HTML Exercises Guide

Lesson Objective

- To introduce students to the **fundamentals of web development** using HTML and CSS.
- To help students understand how to **structure web pages** with semantic HTML tags.
- To enable students to **style and format elements** using CSS (inline, ID, and class selectors).
- To prepare students for using modern CSS/JS libraries like **Bootstrap** by first practicing manual coding of layouts, styling, and simple components.
- To build confidence in **writing clean, well-structured, and reusable code** that serves as a foundation for advanced web development.

Lesson Goals

By the end of this study guide, students should be able to:

HTML Goals

- Identify and correctly use **basic HTML tags** (headings, paragraphs, images, links, lists, tables, forms).
- Understand the **HTML document structure** (doctype, `<html>`, `<head>`, `<body>`).
- Create **structured web content** with semantic meaning.

CSS Goals

- Apply **basic styling** to elements using element selectors.
- Differentiate between **ID selectors** and **Class selectors**, and apply them correctly.

- Demonstrate understanding of the **box model, text formatting, borders, backgrounds, and spacing**.
- Create **responsive layouts** using percentages, flexbox, and positioning.

Integration Goals (HTML + CSS Combined)

- Build **functional mini-projects** like navigation bars, footers, cards, and pricing tables using only HTML and CSS.
- Understand how to structure HTML in a way that **pairs naturally with CSS selectors**.
- Develop **problem-solving skills** by manually coding designs that libraries like Bootstrap automate.

Preparation for Bootstrap Goals

- Recognize **why Bootstrap exists** (to save time by providing pre-designed, responsive components).
- Map **manually coded CSS solutions** (e.g., flexbox navbar) to the **Bootstrap equivalent** (`.navbar`, `.d-flex`).
- Be confident in moving from **raw coding** → **Bootstrap-powered development**.

Exercises

1. Basic Structure

Task: Create the HTML skeleton with `<html>`, `<head>`, and `<body>`.
Explanation: Every webpage starts with this structure.

```
<!DOCTYPE html>
<html>
<head>
  <title>My Personal Webpage</title>
```

```
</head>
<body>
  <!-- Content goes here -->
</body>
</html>
```

2. Page Title

Task: Add a `<title>` inside `<head>` that says “My Personal Webpage”.
Explanation: The `<title>` appears in the browser tab.

```
<head>
  <title>My Personal Webpage</title>
</head>
```

3. Main Heading

Task: Add <h1> with your full name.
Explanation: <h1> is the largest heading, used for the main title.

```
<h1>Juan Dela Cruz</h1>
```

4. Subheading

Task: Add <h2> for “About Me”.
Explanation: <h2> is a smaller heading, often used for sections.

```
<h2>About Me</h2>
```

5. Paragraph

Task: Write a short introduction using <p>.
Explanation: <p> is for paragraphs of text.

```
<p>Hello! My name is Juan and I love web development.</p>
```

---

6. Line Break

Task: Add a `<br>` inside a paragraph.
Explanation: `<br>` inserts a line break.

```
<p>I love coding.<br>I also enjoy photography.</p>
```

---

7. Horizontal Line

Task: Add `<hr>` below your introduction.
Explanation: `<hr>` creates a horizontal divider.

```
<hr>
```

---

8. Bold Text

Task: Make your name bold using <b>.
Explanation: <b> makes text bold.

```
<p>My name is <b>Juan Dela Cruz</b>.</p>
```

---

9. Italic Text

Task: Add italic to a hobby using `<i>`.
Explanation: `<i>` makes text italic.

```
<p>I enjoy <i>painting</i> in my free time.</p>
```

---

10. Underline Text

Task: Underline a phrase using `<u>`.
Explanation: `<u>` underlines text.

```
<p>I am a <u>web developer</u>.</p>
```

---

11. Ordered List

Task: Make a list of 3 favorite foods using `<ol>`.
Explanation: `<ol>` creates a numbered list.

```
<ol>
  <li>Pizza</li>
  <li>Sushi</li>
  <li>Adobo</li>
</ol>
```

---

12. Unordered List

Task: Make a list of hobbies using `<ul>`.
Explanation: `<ul>` creates a bulleted list.

```
<ul>
  <li>Coding</li>
  <li>Photography</li>
  <li>Gaming</li>
</ul>
```

---

13. Image

Task: Add an image with `<img>`.
Explanation: `<img>` displays images.

```
<img src="myphoto.jpg" alt="My Photo" width="200">
```

14. Link

Task: Add a link to Google using `<a>`.
Explanation: `<a>` is used for hyperlinks.

```
<a href="https://www.google.com">Visit Google</a>
```

15. Email Link

Task: Make a link that opens an email app.
Explanation: Use `mailto:` for email links.

```
<a href="mailto:example@email.com">Send me an Email</a>
```

16. Table

Task: Create a 2x2 table.
Explanation: `<table>`, `<tr>`, `<td>` are used for tables.

```
<table border="1">
  <tr>
    <td>Name</td>
    <td>Age</td>
  </tr>
  <tr>
    <td>Juan</td>
    <td>20</td>
  </tr>
</table>
```

---

17. Form (Text Input)

Task: Add a text input with `<form>`.
Explanation: `<input type="text">` creates a text field.

```
<form>
  Name: <input type="text" name="fullname">
</form>
```

---

18. Form (Password Input)

Task: Add a password field.
Explanation: `<input type="password">` hides typed characters.

```
<form>
  Password: <input type="password" name="pwd">
</form>
```

---

19. Form (Button)

Task: Add a submit button.
Explanation: `<button>` creates a clickable button.

```
<form>
  <button type="submit">Submit</button>
</form>
```

---

20. Comment

Task: Write a comment in HTML.
Explanation: Comments don't show in the browser.

```
<!-- This is a comment →
```

21. Form (Radio Buttons)

Task: Add radio buttons for gender selection.
Explanation: Radio buttons let the user pick **one option** from a group.

```
<form>
  <p>Gender:</p>
  <input type="radio" name="gender" value="male"> Male<br>
  <input type="radio" name="gender" value="female"> Female
</form>
```

22. Form (Checkboxes)

Task: Add checkboxes for hobbies.
Explanation: Checkboxes allow **multiple selections**.

```
<form>
  <p>Hobbies:</p>
  <input type="checkbox" name="hobby" value="reading"> Reading<br>
  <input type="checkbox" name="hobby" value="sports"> Sports<br>
  <input type="checkbox" name="hobby" value="travel"> Travel
</form>
```

23. Form (Dropdown List)

Task: Create a dropdown menu for country selection.
Explanation: <select> with <option> creates dropdowns.

```
<form>
  <label for="country">Select Country:</label>
  <select id="country">
    <option>Philippines</option>
    <option>Japan</option>
    <option>USA</option>
  </select>
</form>
```

24. Form (Textarea)

Task: Add a multi-line text box for comments.
Explanation: <textarea> allows longer input.

```
<form>
  <label for="message">Message:</label><br>
  <textarea id="message" rows="4" cols="30"></textarea>
</form>
```

25. Add a Favicon

Task: Add a favicon in the <head>.
Explanation: Favicons appear in the browser tab.

```
<head>
  <link rel="icon" type="image/png" href="favicon.png">
</head>
```

26. Audio Player

Task: Add an audio player with controls.
Explanation: <audio> embeds audio files.

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

27. Video Player

Task: Embed a video with controls.
Explanation: <video> embeds videos.

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
</video>
```

28. Iframe (Embed Website)

Task: Embed Google inside a page.
Explanation: <iframe> displays another webpage.

```
<iframe src="https://www.google.com" width="600"
height="400"></iframe>
```

29. Iframe (Embed YouTube Video)

Task: Embed a YouTube video.
Explanation: YouTube gives <iframe> code for sharing.

```
<iframe width="560" height="315"
  src="https://www.youtube.com/embed/dQw4w9WgXcQ"
  frameborder="0" allowfullscreen>
</iframe>
```

---

30. Superscript

Task: Write “H_2O” with superscript.
Explanation: <sup> raises text.

```
<p>Water formula: H<sub>2</sub>O</p>
```

---

31. Subscript

Task: Write “x^2” with superscript.
Explanation: <sup> raises text.

```
<p>x<sup>2</sup></p>
```

---

32. Strong Text

Task: Use <strong> to emphasize.
Explanation: <strong> means important bold text.

```
<p>This is <strong>very important</strong>!</p>
```

---

33. Emphasized Text

Task: Use <em> for emphasis.
Explanation: <em> italicizes with meaning.

```
<p>Please <em>read carefully</em>.</p>
```

---

34. Small Text

Task: Add small copyright text.
Explanation: <small> makes text smaller.

```
<p><small>© 2025 My Website</small></p>
```

---

35. Blockquote

Task: Add a quote with <blockquote>.
Explanation: <blockquote> highlights long quotations.

```
<blockquote>
  "The journey of a thousand miles begins with one step."
</blockquote>
```

---

36. Inline Quote

Task: Add an inline quotation with <q>.
Explanation: <q> adds quotation marks.

```
<p>He said, <q>Hello, world!</q></p>
```

---

37. Abbreviation

Task: Add an abbreviation with <abbr>.
Explanation: <abbr> gives extra info on hover.

```
<p><abbr title="HyperText Markup Language">HTML</abbr> is the language
of the web.</p>
```

---

38. Address Tag

Task: Add a contact address.
Explanation: <address> is used for contact details.

```
<address>
  Written by Juan Dela Cruz<br>
  Visit us at www.mywebsite.com<br>
  Manila, Philippines
</address>
```

---

39. Marked Text

Task: Highlight text with <mark>.
Explanation: <mark> highlights text in yellow.

```
<p>This is a <mark>highlighted</mark> word.</p>
```

---

40. Progress Bar

Task: Add a progress bar.
Explanation: <progress> shows progress visually.

```
<p>Task Progress:</p>
<progress value="70" max="100"></progress>
```

CSS Exercises

Tag Selectors (1–20)

1. Style an `<h1>`

Explanation: Selects all `<h1>` elements and changes text color.

```
h1 {
  color: blue;
}
```

2. Style `<p>`

Explanation: Changes font size of all `<p>`.

```
p {
  font-size: 18px;
}
```

3. Style `<h2>`

```
h2 {
  text-align: center;
}
```

---

4. Style `<body>` background

```
body {
  background-color: lightgray;
}
```

---

5. Style `<h3>` with uppercase text

```
h3 {
  text-transform: uppercase;
}
```

---

6. Style `<p>` with line-height

```
p {
  line-height: 1.6;
}
```

---

7. Style `<a>` links

```
a {
  color: red;
  text-decoration: none;
}
```

---

8. Style `<ul>` list

```
ul {
  list-style-type: square;
}
```

---

9. Style `<ol>` numbers

```
ol {
  list-style-type: upper-roman;
}
```

---

10. Style `<li>` items

```
li {
  margin: 5px;
}
```

---

11. Style `<table>` borders

```
table {
  border: 1px solid black;
}
```

---

12. Style `<th>` header

```
th {
  background-color: lightblue;
}
```

---

13. Style `<td>` cells

```
td {
  padding: 10px;
}
```

---

14. Style `<img>`

```
img {
  width: 200px;
  border-radius: 10px;
}
```

15. Style `<button>`

```
button {
  background-color: green;
  color: white;
  padding: 8px;
}
```

16. Style `<form>` border

```
form {
  border: 1px solid gray;
  padding: 15px;
}
```

17. Style `<input>` fields

```
input {
  border: 2px solid blue;
}
```

18. Style `<textarea>`

```
textarea {
  width: 300px;
  height: 100px;
}
```

19. Style `<audio>`

```css
audio {
  width: 250px;
}
```

---

20. Style `<video>`

```css
video {
  border: 5px solid black;
}
```

ID Selectors (21–30)

21. Style an `<h1>` by ID

```css
#main-title {
  color: darkred;
  font-size: 30px;
}
```

---

22. Style `<p>` by ID

```css
#intro-text {
  font-style: italic;
}
```

---

23. Style a table by ID

```css
#student-table {
  border-collapse: collapse;
}
```

---

24. Style image by ID

```
#profile-pic {
  border-radius: 50%;
}
```

---

25. Style button by ID

```
#submit-btn {
  background-color: orange;
  padding: 10px;
}
```

---

26. Style div by ID

```
#container {
  margin: 20px;
  padding: 20px;
  background-color: #eee;
}
```

---

27. Style input by ID

```
#email-field {
  border: 1px solid red;
}
```

---

28. Style link by ID

```
#home-link {
  color: purple;
}
```

---

29. Style form by ID

```
#login-form {
  background: #f9f9f9;
  padding: 15px;
}
```

---

30. Style heading by ID

```
#footer-text {
  text-align: right;
}
```

Class Selectors (31–40)

31. Style multiple `<p>` with class

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

---

32. Style headings with class

```
.title {
  color: navy;
}
```

---

33. Style buttons with class

```
.btn {
  background-color: teal;
  color: white;
  padding: 10px 15px;
}
```

---

34. Style images with class

```
.rounded-img {
  border-radius: 15px;
}
```

---

35. Style div containers

```
.card {
  border: 1px solid #ccc;
  padding: 20px;
  margin: 10px;
}
```

---

36. Style highlighted text

```
.highlight {
  background-color: yellow;
}
```

---

37. Style navigation links

```
.nav-link {
  color: darkgreen;
  margin-right: 15px;
}
```

---

38. Style list items

```
.list-item {
  font-weight: bold;
}
```

---

39. Style form inputs

```
.input-field {
  border: 2px solid gray;
  padding: 5px;
}
```

---

40. Style footer text

```
.footer-note {
  font-size: 12px;
  color: gray;
}
```

HTML + CSS Combined Exercises

41. Create a Simple Header with Inline CSS

Goal: Students practice mixing HTML and CSS in one file.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    header {
      background-color: lightblue;
      padding: 20px;
      text-align: center;
    }
  </style>
</head>
<body>
  <header>
    <h1>My Website</h1>
  </header>
</body>
</html>
```

42. Two-Column Layout (Float Method)

Goal: Understand basic layout before Bootstrap's grid system.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .left {
      float: left;
      width: 50%;
      background: lightgray;
      padding: 20px;
    }
    .right {
      float: right;
      width: 50%;
      background: lightyellow;
      padding: 20px;
    }
  </style>
</head>
<body>
  <div class="left">Left Content</div>
  <div class="right">Right Content</div>
</body>
</html>
```

---

43. Responsive Image (CSS Max-Width)

Goal: Teach how to make images scale before Bootstrap's `.img-fluid`.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    img {
      max-width: 100%;
      height: auto;
    }
  </style>
</head>
<body>
  <img src="https://via.placeholder.com/600x300" alt="Sample">
</body>
</html>
```

---

44. Navigation Bar (Basic)

Goal: Create a horizontal nav menu (before Bootstrap's `.navbar`).

```
<!DOCTYPE html>
<html>
<head>
  <style>
    ul {
      list-style-type: none;
      background: #333;
      padding: 0;
      margin: 0;
    }
    li {
      display: inline;
      margin-right: 20px;
    }
    a {
      color: white;
      text-decoration: none;
    }
  </style>
</head>
<body>
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Contact</a></li>
  </ul>
</body>
</html>
```

45. Styled Button with Hover Effect

Goal: Before Bootstrap buttons.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .btn {
      background: green;
      color: white;
      padding: 10px 20px;
      border: none;
      cursor: pointer;
    }
    .btn:hover {
      background: darkgreen;
    }
  </style>
</head>
<body>
  <button class="btn">Click Me</button>
</body>
</html>
```

---

46. Simple Form Styling

Goal: Teach form layout before Bootstrap's `.form-control`.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    form {
      width: 300px;
      margin: 20px auto;
    }
    input, textarea {
      width: 100%;
      margin-bottom: 10px;
      padding: 8px;
    }
  </style>
</head>
<body>
  <form>
    <input type="text" placeholder="Name">
    <input type="email" placeholder="Email">
    <textarea placeholder="Message"></textarea>
    <button>Submit</button>
  </form>
</body>
</html>
```

47. Card Layout (Manual)

Goal: Practice before Bootstrap `.card`.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .card {
      border: 1px solid #ccc;
      border-radius: 5px;
      padding: 20px;
      width: 250px;
      margin: 10px auto;
      box-shadow: 2px 2px 8px rgba(0,0,0,0.1);
    }
  </style>
</head>
<body>
  <div class="card">
    <h3>Card Title</h3>
    <p>Some card content.</p>
  </div>
</body>
</html>
```

48. Modal Simulation (Hide/Show with CSS)

Goal: Before Bootstrap’s `.modal`.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .modal {
      display: none;
      position: fixed;
      top: 50%;
      left: 50%;
      transform: translate(-50%, -50%);
      background: white;
      padding: 20px;
      border: 1px solid black;
    }
    .modal.show {
      display: block;
    }
  </style>
</head>
<body>
  <button
onclick="document.getElementById('myModal').classList.add('show')">Ope
n Modal</button>
  <div id="myModal" class="modal">
    <p>This is a modal!</p>
  </div>
</body>
</html>
```

49. Grid Layout Using CSS Flexbox

Goal: Introduce flexbox before Bootstrap grid system.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .row {
      display: flex;
    }
    .col {
      flex: 1;
      margin: 5px;
      padding: 20px;
      background: lightblue;
    }
  </style>
</head>
<body>
  <div class="row">
    <div class="col">Column 1</div>
    <div class="col">Column 2</div>
    <div class="col">Column 3</div>
  </div>
</body>
</html>
```

50. Simple Dropdown Menu (CSS Hover)

Goal: Before Bootstrap's `.dropdown`.

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

```
  <style>
    .dropdown {
      position: relative;
      display: inline-block;
    }
    .dropdown-content {
      display: none;
      position: absolute;
      background: white;
      border: 1px solid #ccc;
      min-width: 120px;
    }
    .dropdown:hover .dropdown-content {
      display: block;
    }
  </style>
</head>
<body>
  <div class="dropdown">
    <button>Menu</button>
    <div class="dropdown-content">
      <p>Option 1</p>
      <p>Option 2</p>
    </div>
  </div>
</body>
</html>
```

51. Centering Content with Flexbox

Goal: Learn how to center elements before Bootstrap's `.d-flex .justify-content-center`.

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

```
  <style>
    .center {
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }
  </style>
</head>
<body>
  <div class="center">
    <h1>Centered Text</h1>
  </div>
</body>
</html>
```

52. Sticky Header

Goal: Understand `position: sticky` before Bootstrap's sticky nav.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    header {
      position: sticky;
      top: 0;
      background: lightblue;
      padding: 15px;
    }
  </style>
</head>
<body>
  <header>Sticky Header</header>
  <p>Scroll down...</p>
  <div style="height:1500px;"></div>
```

```
</body>
</html>
```

---

53. Fixed Footer

Goal: Create a fixed footer like Bootstrap's `.fixed-bottom`.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    footer {
      position: fixed;
      bottom: 0;
      width: 100%;
      background: gray;
      color: white;
      text-align: center;
      padding: 10px;
    }
  </style>
</head>
<body>
  <p>Page content here...</p>
  <footer>Fixed Footer</footer>
</body>
</html>
```

---

54. Responsive Columns (Percentage Widths)

Goal: Teach column layout before Bootstrap grid system.

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

```
  <style>
    .col {
      float: left;
      width: 33.3%;
      padding: 20px;
      box-sizing: border-box;
    }
  </style>
</head>
<body>
  <div class="col">Column 1</div>
  <div class="col">Column 2</div>
  <div class="col">Column 3</div>
</body>
</html>
```

55. Image Gallery with Flexbox

Goal: Build a grid-like gallery before Bootstrap `.row` & `.col`.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .gallery {
      display: flex;
      flex-wrap: wrap;
    }
    .gallery img {
      width: 30%;
      margin: 1%;
    }
  </style>
</head>
<body>
  <div class="gallery">
```

```
    <img src="https://via.placeholder.com/150">
    <img src="https://via.placeholder.com/150">
    <img src="https://via.placeholder.com/150">
  </div>
</body>
</html>
```

56. Profile Card Layout

Goal: Prepare for Bootstrap's `.card`.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .card {
      width: 250px;
      border: 1px solid #ccc;
      border-radius: 5px;
      padding: 15px;
      text-align: center;
    }
    .card img {
      width: 100px;
      border-radius: 50%;
    }
  </style>
</head>
<body>
  <div class="card">
    <img src="https://via.placeholder.com/100">
    <h3>John Doe</h3>
    <p>Web Developer</p>
  </div>
</body>
</html>
```

57. Simple Banner Section

Goal: Create a hero section before Bootstrap `.jumbotron`.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .banner {
      background: lightblue;
      padding: 50px;
      text-align: center;
    }
  </style>
</head>
<body>
  <div class="banner">
    <h1>Welcome to My Website</h1>
    <p>This is a hero banner</p>
  </div>
</body>
</html>
```

58. Responsive Navbar with Flexbox

Goal: Practice navbar layout before Bootstrap `.navbar`.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    nav {
      display: flex;
      justify-content: space-between;
      background: #333;
      padding: 10px;
    }
    nav a {
      color: white;
      margin: 0 10px;
      text-decoration: none;
    }
  </style>
</head>
<body>
  <nav>
    <div>Logo</div>
    <div>
      <a href="#">Home</a>
      <a href="#">About</a>
    </div>
  </nav>
</body>
</html>
```

59. Pricing Table Layout

Goal: Teach students table-like card layout.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .pricing {
      display: flex;
      justify-content: center;
    }
    .plan {
      border: 1px solid #ccc;
      padding: 20px;
      margin: 10px;
      width: 200px;
      text-align: center;
    }
  </style>
</head>
<body>
  <div class="pricing">
    <div class="plan">Basic</div>
    <div class="plan">Standard</div>
    <div class="plan">Premium</div>
  </div>
</body>
</html>
```

60. Simple Sidebar Layout

Goal: Before Bootstrap `.offcanvas`.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .sidebar {
      width: 200px;
      float: left;
      background: lightgray;
      height: 100vh;
      padding: 20px;
    }
    .content {
      margin-left: 220px;
      padding: 20px;
    }
  </style>
</head>
<body>
  <div class="sidebar">Sidebar</div>
  <div class="content">Main Content</div>
</body>
```

```
</html>
```

Building a Login Form

Exercise 1: Create the Basic HTML Form

Explanation: Start with a simple `<form>` element that will hold inputs for username and password.

Code:

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

  <label>Password:</label>
  <input type="password" name="password">

  <button type="submit">Login</button>
</form>
```

Exercise 2: Add `for` and `id` Attributes for Accessibility

Explanation: Linking `<label>` with `<input>` using `for` and `id` improves accessibility.

Code:

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

  <label for="password">Password:</label>
  <input type="password" id="password" name="password">
  <button type="submit">Login</button>
</form>
```

Exercise 3: Add Placeholder Text

Explanation: Placeholders give users hints inside the input fields.

Code:

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

  <label for="password">Password:</label>
  <input type="password" id="password" name="password"
placeholder="Enter your password">

  <button type="submit">Login</button>
</form>
```

Exercise 4: Group Form Elements Inside a Container

Explanation: Wrapping the form in a `<div>` (like `.login-box`) allows us to style it later.

Code:

```
<div class="login-box">
  <form>
    <label for="username">Username:</label>
    <input type="text" id="username" name="username"
placeholder="Enter your username">

    <label for="password">Password:</label>
    <input type="password" id="password" name="password"
placeholder="Enter your password">
```

```
    <button type="submit">Login</button>
  </form>
</div>
```

Exercise 5: Add Basic CSS for Inputs

Explanation: Style inputs for readability.

Code:

```
input {
  display: block;
  width: 100%;
  padding: 8px;
  margin: 8px 0;
}
```

Exercise 6: Style the Button

Explanation: Add background color and hover effect to the login button.

Code:

```
button {
  background-color: blue;
  color: white;
  padding: 10px;
  width: 100%;
  border: none;
  cursor: pointer;
}
button:hover {
  background-color: darkblue;
}
```

Exercise 7: Center the Login Box

Explanation: Center the form on the page using CSS Flexbox.

Code:

```
body {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  background-color: #f2f2f2;
}
.login-box {
  width: 300px;
  padding: 20px;
  background: white;
  border-radius: 8px;
  box-shadow: 0 0 10px rgba(0,0,0,0.1);
}
```

Exercise 8: Add a Title

Explanation: Adding a heading helps guide the user.

Code:

```
<div class="login-box">
  <h2>Login</h2>
  <form>
    <!-- inputs and button here -->
  </form>
</div>
```

CSS:

```
h2 {
  text-align: center;
  margin-bottom: 20px;
}
```

Exercise 9: Add “Remember Me” Checkbox

Explanation: Give the user an option to stay logged in.

Code:

```
<div class="login-box">
  <h2>Login</h2>
  <form>
    <label for="username">Username:</label>
    <input type="text" id="username" name="username"
placeholder="Enter your username">

    <label for="password">Password:</label>
    <input type="password" id="password" name="password"
placeholder="Enter your password">

    <label>
      <input type="checkbox" name="remember"> Remember Me
    </label>

    <button type="submit">Login</button>
  </form>
</div>
```

Exercise 10: Add “Forgot Password?” Link

Explanation: Provide users with a link in case they forget their password.

Code:

```
<div class="login-box">
  <h2>Login</h2>
  <form>
    <label for="username">Username:</label>
    <input type="text" id="username" name="username"
placeholder="Enter your username">

    <label for="password">Password:</label>
    <input type="password" id="password" name="password"
placeholder="Enter your password">

    <label>
      <input type="checkbox" name="remember"> Remember Me
    </label>

    <button type="submit">Login</button>
    <p style="text-align:center; margin-top:10px;">
      <a href="#">Forgot Password?</a>
    </p>
  </form>
</div>
```

Introduction to JavaScript & jQuery

JavaScript

JavaScript is a programming language that makes web pages interactive. While **HTML** structures the content and **CSS** styles it, **JavaScript** adds behavior (e.g., animations, form validation, dynamic updates without refreshing the page).

jQuery

jQuery is a **JavaScript library** created in 2006 that simplifies common JavaScript tasks. It makes code shorter and easier to write, especially for selecting elements, handling events, and animations.
👉 Example: Instead of writing `document.getElementById("myDiv").style.color = "red";` you can simply write `$("#myDiv").css("color", "red");`.

Steps to Integrate jQuery

1. **Download or CDN**
    - You can either **download jQuery** from https://jquery.com/ and include it in your project.
    - Or use a **CDN (Content Delivery Network)** link (recommended for beginners).

Include in `<head>` or before `</body>`
Add this line to your HTML:

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
```

2. **Write jQuery code inside `$(document).ready()`**
   This ensures the HTML loads first before running the code.

```
    $(document).ready(function(){
  // your jQuery code here
    });
```

Exercises

Exercise 1: Change Text of an Element

Goal: Change the text of a `<p>` element.

Native JS:

```
document.getElementById("demo").innerText = "Hello JavaScript!";
```

jQuery:

```
$("#demo").text("Hello jQuery!");
```

Exercise 2: Change Background Color

Goal: Change background color of a `<div>`.

Native JS:

```
document.getElementById("box").style.backgroundColor = "lightblue";
```

jQuery:

```
$("#box").css("background-color", "lightblue");
```

Exercise 3: Hide an Element

Goal: Hide a paragraph when a button is clicked.

Native JS:

```
document.getElementById("btn").onclick = function() {
  document.getElementById("demo").style.display = "none";
};
```

jQuery:

```
$("#btn").click(function(){
  $("#demo").hide();
});
```

Exercise 4: Show an Element

Goal: Show a hidden element.

Native JS:

```
document.getElementById("btn").onclick = function() {
  document.getElementById("demo").style.display = "block";
};
```

jQuery:

```
$("#btn").click(function(){
  $("#demo").show();
});
```

Exercise 5: Toggle Visibility

Goal: Toggle visibility on button click.

Native JS:

```
document.getElementById("btn").onclick = function() {
  let el = document.getElementById("demo");
  if(el.style.display === "none") {
    el.style.display = "block";
  } else {
    el.style.display = "none";
  }
};
```

jQuery:

```
$("#btn").click(function(){
  $("#demo").toggle();
});
```

Exercise 6: Add a Class

Goal: Add a CSS class to an element.

Native JS:

```
document.getElementById("demo").classList.add("highlight");
```

jQuery:

```
$("#demo").addClass("highlight");
```

Exercise 7: Remove a Class

Goal: Remove a CSS class.

Native JS:

```
document.getElementById("demo").classList.remove("highlight");
```

jQuery:

```
$("#demo").removeClass("highlight");
```

Exercise 8: Toggle a Class

Goal: Toggle a CSS class on click.

Native JS:

```
document.getElementById("btn").onclick = function() {
  document.getElementById("demo").classList.toggle("highlight");
};
```

jQuery:

```
$("#btn").click(function(){
  $("#demo").toggleClass("highlight");
});
```

Exercise 9: Change Image Source

Goal: Replace an image with another.

Native JS:

```
document.getElementById("myImg").src = "new.jpg";
```

jQuery:

```
$("#myImg").attr("src", "new.jpg");
```

Exercise 10: Get Input Value

Goal: Get the text typed into an input.

Native JS:

```
let val = document.getElementById("name").value;
console.log(val);
```

jQuery:

```
let val = $("#name").val();
console.log(val);
```

Exercise 11: Set Input Value

Goal: Pre-fill an input with text.

Native JS:

```
document.getElementById("name").value = "John Doe";
```

jQuery:

```
$("#name").val("John Doe");
```

Exercise 12: Append Content

Goal: Add new text at the end of a paragraph.

Native JS:

```
document.getElementById("demo").innerHTML += " Extra text!";
```

jQuery:

```
$("#demo").append(" Extra text!");
```

Exercise 13: Prepend Content

Goal: Add new text at the beginning.

Native JS:

```
document.getElementById("demo").innerHTML = "Start " +
document.getElementById("demo").innerHTML;
```

---

jQuery:

```
$("#demo").prepend("Start ");
```

Exercise 14: Fade Out Element

Goal: Smoothly fade out a div.

Native JS (CSS + JS):

```
document.getElementById("box").style.transition = "opacity 1s";
document.getElementById("box").style.opacity = 0;
```

jQuery:

```
$("#box").fadeOut();
```

Exercise 15: Fade In Element

Goal: Smoothly fade in a div.

Native JS (CSS + JS):

```
let box = document.getElementById("box");
box.style.transition = "opacity 1s";
box.style.opacity = 1;
```

jQuery:

```
$("#box").fadeIn();
```

Exercise 16: Slide Up

Goal: Slide up a panel.

Native JS (needs CSS animation):

```
document.getElementById("panel").style.maxHeight = "0px";
```

jQuery:

```
$("#panel").slideUp();
```

Exercise 17: Slide Down

Goal: Slide down a panel.

Native JS:

```
document.getElementById("panel").style.maxHeight = "200px";
```

jQuery:

```
$("#panel").slideDown();
```

Exercise 18: Change Multiple CSS Properties

Goal: Set color and background.

Native JS:

```
let el = document.getElementById("demo");
el.style.color = "white";
el.style.backgroundColor = "black";
```

jQuery:

```
$("#demo").css({
  "color": "white",
  "background-color": "black"
});
```

Exercise 19: Mouseover Event

Goal: Change text color on hover.

Native JS:

```
document.getElementById("demo").onmouseover = function() {
  this.style.color = "red";
};
```

jQuery:

