HTML5_Day3_Hands_On_Koduru_Ushasree

Problem 1

Assessment Goal: Check ability to apply styling and understand selectors and box model.

Hands-on Tasks:

1. Create an external CSS file and link it to HTML
2. Apply different text styles to headings and paragraphs
3. Add background color and borders to a section
4. Demonstrate margin and padding on a div
5. Style the form created on Day 2

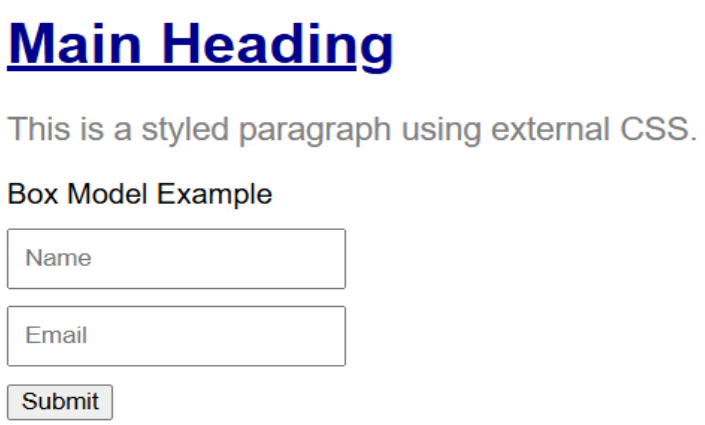
Expected Outcome:
A visually styled webpage using external CSS and proper selectors**.**

Code :-

```
p1 > index.html > html
<!DOCTYPE html>
<html>
<head>
    <title>Day 3 Styling</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>

    <section class="content">
        <h1>Main Heading</h1>
        <p>This is a styled paragraph using external CSS.</p>
    </section>

    <div class="box">
        Box Model Example
    </div>

    <form class="form-style">
        <input type="text" placeholder="Name">
        <input type="email" placeholder="Email">
        <button>Submit</button>
    </form>

</body>
</html>
```

```
p1 > # style.css > .form-style input
   1   body {
   2       font-family: Arial, sans-serif;
   3   }
   4
   5   h1 {
   6       color: darkblue;
   7       text-decoration: underline;
   8   }
   9
  10   p {
  11       font-size: 18px;
  12       color: gray;
  13   }
  14
  15   .form-style input {
  16       display: block;
  17       margin: 10px 0;
  18       padding: 8px;
  19   }
```

Output :-



Technical Requirements :-

This task must use an external CSS file linked to the HTML using the <link> tag. Styling should be applied using proper CSS selectors such as element, class, or descendant selectors. Text styles must be applied to headings and paragraphs, and background color and borders should be added to a section element. The CSS Box Model must be demonstrated using margin and padding on a . The form created earlier must also be styled using external CSS. Inline CSS is not allowed.

Problem 2: Personal Profile Card (Level-1)

Scenario
You are building a simple personal profile card for a portfolio website.

The card will display a user's name, role, short bio, and contact button, styled purely using CSS.

Requirements

Create a profile card layout containing:

- Name (heading)
- Role/title
- Short description paragraph
- Contact button

Apply:

- Text formatting (alignment, decoration)
- Font styles (font-family, size, weight)
- Typography enhancements (line-height, letter-spacing)

Use the CSS Box Model:

- Padding for spacing inside the card
- Border to define the card boundary
- Margin to separate the card from the page edges

Add:

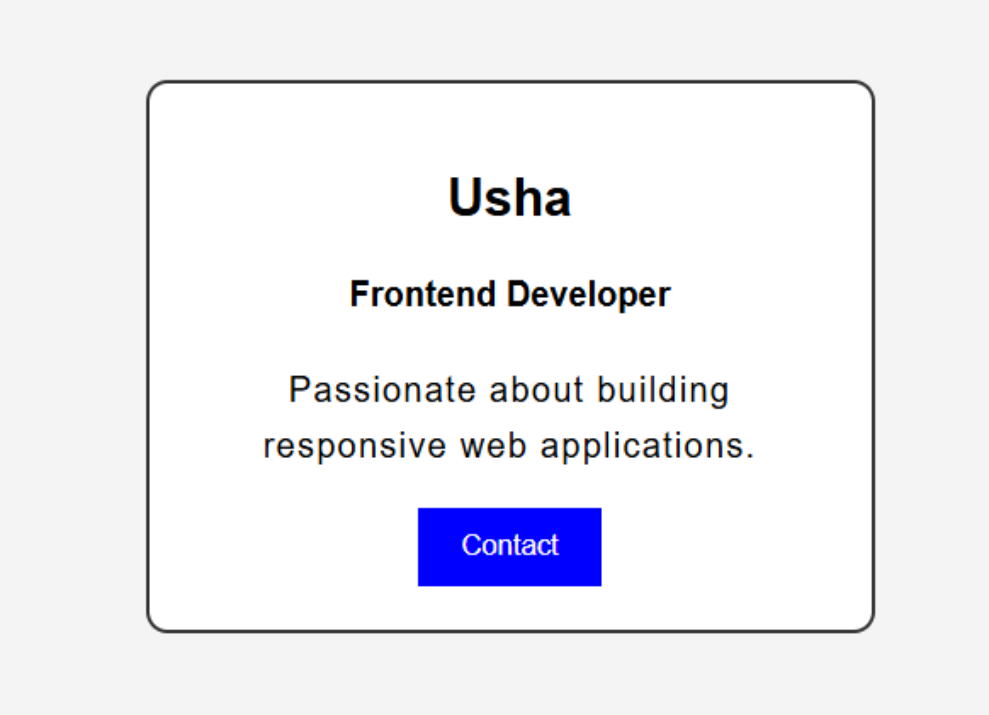
- Background color
- Border radius
- Hover effect on the button

Code :-

```
p2.html > html > body
<!DOCTYPE html>
<html>
<head>
<title>Profile Card</title>
<style>

body {
    font-family: Arial;
    background-color: #f4f4f4;
}
.card {
    width: 300px;
    margin: 50px auto;
    padding: 20px;
    border: 2px solid #333;
    border-radius: 10px;
    background-color: white;
    text-align: center;
}
.card h2 {
    font-size: 24px;
    font-weight: bold;
}
.card p {
    line-height: 1.6;
    letter-spacing: 1px;
}
button {
    padding: 10px 20px;
    border: none;
    background-color: blue;
    color: white;
    cursor: pointer;
}
button:hover {
    background-color: darkblue;
}
```

```
</style>
</head>
<body>

<div class="card">
    <h2>Usha</h2>
    <h4>Frontend Developer</h4>
    <p>Passionate about building responsive web applications.</p>
    <button>Contact</button>
</div>

</body>
</html>
```

Output:-



Technical Requirements :-

The profile card must be built using only HTML and CSS without any frameworks or inline styling. CSS should be written either inside a <style> tag or in an external file. Typography styling such as font-family, font-size, font-weight, line-height, and letter-spacing must be applied. The CSS Box Model should be used by adding padding inside the card, margin around it, and a border to define its boundary. Background color, border-radius, and a hover effect on the button must be included.

Problem 3: Product Feature List Section (Level-1)

Scenario

You are designing a feature highlight section for a product landing page that lists key features in a visually appealing way.

Requirements

Create a section with:

- Section heading
- List of 4–5 features

Style the list using:

- Custom fonts
- Text color and background color
- Border and padding for each feature item

Use:

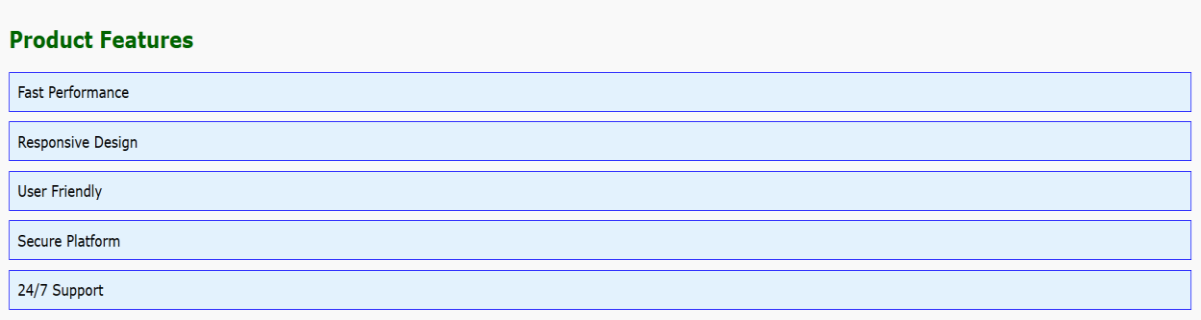
- CSS selectors (element, class, descendant)

Apply hover effects on feature items

Code :-

```
</head>
<body>

<section class="features">
    <h2>Product Features</h2>
    <ul>
        <li>Fast Performance</li>
        <li>Responsive Design</li>
        <li>User Friendly</li>
        <li>Secure Platform</li>
        <li>24/7 Support</li>
    </ul>
</section>

</body>
</html>
```

Output:-



Technical Requirements :-

The feature section must use basic HTML structure with an unordered list (<ul>). Styling should be applied using CSS selectors such as element selectors, class selectors, and descendant selectors. Each list item must have padding, borders, and background color. Hover effects must be applied to feature items using the :hover pseudo-class. No JavaScript or layout frameworks are allowed.

