Okay, this is a solid rubric! It's very clear about what's expected. Let's break this down piece by piece so you know exactly what to focus on for maximum points.

Here's my take on how to approach each requirement:

📁 Folder Structure & HTML5 Standards (10 points)

- **Requirement:** Well-organized folders (`assets`, `css`, `images`, etc.), HTML files in the root, and HTML5 standard formatting.
- **My Advice:**
    - Create a main project folder. Inside it, immediately create:
        - `index.html` (and other HTML files like `about.html`, `contact.html`, etc.) directly in this root folder.
        - `css/` folder (for your `style.css` or similar).
        - `js/` folder (for your JavaScript files, if you add custom JS beyond jQuery for specific tasks).
        - `images/` folder (for all your `.jpg`, `.png`, `.svg` files).
        - You might also have an `assets/` folder if you have other types of files like downloadable documents, but `images` is the key one mentioned.
    - Ensure every HTML page starts with `<!DOCTYPE html>` and includes the `<html>`, `<head>`, and `<body>` tags correctly. Use meaningful `<title>` tags for each page.
    - Use proper indentation for your HTML code. It makes it much easier to read and debug.

🧱 HTML5 Semantics & Navigation (10 points)

- **Requirement:** Use HTML5 sectioning elements (`header`, `nav`, `section`, `footer`, `aside`, `address`), inline semantic elements, a `ul`-based navigation with at least 5 links, and HTML comments.
- **My Advice:**
    - **Structure:**
        - `<header>`: For your site title/logo and main navigation.
        - `<nav>`: Specifically for your main navigation menu (likely inside the `<header>` or directly below it).

        - `<main>`: This should wrap the primary content of each page. While not explicitly listed, it's a crucial semantic element. Inside `<main>`, you'll use `<section>` for distinct content blocks.
        - `<section>`: For grouping related content. You can have multiple sections per page.
        - `<footer>`: For copyright information, contact links, etc.
        - `<aside>`: For content tangentially related to the main content (e.g., a sidebar with links, ads, or related information).
        - `<address>`: Use this for contact information (could be in the footer or on a contact page).

Navigation: Your `nav` element should contain an `<ul>`, and each `<li>` within that `<ul>` will hold an `<a>` tag for your links. Example:
HTML
<nav>

Home

        ○
    -

About Us Services Portfolio Contact </nav> ``` * **Inline Semantics:** Use `<em>` for emphasis, `<strong>` for importance, `<time>` for dates/times, etc., where appropriate. * **Comments:** Add comments like `...` or `` to help your instructor (and yourself!) navigate the code.

📄 Web Pages & Links (20 points)

- **Requirement:** 5-10 web pages, seamless linking (no broken links), at least 1 `mailto:` link, and at least 3 images (not CSS backgrounds).
- **My Advice:**
    - **Page Count:** Plan out your site. Common pages include: Home, About, Services/Products, Portfolio/Gallery, Blog (even if just a placeholder), Contact. This easily gets you to 5+ pages.
    - **Linking:** Double-check all `href` attributes. For internal links, use relative paths (e.g., `about.html`, `../css/style.css`). Test every single link!

`mailto:` Link: Easy to add, often in the footer or on the contact page:
HTML
Email Us

-

Images: Use the `<img>` tag with descriptive `alt` attributes for accessibility.
HTML
<img src="images/my-image.jpg" alt="Description of my image">

- These must be actual images in your content, not background images applied via CSS.

✒️ Google Web Font (10 points)

- **Requirement:** Add a Google Web Font. No web-safe fonts (Arial, Times New Roman, etc.).
- **My Advice:**
  - Go to fonts.google.com.
  - Choose a font (or a couple for headings and body text).
  - Select the styles you need (e.g., Regular 400, Bold 700).
  - Google Fonts will provide you with a `<link>` tag to put in the `<head>` of your HTML documents and the CSS `font-family` rules to use in your stylesheet. <!-- end list -->

Example (in `<head>`):
HTML
<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">

-

Example (in `style.css`):
CSS
body {

font-family: 'Roboto', sans-serif;

}

-

🎨 CSS Implementation (25 points)

- **Requirement:** External CSS file, linked to all pages. No embedded (`<style>`) or inline (`style=""`) styles. Use 7+ selector types. CSS comments.
- **My Advice:**
    - Create `css/style.css` (or a similar name).

