Training Day 16

Day 16– 12th July 2025

Responsive Grid Layout using CSS Grid

Detailed Description:

On Day 16, we explored **CSS Grid**, a modern layout system that allows us to **design complex web layouts easily**.
The instructor explained that CSS Grid provides **rows and columns**, making it easier to control alignment, spacing, and responsiveness compared to older methods like floats or positioning.

🔹 1. Introduction to CSS Grid

CSS Grid divides a container into **rows and columns**, where child elements can be placed precisely.

Basic syntax:

```
.container {

  display: grid;

  grid-template-columns: 1fr 1fr 1fr; /* 3 equal columns */

  grid-gap: 20px; /* spacing between grid items */

}
```

Example practiced:

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

```
  <div class="item">1</div>

  <div class="item">2</div>

  <div class="item">3</div>

</div>
```

- Each .item is automatically placed into the grid.
- fr unit represents a fraction of available space, making the layout **flexible**.

◆ 2. Placing Items in Grid

We learned to **position elements explicitly** using grid-column and grid-row:

```
.item1 {

  grid-column: 1 / 3; /* spans from column 1 to 2 */

  grid-row: 1 / 2;

}
```

- This allows us to **create complex layouts** like magazine-style designs or dashboard interfaces.

◆ 3. Grid Template Areas

Grid template areas allow naming of sections for **readable layout structure**:

```
.container {
  display: grid;

  grid-template-areas:

    "header header"

    "sidebar main"

    "footer footer";

  grid-gap: 10px;

}


.header { grid-area: header; }

.sidebar { grid-area: sidebar; }

.main { grid-area: main; }

.footer { grid-area: footer; }
```

- Makes the layout **intuitive and easy to maintain**.

🔹 **4. