Module 3

CSS Selectors and Styling

Theory Assignment :

Q.1: What is a CSS selector? Provide examples of element, class, and ID selectors.

A CSS selector is a pattern used to select and style specific HTML elements on a web page. It tells the browser *which elements* the CSS rules should apply to.

1) **Element Selector :**

- Selects all elements of a given type (HTML tag).
- **Syntax:**

```
 element {
        property : value;
   }
```

Ex : p {

```
  color: blue;
  font-size: 16px;
}
```

2) **Class Selector :**

- Selects elements with a specific class attribute.
- **Syntax:**

```
. classname {
  property: value;
}
```

Ex : .highlight {

```
  background-color: yellow;
  font-weight: bold;
}
```

3) **ID Selector :**

- Selects a single element with a specific **id** attribute.
- **Syntax :**

```
#idname {
  property: value;
}
```

Ex :
```
#main-title {
    color: green;
    text-align: center;
}
```

Q.2 : Explain the concept of CSS specificity. How do conflicts between multiple styles get resolved?

CSS specificity determines which style rule is applied when multiple rules target the same element.

In simple terms - specificity decides “who wins” when there’s a conflict between CSS rules.

How Specificity Works :

Each CSS selector has a, specificity works which is based on the types of selectors used.

Selector Type	Example	Specificity Value
Inline style	<p style="color : red;">	(1, 0, 0, 0)
Id selectors	#title {}	(0,1,0,0)
Class, attributes, pseudo-class selectors	.intro {}, [type="text"], :hove	(0, 0, 1, 0)
Element and pseudo-element selectors	p {}, ::before	(0, 0, 0, 1)
Universal selector	* {}	(0, 0, 0, 0)

Resolving Conflicts Between Multiple Styles,

When multiple rules apply to the same element:

1. **Calculate specificity** of each selector.
2. The rule with **higher specificity** takes precedence.
3. If specificity is **equal**, the **later rule** in the stylesheet wins.
4. If a style is marked **!important**, it overrides normal specificity rules (but can be overridden by another !important with higher specificity).

Q.3 : What is the difference between internal, external, and inline CSS? Discuss the advantages and disadvantages of each approach.

CSS (Cascading Style Sheets) can be added to an HTML document in three main ways - Inline, Internal, and External.

Types	Definition	How it is written	Applies to
Inline CSS	CSS written directly within an HTML tag's style attribute.	<p style="color:red;">Hello</p>	A single element
Internal CSS	CSS written inside a <style> tag within the <head> section of an HTML file.	<style> p { color: red; } </style>	The entire HTML page
External CSS	CSS written in a separate .css file and linked using <link> tag in the <head>.	<link rel="stylesheet" href="style.css">	Multiple Web Pages

1) **Inline CSS:**

Advantages:

- Quick and easy for small changes.
- Styles apply immediately to a specific element.
- No need for a separate CSS file.

Disadvantages:

- Difficult to maintain and debug.
- Not reusable - must be repeated for every element.
- Mixes content and design (violates separation of concerns).
- Increases page size if overused.

2) Internal CSS:

Advantages:

- Useful for styling a single page.
- Keeps design separate from content (better than inline).
- No need for external files - loads faster for small pages.

Disadvantages:

- Styles apply only to one page.
- Increases page size if many styles are added.
- Slower loading if used excessively on multiple pages.

3) External CSS :

Advantages :

- Styles can be applied to multiple pages (reusability).
- Cleaner HTML code (content and design separated).
- Reduces page size and improves website maintenance.
- Browser caching improves performance.

Disadvantages:

- Requires additional HTTP request to load the CSS file.
- If the CSS file fails to load, the webpage appears unstyled.

Lab Assignment:

Task:

Style the contact form (created in the HTML Forms lab) using external CSS. The following should be implemented:

- Change the background color of the form.
- Add padding and margins to form fields.
- Style the submit button with a hover effect.
- Use class selectors for styling common elements and ID selectors for unique elements.

Contact.html

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Contact Form</title>

   <link rel="stylesheet" href="styles.css">

</head>

<body>



   <h2>Contact Us</h2>



   <form id="contactForm">

      <label for="name">Name:</label>

      <input type="text" id="name" class="form-field" placeholder="Enter your
name">
```

```
        <label for="email">Email:</label>

        <input type="email" id="email" class="form-field" placeholder="Enter
your email">


        <label for="message">Message:</label>

        <textarea id="message" class="form-field" placeholder="Type your
message"></textarea>


        <button type="submit" id="submitBtn">Send Message</button>

    </form>

   </body>

  </html>
```

Style.css

```
#contactForm {

    background-color: #f2f7ff;

    width: 350px;

    padding: 20px;

    border-radius: 10px;

    margin: 20px auto;

    box-shadow: 0 0 10px rgba(0,0,0,0.1);

}
```

```
.form-field {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #aaa;
    border-radius: 5px;
    font-size: 14px;
}
#submitBtn {
    width: 100%;
    padding: 10px;
    margin-top: 10px;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 5px;
    font-size: 16px;
    cursor: pointer;
}
#submitBtn:hover {
    background-color: #0056b3;
}
```

CSS Box Model

Q.1: Explain the CSS box model and its components (content, padding, border, margin). How does each affect the size of an element?

```
+-----------------------------+
|          Margin             |
|  +-----------------------+  |
|  |        Border         |  |
|  |  +-----------------+  |  |
|  |  |     Padding     |  |  |
|  |  |  +-----------+  |  |  |
|  |  |  |  Content  |  |  |  |
|  |  |  +-----------+  |  |  |
|  |  +-----------------+  |  |
|  +-----------------------+  |
+-----------------------------+
```

1. Content

- The actual data inside the box (text, images, etc.).
- Controlled using width and height.
- Directly affects the element size

2. Padding

- Space between content and border.
- Creates "breathing room" inside the box.
- Example: padding: 10px;

Effect on size:

- Padding increases the total size of the element because it adds extra space around the content.
- Total width = width + left padding + right padding
  Total height = height + top padding + bottom padding

3. Border

- A line that surrounds the padding and content.
- Example: border: 2px solid black;

Effect on size:

Border also adds to the total size of the element.

Total width = width + padding + border
Total height = height + padding + border

4. Margin

- The space outside the border.
- Creates distance between this element and other elements.
- Example: margin: 20px;

Effect on size:

Margins do NOT increase the element's internal size,
but they increase the space it occupies on the page.

Q.2 : **What is the difference between border-box and content-box box-sizing in CSS? Which is the default?**

1. content-box

- This is the default box-sizing in CSS.
- The width and height you set apply only to the content area.
- Padding and border are added **outside** the specified width/height, increasing the total size.

Ex:
```
.box {

  box-sizing: content-box;

  width: 200px;

  padding: 20px;

  border: 5px solid;

}
```

2. border-box

- The width and height include content + padding + border.
- The total size remains fixed to the width/height you specify.
- Makes layout easier and prevents unexpected expansion.

```
Ex: .box {

  box-sizing: border-box;

  width: 200px;

  padding: 20px;

  border: 5px solid;

}
```

- content-box is the default box-sizing in CSS.

Lab Assignment

Task

Create a profile card layout using the box model. The profile card should include:

  - A profile picture.
  - The user's name and bio.
  - A button to "Follow" the user.

Additional Requirements:

  - Add padding and borders to the elements.
  - Ensure the layout is clean and centered on the page using CSS margins.
  - Use the box-sizing property to demonstrate both content-box and border- box on different elements.

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Profile Card</title>

   <style>
```

```css
body {

    font-family: Arial, sans-serif;

    background-color: #f2f2f2;

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    margin: 0;

}


.profile-card {

    width: 300px;

    background-color: #fff;

    border: 2px solid #ccc;

    padding: 20px;

    border-radius: 10px;

    text-align: center;

    margin: 20px;

    box-sizing: border-box; /* Using border-box */

}


.profile-card img {

    width: 120px;

    height: 120px;

    border-radius: 50%;
    border: 3px solid #555;

    padding: 5px;

    box-sizing: content-box; /* Using content-box */

}


.profile-name {

    font-size: 22px;

    margin: 15px 0 5px;

    font-weight: bold;

}


.profile-bio {

    font-size: 14px;

    color: #555;

    margin-bottom: 20px;

    padding: 10px;

    border: 1px solid #ddd;

    box-sizing: border-box;

}


.follow-btn {

    background-color: #007bff;

    color: white;

    padding: 10px 20px;

    border: none;
```

```
            border-radius: 5px;

            cursor: pointer;

            font-size: 16px;

            box-sizing: content-box;

        }


        .follow-btn:hover {

            background-color: #0056b3;

        }

    </style>

</head>

<body>

    <div class="profile-card">

        <img src="https://via.placeholder.com/120" alt="Profile Picture">

        <div class="profile-name">John Doe</div>

        <div class="profile-bio">Web developer passionate about creating clean UI and
responsive designs.</div>

        <button class="follow-btn">Follow</button>

    </div>

</body>

</html>
```

CSS Flex Box

Theory Assignment

Q.1 : **What is CSS Flexbox, and how is it useful for layout design? Explain the terms flex-container and flex-item.**

CSS Flexbox is a one-dimensional layout system that arranges items in a row **or** a column. It automatically adjusts space distribution between items and handles alignment, spacing, and resizing efficiently.

How Flexbox Is Useful for Layout Design

1). Makes Responsive Layouts Easy

Flexbox automatically adjusts the size of items based on screen size.
Items shrink, grow, or wrap as needed—perfect for mobile-friendly design.

2). Simple Vertical and Horizontal Centering

Centering items (which used to be difficult in CSS) becomes very easy.

3). Controls Spacing Between Items Efficiently

Flexbox provides clean spacing options like:

- justify-content: space-between
- justify-content: space-around
- gap: 20px (no need for margin hacks!)

4). Flexible Box Sizes

Elements can grow or shrink depending on available space.

5). Supports Both Row and Column Layouts

Flexbox can switch between horizontal and vertical layouts

- ➢ **Flex Container -** A flex container is the parent element that has display: flex or display: inline-flex applied to it.

- **It :** Activates Flexbox layout.
- Controls how its children (flex items) are arranged.
- Manages direction, alignment, spacing, and wrapping.

- **Flex Item** - A flex item is any direct child of a flex container.
    - **It** : Each flex item can grow, shrink, or adjust its size based on flex properties.
    - Its position and spacing are controlled by the flex container.
    - It can also override alignment individually using align-self

Q.2 : **Describe the properties justify-content, align-items, and flex-direction used in Flexbox.**

1).flex-direction

This property sets the direction of the main axis along which flex items are placed.

Values : row → Items placed left to right (default)

- row-reverse → Items placed right to left
- column → Items placed top to bottom
- column-reverse → Items placed bottom to top

2).justify - content

This property controls how flex items are aligned along the main axis
(Main axis = horizontal if flex-direction: row, vertical if column)

Values:

- flex-start → Items packed at the start
- flex-end → Items packed at the end
- center → Items centered
- space-between → Equal space between items
- space-around → Equal space around items
- space-evenly → Equal space everywhere

3) align – items

This property aligns items along the cross axis
(Cross axis = perpendicular to main axis)

Values:

- flex-start → Aligned at the start of the cross axis
- flex-end → Aligned at the end
- center → Centered across the cross axis
- stretch → Items stretch to fill (default)
- baseline → Align by text baseline

Lab Assignment

Task

Create a simple webpage layout using Flexbox. The layout should include:

- A header
- A sidebar on the left
- A main content area in the center.
- A footer

Additional Requirements:

- Use Flexbox to position and align the elements.
- Apply different justify-content and align-items properties to observe their effects.
- Ensure the layout is responsive, adjusting for smaller screens.

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Flexbox Webpage Layout</title>

  <style>

    body {

      margin: 0;

      font-family: Arial, sans-serif;

      display: flex;
      flex-direction: column;

      height: 100vh;

    }


    header {

      background-color: #4CAF50;

      color: white;

      padding: 20px;

      text-align: center;

    }


    .container {

      display: flex;

      flex: 1;

      justify-content: flex-start;

      align-items: stretch;

    }


    .sidebar {

      background-color: #ddd;

      width: 200px;

      padding: 20px;

      box-sizing: border-box;

    }
```

```
 .main-content {
    background-color: #f4f4f4;
    flex: 1;
    padding: 20px;
    box-sizing: border-box;
 }

 footer {
    background-color: #333;
    color: white;
    padding: 15px;
    text-align: center;
 }


@media (max-width: 700px) {
    .container {
       flex-direction: column;
       align-items: center;
    }

    .sidebar {
       width: 100%;
       text-align: center;
    }
```

```
            .main-content {
                width: 100%;
            }
        }
    </style>
</head>
<body>
 <header>
        <h1>My Flexbox Webpage</h1>
    </header>
<div class="container">
        <div class="sidebar">Sidebar Content</div>
        <div class="main-content">
            <h2>Main Content Area</h2>
            <p>This is the central content section of the webpage.</p>
        </div>
    </div>
<footer>
        <p>Footer Section</p>
    </footer>
</body>
</html>
```

CSS Grid

Theory Assignment

Q.1: Explain CSS Grid and how it differs from Flexbox. When would you use Grid over Flexbox?

CSS Grid Layout is a two-dimensional layout system used to design webpages using rows and columns.
It allows you to place items precisely within a structured grid, similar to a table but far more flexible and responsive.

It is easy to:

- Create complex page layouts
- Define explicit rows and columns
- Position items in both directions (horizontal + vertical)
- Overlap elements (like layers)

Ex:
```
.container {

   display: grid;

   grid-template-columns: 1fr 2fr;

   grid-template-rows: auto auto;

}
```

- **How Grid Differs from Flexbox**

Feature	**Flexbox**	**CSS Grid**
Layout Type	One-dimensional (row *or* column)	Two-dimensional (rows *and* columns)
Best For	Content-driven layouts	Layout-driven (page structure)
Alignment	Works along one axis at a time	Works along both axes simultaneously
Item Placement	Auto-flow; items follow source order	Items can be placed anywhere in the grid
Control	Good for distributing space	Ideal for full page layouts
Complexity	Simple to moderate	Great for complex designs

- **When to Use Grid Over Flexbox**

- You need a two-dimensional layout
- The design requires precise control over rows and columns
- Items must be placed in specific location
- You want cleaner code for structured layouts

Q.2: Describe the grid-template-columns, grid-template-rows, and grid-gap properties. Provide examples of how to use them.

1)grid – template columns

Defines the number of columns in a grid and their widths. Define column count + width.

How it works:

You specify sizes using units like:

- px (pixels)
- fr (fractional unit)
- % (percentage)
- Auto

Ex: grid-template-columns: 1fr 1fr 1fr;

2)grid – template rows

What it does:

Defines the number of rows and their heights.

Ex: grid-template-rows: 100px 150px 100px;

3) grid-gap (or gap)

What it does:

Adds spacing between grid items (both rows and columns).

Ex: grid-gap: 20px;

Lab Assignment

Task

Create a 3x3 grid of product cards using CSS Grid. Each card should contain:

- A product image.
- A product title
- A price

Additional Requirements:

- o Use grid-template-columns to create the grid layout.
- o Use grid-gap to add spacing between the grid items.
- o Apply hover effects to each card for better interactivity.

⇨ Product.html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Product Grid</title>

  <link rel="stylesheet" href="style.css">

</head>

<body>



  <div class="product-grid">

    <div class="card">

      <img src="https://via.placeholder.com/200" alt="Product">

      <h3>Product Title</h3>
    <p class="price">$19.99</p>
</div>

<div class="card">
    <img src="https://via.placeholder.com/200" alt="Product">
    <h3>Product Title</h3>
    <p class="price">$29.99</p>
</div>

<div class="card">
    <img src="https://via.placeholder.com/200" alt="Product">
    <h3>Product Title</h3>
    <p class="price">$39.99</p>
</div>

<div class="card">
    <img src="https://via.placeholder.com/200" alt="Product">
    <h3>Product Title</h3>
    <p class="price">$49.99</p>
</div>

<div class="card">
    <img src="https://via.placeholder.com/200" alt="Product">
    <h3>Product Title</h3>
    <p class="price">$59.99</p>
```

```
        </div>
    <div class="card">
            <img src="https://via.placeholder.com/200" alt="Product">
            <h3>Product Title</h3>
            <p class="price">$69.99</p>
        </div>

        <div class="card">
            <img src="https://via.placeholder.com/200" alt="Product">
            <h3>Product Title</h3>
            <p class="price">$79.99</p>
        </div>

        <div class="card">
            <img src="https://via.placeholder.com/200" alt="Product">
            <h3>Product Title</h3>
            <p class="price">$89.99</p>
        </div>
    <div class="card">
            <img src="https://via.placeholder.com/200" alt="Product">
            <h3>Product Title</h3>
            <p class="price">$99.99</p>
        </div>

      </div
```

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

⇨ style.css

```
body {
    font-family: Arial, sans-serif;
    background: #f4f4f4;
    padding: 40px;
}
.product-grid {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-gap: 20px;
}
.card {
    background: #fff;
    padding: 15px;
    border-radius: 10px;
    text-align: center;
    box-shadow: 0px 3px 8px rgba(0,0,0,0.1);
    transition: 0.3s ease;
}
.card:hover {
    transform: translateY(-8px);
    box-shadow: 0px 8px 15px rgba(0,0,0,0.2);
```

```
  }

   .card img {
      width: 100%;
      border-radius: 10px;
   }
.price {
      font-size: 18px;
      font-weight: bold;
      color: #27ae60;
      margin-top: 10px;
   }
```

Responsive Web Design with Media Queries

Theory Assignment

Q.1: What are media queries in CSS, and why are they important for responsive design?

A media query checks the device or browser's properties and applies CSS only when the condition is true.

```
Ex: @media (max-width: 600px) {

  body {

    background-color: lightblue;

  }

}
```

➢ **Why Are Media Queries Important for Responsive Design?**

Media queries make websites adapt automatically to different screen sizes, such as:

- Mobile phones
- Tablets
- Laptops
- Desktops
- Large monitors

✔ **Importance:**

1) Improve User Experience
2) Mobile-Friendly Layouts
3) Better Performance
4) Avoid Layout Breaks
5) Required for Modern Web Development

Q.2 : Write a basic media query that adjusts the font size of a webpage for screens smaller than 600px

Here is a simple, clean media query that adjusts the font size for screens smaller than 600px

Code : @media (max-width: 600px) {

```
   body {
      font-size: 14px;
   }
}
```

Lab Assignment (Task)

Build a responsive webpage that includes:

- A navigation bar.
- A content section with two columns.
- A footer.

Additional Requirements:

- Use media queries to make the webpage responsive for mobile devices.
- On smaller screens (below 768px), stack the columns vertically.
- Adjust the font sizes and padding to improve readability on mobile.

⇨ Html structure

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Responsive Webpage</title>
   <link rel="stylesheet" href="style.css">
</head>
<body>
```

```
<nav class="navbar">
        <h2 class="logo">My Website</h2>
        <ul class="nav-links">
           <li><a href="#">Home</a></li>
           <li><a href="#">About</a></li>
           <li><a href="#">Services</a></li>
           <li><a href="#">Contact</a></li>
        </ul>
    </nav>
<div class="content">
        <div class="column left">
           <h2>Left Column</h2>
           <p>
              This is some content in the left column. Lorem ipsum dolor sit amet,
              consectetur adipiscing elit. Nulla vitae.
           </p>
        </div>

        <div class="column right">
           <h2>Right Column</h2>
           <p>
              This is some content in the right column. Lorem ipsum dolor sit amet,
              consectetur adipiscing elit. Nulla vitae.
           </p>
```

```
            </div>
          </div>
      <footer class="footer">
            <p>© 2025 My Website. All rights reserved.</p>
          </footer>
       </body>
       </html>
```

⇨ CSS

```
/* General Page Styling */
body {
   margin: 0;
   font-family: Arial, sans-serif;
   background: #f4f4f4;
   line-height: 1.6;
}
.navbar {
   display: flex;
   justify-content: space-between;
   align-items: center;
   background: #333;
   padding: 15px 20px;
}
```

```
.navbar .logo {
   color: #fff;
}

.navbar .nav-links {
   list-style: none;
   display: flex;
   gap: 20px;
}

.navbar .nav-links a {
   color: white;
   text-decoration: none;
   font-size: 16px;
}
.content {
   display: grid;
   grid-template-columns: 1fr 1fr;
   gap: 20px;
   padding: 40px;
}

.column {
   background: white;
```

```
    padding: 20px;
    border-radius: 10px;
}
.footer {
    background: #333;
    color: white;
    text-align: center;
    padding: 15px 0;
}
@media (max-width: 768px) {
    .content {
        grid-template-columns: 1fr;
        padding: 20px;
    }

    /* Adjust font sizes */
    body {
        font-size: 16px;
    }

    .navbar .nav-links {
        gap: 10px;
    }
    .navbar .nav-links a {
        font-size: 14px;
    }
.column {
        padding: 15px;
    }
}