```
$("#demo").mouseover(function(){
  $(this).css("color", "red");
});
```

Exercise 20: Click Event to Change Text

Goal: Change text when clicked.

Native JS:

```
document.getElementById("demo").onclick = function() {
  this.innerText = "You clicked me!";
};
```

jQuery:

```
$("#demo").click(function(){
  $(this).text("You clicked me!");
});
```

Introduction to Bootstrap

Bootstrap is a **front-end framework** used to design modern, responsive, and mobile-first websites quickly.
It provides **ready-made CSS styles** and **JavaScript components** so developers don't have to write everything from scratch.

What Bootstrap is Used For:

1. **Responsive Layouts** – Your site adjusts automatically for desktop, tablet, and mobile screens.
2. **Pre-designed Components** – Buttons, forms, navbars, cards, modals, etc.
3. **Grid System** – A 12-column layout that makes arranging elements easier.
4. **Built-in Utilities** – Quick classes for colors, spacing, text alignment, etc.
5. **JavaScript Plugins** – Dropdowns, carousels, modals, tooltips (powered by JS/jQuery).

Steps to Integrate Bootstrap

There are two ways to use Bootstrap:

1. Using CDN (Quick & Recommended)

Add this inside the `<head>` of your HTML:

```
<!-- Bootstrap CSS -->
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.
min.css" rel="stylesheet">

<!-- Bootstrap JS + Popper.js (for components like dropdowns,
tooltips) -->
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/poppe
r.min.js"></script>
```

```
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.mi
n.js"></script>
```

2. Download and Host Locally

- Download from https://getbootstrap.com/
- Link the CSS and JS files in your project just like normal stylesheets and scripts.

A Simple Bootstrap Login Form

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Bootstrap Login</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.
min.css" rel="stylesheet">
</head>
<body class="bg-light d-flex justify-content-center align-items-center
vh-100">

  <div class="card shadow p-4" style="width: 350px;">
    <h3 class="text-center mb-4">Login</h3>
    <form>
      <div class="mb-3">
        <label class="form-label">Email</label>
        <input type="email" class="form-control" placeholder="Enter
your email">
      </div>
      <div class="mb-3">
        <label class="form-label">Password</label>
        <input type="password" class="form-control" placeholder="Enter
your password">
      </div>
      <button type="submit" class="btn btn-primary
w-100">Login</button>
    </form>
  </div>

</body>
</html>
```

The Bootstrap Grid System

The **Grid System** is Bootstrap's layout engine. It uses **rows** and **columns** inside a **container** to align content.

⚙️ How it Works:

1. **Container** → The wrapper that holds everything.
   - `.container` (fixed width, adjusts at breakpoints)
   - `.container-fluid` (always 100% width)
2. **Row** → Creates a horizontal group of columns.
   - Must be placed inside a container.
3. **Columns** → Divide the row into sections using `.col`.
   - Bootstrap's grid has **12 columns** per row.
   - You can split them in any way (e.g., 6+6, 4+4+4, 3+3+3+3, etc.).

Example 1: Equal Columns

```
<div class="container">
  <div class="row">
```

```
    <div class="col bg-primary text-white">Column 1</div>
    <div class="col bg-success text-white">Column 2</div>
    <div class="col bg-danger text-white">Column 3</div>
  </div>
</div>
```

➡️ Each `.col` automatically divides space **equally**.

Example 2: Fixed Column Sizes

```
<div class="container">
  <div class="row">
    <div class="col-4 bg-primary text-white">Column 1 (4/12)</div>
    <div class="col-8 bg-success text-white">Column 2 (8/12)</div>
  </div>
</div>
```

➡️ Here, **Column 1 takes 4 parts**, Column 2 takes **8 parts** = total **12 columns**.

Responsive Grid Breakpoints

Bootstrap’s grid is **responsive**. You can define column sizes for different screen widths:

- `col-` → Extra small (mobile, <576px)
- `col-sm-` → Small (≥576px)
- `col-md-` → Medium (≥768px)
- `col-lg-` → Large (≥992px)
- `col-xl-` → Extra large (≥1200px)

- `col-xxl-` → Extra extra large (≥1400px)

Example 3: Responsive Grid

```
<div class="container">
  <div class="row">
    <div class="col-12 col-md-6 col-lg-4 bg-primary text-white">Box
1</div>
    <div class="col-12 col-md-6 col-lg-4 bg-success text-white">Box
2</div>
    <div class="col-12 col-md-12 col-lg-4 bg-danger text-white">Box
3</div>
  </div>
</div>
```

- On **mobile (<576px)** → Each column takes full width (stacked).
- On **tablet (≥768px)** → First two share the row, third one below.
- On **desktop (≥992px)** → All three fit in one row equally.