In the `<head>` of *every* HTML page, link it:
HTML
<link rel="stylesheet" href="css/style.css">

    -
    - **Selector Types (Examples):**
        1. **Type:** `p { color: blue; }`
        2. **Class:** `.my-class { font-weight: bold; }`
        3. **ID:** `#unique-element { border: 1px solid black; }`
        4. **Descendant:** `nav ul li { list-style-type: none; }`
        5. **Pseudo-class:** `a:hover { text-decoration: underline; }` or `input:focus { border-color: green; }`
        6. **Adjacent Sibling:** `h1 + p { margin-top: 0; }` (selects `p` only if it immediately follows `h1`)
        7. **General Sibling:** `h1 ~ p { color: gray; }` (selects any `p` that is a sibling of `h1` and comes after it)
        8. **Child:** `ul > li { padding-left: 10px; }` (selects `li` only if it's a direct child of `ul`)
        9. **Attribute:** `input[type="text"] { width: 200px; }`
    - You'll easily hit 7 types by just styling your site naturally.
    - **Comments:** Similar to HTML, comment your CSS sections: `/* --- Navigation Styles --- */`, `/* --- Footer Styles --- */`.

📐 CSS Structure (Box Model/Flexbox/Grid) (15 points)

- **Requirement:** Use box model properties (width, height, padding, margin, border) OR Flexbox/Grid for structure. Correct selector use.
- **My Advice:**

Box Model: You'll be using this constantly. Remember `box-sizing: border-box;` is your friend! Add it to your global reset:

```
CSS
*, *::before, *::after {

  box-sizing: border-box;

}
```

-
- **Flexbox/Grid:** Highly recommended for layout, especially for responsiveness.
  - **Flexbox** is great for arranging items in a single dimension (a row or a column), like navigation items, aligning items in a card, etc.
  - **CSS Grid** is excellent for two-dimensional layouts (rows AND columns), like the overall page structure.
- The rubric says "instead of the box model if you choose," but you'll use box model properties *even with* Flexbox/Grid to control spacing and sizing of the flex/grid items themselves. The "instead of" likely refers to *not* using older float-based layouts.
- Focus on using these tools to define the main structure of your pages and components.

✨ CSS Formatting (10 points)

- **Requirement:** Use CSS for visual appeal: background colors/images, font weights, line heights, colors, etc.
- **My Advice:**
  - This is where your site comes to life!
  - Choose a color palette (2-3 main colors, a few accent colors).
  - Ensure good contrast between text and background for readability.
  - Use `line-height` for comfortable reading (e.g., `1.6` for body text).
  - Apply `font-weight` for emphasis (e.g., `bold` for headings).
  - Consider subtle `background-image` or `background-color` changes for different sections.

📝 Contact Form (10 points)

- **Requirement:** A very specific list of form controls, HTML5 validation, CSS styling for valid/invalid states, and `label` elements.

- **My Advice:** This is a checklist within a checklist. Go through each item:
    - `<form>`
        - `<fieldset>` (group related controls)
            - `<legend>` (caption for the fieldset)
        -
        - **Text Boxes (at least 4):**
            - `<label for="name">Name:</label><input type="text" id="name" name="name" required>`
            - `<label for="email">Email:</label><input type="email" id="email" name="email" required pattern="[a-z0-9._%+\-]+@[a-z0-9.\-]+\.[a-z]{2,}$">` (example pattern)
            - `<label for="subject">Subject:</label><input type="text" id="subject" name="subject">`
            - `<label for="phone">Phone (optional):</label><input type="tel" id="phone" name="phone">`
        -

Radio Buttons (at least 1 set): (Only one can be selected)
HTML
```
<fieldset>

  <legend>Preferred Contact Method:</legend>

  <label><input type="radio" name="contact_method" value="email" checked> Email</label>

  <label><input type="radio" name="contact_method" value="phone"> Phone</label>

</fieldset>
```

-

Checkboxes (at least 1 set): (Multiple can be selected)
HTML
```
<fieldset>

  <legend>Interests:</legend>

  <label><input type="checkbox" name="interests" value="design"> Design</label>

  <label><input type="checkbox" name="interests" value="development"> Development</label>

  <label><input type="checkbox" name="interests" value="ux"> UX</label>
```

</fieldset>

- **Dropdown Menu (at least 1):**
  - <label for="query_type">Reason for Contact:</label> <select id="query_type" name="query_type" required> <option value="">--Please choose an option--</option> <option value="general">General Inquiry</option> <option value="support">Support</option> <option value="feedback">Feedback</option> </select> ```

List Box (at least 1): (This is typically a `<select>` with the `multiple` attribute or a `size` attribute greater than 1)
HTML
<label for="preferred_services">Preferred Services (select multiple):</label>

<select id="preferred_services" name="preferred_services" multiple size="3">

<option value="web_design">Web Design</option>

<option value="development">Web Development</option>

<option value="seo">SEO</option>

<option value="consulting">Consulting</option>

</select>

Multiline Textarea (at least 1):
HTML
<label for="message">Message:</label>

<textarea id="message" name="message" rows="5" required></textarea>

Submit Button:
HTML
<button type="submit">Send Message</button>

- **HTML5 Constraint Validation:** Use `required` attribute for mandatory fields. Use `type` attributes like `email`, `tel`, `url`, `number` for built-in validation. Use the `pattern` attribute for custom validation (e.g., specific phone number format).

CSS3 Selectors for Styling:

CSS

```
input:required {

  border-left: 3px solid orange; /* Just an example */

}

input:valid {

  border-left: 3px solid green;

}

input:invalid {

  border-left: 3px solid red;

}

/* You can also use :user-invalid after user interaction */
```

-
- **`label` elements:** Crucial for accessibility. Ensure the `for` attribute of the label matches the `id` of the form control.

📱 Responsiveness (40 points)

- **Requirement:** Fully functional on desktop, tablet, and smartphone. Use media queries OR Bootstrap/Skeleton grid system. **If Bootstrap, ONLY the grid system is evaluated; other classes must be custom.**
- **My Advice:** This is the heaviest weighted section, so give it attention!
  - **Mobile-First Approach (Recommended):** Design for small screens first, then use media queries to adapt the layout for larger screens.

Media Queries:

CSS

```
/* Base styles (mobile) */

.container { width: 95%; margin: 0 auto; }

nav ul { flex-direction: column; } /* Example */
```

```
/* Tablet styles */
@media (min-width: 768px) {
  .container { width: 90%; }
  nav ul { flex-direction: row; } /* Example */
}

/* Desktop styles */
@media (min-width: 1024px) {
  .container { width: 80%; max-width: 1200px; }
}
```

-
- **Bootstrap/Skeleton Grid:**
  - **Bootstrap:** If you use Bootstrap, include the CSS for the grid system. Then, for layout, use classes like `.container`, `.row`, `.col-md-6`, `.col-lg-4`, etc.
    - **VERY IMPORTANT:** The rubric states: "If you do decide to use Bootstrap, only the grid system will be evaluated. Any other classes should be your own and not point to existing Bootstrap helper classes."
      - This means **NO** Bootstrap buttons (`.btn`, `.btn-primary`), NO Bootstrap cards (`.card`), NO Bootstrap navbars (`.navbar`), etc., for styling. You must write your own CSS for these components. You can use Bootstrap's grid for the layout of these components, but their appearance should come from *your* `style.css`.
  - **Skeleton:** Skeleton is a much simpler boilerplate. If you choose this, you'll be writing more custom CSS by default, which fits the spirit of the requirement well.
- **Testing:** Constantly test your site by resizing your browser window. Use your browser's developer tools (responsive design mode) to emulate different devices.