Problem 4: Responsive Blog Layout (Level-2)

Scenario

You are creating a blog homepage that displays articles in a modern responsive layout suitable for desktop, tablet, and mobile screens.

Requirements

Create a page layout with:

- Header
- Main content area with multiple blog cards

Use:

- **Flexbox or CSS Grid** for layout
- Box model for spacing

Each blog card should contain:

- Title
- Short description
- Read More button

Apply:

- Background colors
- Borders
- Typography enhancements

Use **media queries** to:

- Display multiple columns on desktop
- Switch to single column on mobile

Technical Constraints

- Only HTML + CSS
- No CSS frameworks
- No fixed widths (use percentages or flexible units)

Code :-

p4.html > html

```
<!DOCTYPE html>
<html>
<head>
<title>Responsive Blog</title>
<style>

body {
    font-family: Arial;
    margin: 0;
}

header {
    background-color: #333;
    color: white;
    padding: 20px;
    text-align: center;
}

.container {
    display: flex;
    flex-wrap: wrap;
    padding: 20px;
}

.card {
    background-color: #f4f4f4;
    border: 1px solid #ccc;
    padding: 15px;
    margin: 10px;
    flex: 1 1 30%;
}

.card h3 {
    margin-top: 0;
}

.card button {
```

```
@media (max-width: 768px) {
    .card {
        flex: 1 1 100%;
    }
}
</style>
</head>
<body>

<header>
    <h1>My Blog</h1>
</header>

<div class="container">
    <div class="card">
        <h3>Blog Title 1</h3>
        <p>Short description about blog post.</p>
        <button>Read More</button>
    </div>
<div class="card">
        <h3>Blog Title 2</h3>
        <p>Short description about blog post.</p>
        <button>Read More</button>
    </div>

    <div class="card">
        <h3>Blog Title 3</h3>
        <p>Short description about blog post.</p>
        <button>Read More</button>
    </div>
</div>
</body>
</html>
```

Output :-

Technical Requirements :-

The blog layout must be built using only HTML and CSS. Flexbox or CSS Grid must be used to create a responsive multi-column layout. Blog cards should use the box model for spacing, including padding, margin, and borders. Media queries must be implemented to display multiple columns on larger screens and switch to a single-column layout on smaller screens. Fixed widths are not allowed; flexible units such as percentages should be used.

Problem 5: Responsive Dashboard Layout (Level-2)

Scenario

You are designing a simple admin dashboard layout for a web application.

Requirements

Create a layout with:

- Sidebar navigation
- Main content area
- Info cards inside the main section

Use:

- display properties
- Flexbox or Grid for page structure

Style:

- Sidebar with background color
- Cards with borders, padding, and margins

Apply media queries to:

- Convert sidebar into top navigation on mobile
- Stack cards vertically on small screens

Code :-

p5.html > html > head > style

```
<!DOCTYPE html>
<html>
<head>
<title>Dashboard</title>
<style>
body {
    margin: 0;
    font-family: Arial;
}
.dashboard {
    display: flex;
    min-height: 100vh;
}
.sidebar {
    background-color: #333;
    color: white;
    padding: 20px;
    width: 200px;
}
.main {
    flex: 1;
    padding: 20px;
    display: flex;
    flex-wrap: wrap;
}
.card {
    background-color: #f4f4f4;
    border: 1px solid #ccc;
    padding: 20px;
    margin: 10px;
    flex: 1 1 30%;
}
@media (max-width: 768px) {
    .dashboard {
        flex-direction: column;
    }
```

```
        .sidebar {
            width: 100%;
        }

        .card {
            flex: 1 1 100%;
        }
    }
    </style>
    </head>
    <body>

    <div class="dashboard">
        <div class="sidebar">
            <h2>Menu</h2>
            <p>Dashboard</p>
            <p>Reports</p>
            <p>Settings</p>
        </div>

        <div class="main">
            <div class="card">Card 1</div>
            <div class="card">Card 2</div>
            <div class="card">Card 3</div>
        </div>

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

Technical Requirements :-

The dashboard layout must use HTML and CSS only, without JavaScript or external UI libraries. Flexbox or Grid must be used to structure the sidebar and main content area. The sidebar should have a background color, and cards inside the main section should include borders, padding, and margins. Media queries must convert the sidebar into a top navigation layout on smaller screens and stack cards vertically for responsiveness.

Output:-