@media (max-width: 480px) {

    /* Even smaller screens */
    body {
        font-size: 14px;
    }

    .navbar {
        flex-direction: column;
        text-align: center;
        gap: 10px;
    }
 .column {
        padding: 10px;
    }
}
```

Typography and Web Fonts

Theory Assignment

Q.1: Explain the difference between web-safe fonts and custom web fonts. Why might you use a web-safe font over a custom font?

1. Web-Safe Fonts

Web-safe fonts are pre-installed on most operating systems (Windows, macOS, Linux, Android, iOS).
They do not need to be downloaded from the internet.

Examples

- Arial
- Times New Roman
- Verdana

Characteristics

- Fast to load
- High compatibility across devices
- No external files needed

2. Custom Web Fonts

Custom web fonts are fonts that are not pre-installed on user devices.
They must be downloaded from a server using @font-face or services like:

- Google Fonts
- Adobe Fonts

Examples

- Poppins
- Roboto
- Lato

Characteristics

- Provide unique and modern typography
- Allow branding and design flexibility
- Require additional HTTP requests (slower than web-safe fonts)

- **Why Use a Web-Safe Font Over a Custom Font?**
  - Better Performance
  - No Flash of Unstyled Text (FOUT)
  - Improved Accessibility
  - Higher Reliability
  - Useful for Simple or Lightweight Projects

Q.2: What is the font-family property in CSS? How do you apply a custom Google Font to a webpage?

The font-family property is used to specify the typeface (font) that should be applied to text on a webpage.

- It can include multiple fonts as a fallback list.
- Browsers will use the first available font in the list.
- Either web-safe fonts or custom fonts can be used.

```
 Ex: body {

   font-family: Arial, Helvetica, sans-serif;

}
```

How to apply a custom Google font to a webpage ?

- Import font using a <link> or @import.
- Apply it using font-family.

Lab Assignment

Task

Create a blog post layout with the following:

- A title, subtitle, and body content.
- Use at least two different fonts (one for headings, one for body content).
- Style the text to be responsive and easy to read.

Additional Requirements:

  - Use a custom font from Google Fonts.
  - Adjust line-height, font-size, and spacing for improved readability.

⇨ Blog.html

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Blog Post</title>

   <link
href="https://fonts.googleapis.com/css2?family=Merriweather:wght@700&fam
ily=Open+Sans:wght@400;600&display=swap" rel="stylesheet">


   <link rel="stylesheet" href="style.css">

</head>



<body>



   <div class="blog-container">

      <h1 class="title">The Future of Web Development</h1>

      <h3 class="subtitle">How AI and Modern Tools Are Changing the
Web</h3>



      <p class="body-text">
```

```
            Web development is evolving faster than ever. With advancements in
artificial intelligence, automated tools, and improved frameworks, developers
now have powerful ways to build more efficient, responsive, and intelligent
applications.

            <br><br>

            As we move forward, understanding how these technologies integrate
into the development process will be essential for building modern and scalable
solutions. The future of web development is not just about writing code—it's
about creating adaptive, intelligent, and efficient experiences for users.

        </p>

    </div>

</body>

</html>
```

⇨ style.css

```
body {

    margin: 0;

    padding: 0;

    background: #fafafa;

    font-family: 'Open Sans', sans-serif;

    font-size: 18px;

    line-height: 1.7;

}


.blog-container {

    max-width: 800px;
    margin: 50px auto;
    padding: 20px;
}
.title {
    font-family: 'Merriweather', serif;
    font-size: 36px;
    margin-bottom: 10px;
    color: #333;
}

.subtitle {
    font-family: 'Merriweather', serif;
    font-size: 22px;
    margin-bottom: 25px;
    color: #555;
}
.body-text {
    font-size: 18px;
    color: #444;
}
@media (max-width: 768px) {
    .title {
        font-size: 28px;
    }

    .subtitle {
        font-size: 18px;
    }

    body {
        font-size: 16px;
        line-height: 1.6;
    }

    .blog-container {
        padding: 15px;
        margin: 20px;
    }
}

@media (max-width: 480px) {
    .title {
        font-size: 24px;
    }

    .subtitle {
        font-size: 16px;
    }

    body {
        font-size: 15px;
        line-height: 1.55;
    }
}
```