# HTML & CSS Practice Exercises

## Web Development

## Exercise 1: Personal Profile Page

**Skills:** Basic HTML structure, text formatting, images, lists

**Requirements:**

- Create a page about yourself with a heading, paragraph bio, and profile photo
- Add an unordered list of your hobbies
- Add an ordered list of your top 3 favorite movies and books
- Include at least 3 different heading levels (h1, h2, h3)
- Add a horizontal rule to separate sections
- Add links to your social media profiles or favorite websites

## Exercise 2: Recipe Card

**Skills:** HTML structure, CSS styling, colors, fonts

**Requirements:**

- Create a recipe page with a title, image of the dish, and description
- List ingredients in an unordered list
- List cooking steps in an ordered list
- Style the page with CSS: change font family, colors, and text sizes
- Add a background color to the page
- Style the heading with a different color
- Add a border around the recipe card and center it on the page

# Exercise 4: Contact Form

**Skills:** HTML forms, form styling, layout

**Requirements:**

- Create a contact form with: name, email, subject, and message fields
- Include a submit button
- Style all form inputs with borders, padding, and proper spacing
- Make labels bold and aligned properly
- Style the submit button with a background color and hover effect
- Make the form centered on the page with a maximum width

# Exercise 5: Image Gallery

**Skills:** CSS layout, responsive basics, image styling

**Requirements:**

- Display 6-9 images in a grid layout
- Add captions below each image
- Make images the same size
- Add spacing between images
- Add a hover effect that changes image opacity or adds a border

# Exercise 6: Semantic Text Elements Practice

**Skills:** Text-level semantic HTML elements, inline styling

**Requirements:**

- Create a page about a programming language or technology topic
- Use `<strong>` for at least 3 important terms or warnings
- Use `<em>` to emphasize at least 2 key concepts
- Include a code example using the `<code>` tag (e.g., showing HTML or CSS syntax)
- Add abbreviations with `<abbr>` and use the title attribute to show full meaning
- Mark up at least one citation using `<cite>` (e.g., referencing a book or article)
- Add a `<small>` element for copyright or legal notice in the footer
- Use `<mark>` to highlight important text that needs attention
- Include `<time>` elements for any dates mentioned (with datetime attribute)
- Use `<span>` to style specific words or phrases with custom CSS

**Bonus Challenge:** Create a glossary section that uses multiple `<abbr>` tags with proper tooltips

# Exercise 7: Simple Navigation Menu

**Skills:** HTML links, CSS styling, layout basics

**Requirements:**

- Create a horizontal navigation bar with 4-5 links (Home, About, Services, Contact)
- Style the navigation with background color
- Change link colors and remove underlines
- Add hover effects that change the link color or background
- Make the navigation bar stick to the top of the page

**Bonus Challenge:** Add padding and spacing to make links look like buttons

# Exercise 8: Product Showcase

**Skills:** CSS box model, margins, padding, borders

**Requirements:**

- Create 3 product cards side by side (use divs)
- Each card should have: an image, product name, price, and short description
- Add borders, padding, and margins to each card
- Make cards have equal heights and widths
- Add a "Buy Now" button to each card with hover effects

# Exercise 9: Blog Post Layout

**Skills:** Semantic HTML, text styling, spacing

**Requirements:**

- Use semantic HTML tags (header, article, section, footer)
- Create a blog post with title, author name, date, and multiple paragraphs
- Add at least one image with a caption
- Style the text: adjust line height, paragraph spacing, and font sizes
- Create a clear visual hierarchy with headings

**Bonus Challenge:** Add a sidebar with "Related Articles" links

# Exercise 10: Class Schedule Table (HTML Tables Basics)

**Skills:** HTML table structure, table elements, basic table attributes

**Requirements:**

- Create a weekly class schedule table with days (Monday-Friday) as columns
- Use proper table structure: `<table>`, `<thead>`, `<tbody>`, `<tfoot>`
- Include `<tr>` (table rows), `<th>` (table headers), and `<td>` (table data cells)
- Create at least 5 time slots (rows) for different class periods
- Use `colspan` to merge cells for lunch break or double periods
- Use `rowspan` to show classes that span multiple time slots
- Add a footer row with total hours per day
- Include proper table headers with scope attribute for accessibility

**Bonus Challenge:** Add a summary row in `<tfoot>` showing total weekly hours per subject

# Exercise 11: Styled Data Table with HTML

**Skills:** CSS table styling, borders, colors, hover effects

**Requirements:**

- Create a student grades table
- Style the table using HTML only (no inline CSS)
- Add borders to table, cells, or just specific areas
- Use `border-collapse` to control border spacing
- Apply different background colors to header (`<thead>`) and body (`<tbody>`)
- Style `<th>` elements differently from `<td>` elements
- Add padding to cells for better spacing
- Center-align or right-align specific columns (like numbers or prices)
- Make text bold in the header row

# Exercise 12: Responsive Table with Advanced CSS

**Skills:** Advanced CSS styling, pseudo-selectors, responsive design basics

**Requirements:**

- Create a table displaying employee information
- Use CSS to style the entire table without any HTML styling attributes
- Apply `border-collapse: collapse` or `border-collapse: separate`
- Style borders using `border` property (color, width, style)
- Use `:nth-child()` selector to create zebra striping (alternate row colors)
- Style the first column differently using `:first-child`
- Style the last row differently using `:last-child`
- Add hover effects using `:hover` pseudo-class
- Apply `text-align` for proper alignment (left for text, right for numbers)
- Use `vertical-align` to control content positioning in cells
- Add box-shadow or border-radius for modern styling
- Set a maximum width and make the table scrollable if content overflows

**Bonus Challenge:** Make the table responsive by adding horizontal scroll on small screens or converting to a card layout

# Exercise 13: Interactive Pricing Table with CSS

**Skills:** CSS-only interactivity, advanced styling, layout

**Requirements:**

- Create a pricing comparison table for subscription plans
- Style completely with CSS (no table HTML attributes for styling)
- Use `<colgroup>` and `<col>` to style entire columns
- Apply distinct styling to each pricing tier column
- Add borders strategically (not just border-all)
- Create a highlighted "recommended" column with different styling
- Style the "Sign Up" buttons in the footer row
- Add hover effects to columns or rows
- Use CSS transitions for smooth hover effects
- Apply different font weights and sizes for emphasis
- Add icons or special characters for features (✓ for included, — for not included)

**Bonus Challenge:** Make one column "pop out" with transform and shadow effects when hovered

# Learning Tips

1. **Start Simple:** Begin with HTML structure, then add CSS styling
2. **Use Comments:** Add comments in your code to remember what sections do
3. **Test Often:** View your page in the browser frequently as you code
4. **Inspect Elements:** Use browser developer tools (F12) to debug CSS
5. **Experiment:** Try changing colors, sizes, and layouts to see what happens

# Resources to Help You

- **W3Schools https://www.w3schools.com/html/ :** Quick references and examples
- **Google Fonts:** Free fonts to make your Exercises look professional
- **Color Hunt:** https://colorhunt.co/

# Assessment Criteria

For each Exercise, you'll be evaluated on:

✓ Code is properly structured and indented

✓ All requirements are met

✓ CSS is organized and uses appropriate selectors

✓ Page displays correctly in the browser

✓ Code is readable with meaningful class/id names

✓ Bonus challenges attempted (optional)

**Good luck with your practice!**

Remember, the more you code, the better you'll get.