Write descriptive link text instead of “click here.” If it opens a new tab, use rel="noopener".

8. Display an Image with a Caption

Put a landscape <img> and <figcaption> inside a <figure>. Add useful alt text to the image.

9. Create a CSS Animation

Display “Welcome to my website!” in a <p> or . Use @keyframes and the animation property to make it gently slide or fade in. Also add a prefers-reduced-motion rule that disables the movement for visitors who request less motion.

10. Embed a Video Responsibly

Embed a video with an <iframe> inside a <figure>. Give the iframe a descriptive title, add loading="lazy", and include a <figcaption>. Use a privacy-enhanced embed option when the provider offers one.

CSS Exercises

11. Change the Background Color
Use an external stylesheet to change the page background to light blue. Make sure the text remains easy to read.

12. Style the Main Heading
Change the color and weight of the <h1> with CSS. Choose a text color with good contrast against the background.

13. Center-Align Text
Use CSS to center-align a short paragraph <p>. Keep longer paragraphs left-aligned for readability.

14. Use Classes and IDs
Create three <p> elements about plants. Use a reusable class for their shared appearance and an ID only when one element needs a unique identifier, such as for an in-page link or JavaScript target.

15. Create a Highlight Class
Wrap one important word in <mark> or . Define the highlight colors in the stylesheet instead of using an inline style attribute.

16. Change Text Color
Use CSS to change all paragraph <p> text to a readable shade of green. Check its contrast against the background.

17. Use Responsive Font Sizing
Set paragraph text with a relative unit such as rem. Try font-size: 1.125rem and confirm the page still responds to browser zoom.

JavaScript Exercises

18. Display a Welcome Message in the Page

Add an empty <p> with an ID. When the DOM is ready, set its text to “Welcome to my webpage!” Avoid an automatic alert that interrupts the visitor.

19. Change Background Color with a Button

Add a <button type="button"> that toggles a CSS class on the page to change the background color. Keep presentation rules in CSS.

20. Read and Display Form Input

Ask for a favorite color with a labeled <input> and a submit button. Display the response on the page. This provides a clearer, non-blocking experience than prompt().

21. Create a Timer Function

Add a button that starts a five-second timer, then updates a visible element with “Time’s up!” Use an aria-live="polite" status region so assistive technology can announce the result.

22. Use a For Loop

Use a for loop to create a list of the numbers 1 through 5. Add the completed list to the document in one operation.

23. Toggle a Class on an Element

Give a link an ID, select it with JavaScript, and toggle a CSS class that changes its color and font size. Use classList rather than setting multiple inline style properties.

24. Support Pointer and Keyboard Interaction

Create a button whose appearance changes on pointer hover and keyboard focus. Use CSS :hover and :focus-visible, and make sure the focus indicator stays visible.

25. Use a While Loop

Use a while loop to build a list containing “Learning is fun!” five times, then add the list to the page.

Web Page Exercise 1: “Flower Showcase”

Goal: Build a webpage that introduces different types of flowers.

Features to include:

- A logical heading structure: <h1>Beautiful Flowers</h1> followed by <h2> section headings
- A paragraph explaining why you love flowers
- A <figure> with an image, useful alt text, and a caption for each flower
- An accessible flower table with a caption and header cells
- A <ul> of flowers you want to plant
- A semantic <section> for each flower, styled with CSS

Web Page Exercise 2: “Dream Cars Garage”

Goal: Introduce your three favorite car models.

Features to include:

- One <h1>My Dream Garage</h1>
- Three car images with useful alt text and captions
- A <ul> of car brands and an <ol> of top features
- A labeled form control that asks for the visitor’s favorite car brand and displays the answer on the page
- A styled <hr> separating luxury cars from sports cars
- A button that toggles a high-contrast “Garage Mode” class

Website Exercise 1: “My Hobby Zone” (3 Pages)

Structure:

- index.html - homepage with an introduction to your hobbies
- hobbies.html - hobbies with images, paragraphs, and semantic sections
- gallery.html - a responsive image gallery using figures, alt text, and captions
- Use a consistent <nav> with descriptive links on every page; clearly mark the current page

Website Exercise 2: “Nature Explorer” (5 Pages)

Pages:

1. Home - welcome message and background color controls

2. Flowers - an accessible data table

3. Animals - image, caption, and descriptive link to a reliable video

4. Landscapes - a responsibly embedded video and readable text alignment

5. About - your journey as a nature lover, a subtle CSS entrance animation with reduced-motion support, and an accessible contact form with labels

Technical highlights:

- Use <header>, <nav>, <main>, and <footer> landmarks
- Use one external stylesheet and a responsive viewport meta tag
- Use a labeled form control instead of a JavaScript prompt
- Test navigation and controls with both a pointer and keyboard

Web App Exercise 1: “Daily Quote Launcher”

What it does: Displays a motivational quote when you open it.

How to build:

1. Create a page with <h1>Your Daily Quote</h1> and a quote inside <blockquote>

2. Use CSS to center the card, preserve readable line lengths, and create sufficient contrast

3. When the DOM is ready, choose a random quote and place it in the blockquote without showing an alert

4. Add a “New quote” button so the visitor controls when the content changes

Optional app experience: Add a standards-based web app manifest and icons, then test the browser’s install or Add to Home Screen option. Availability and menu wording vary by browser and device.

Web App Exercise 2: “My Color Picker”

What it does: Lets you choose a color and updates the background.

1. HTML: Add a visible <label> for <input type="color"> and a <button type="button"> labeled “Set Background”

2. CSS: Add a short background-color transition, disabled when prefers-reduced-motion requests reduced motion

3. JavaScript: On button click, toggle or update a CSS custom property instead of writing several inline styles

4. Check that the foreground text keeps sufficient contrast with every selectable background color

5. Use a responsive layout and test it at narrow widths and at 200% browser zoom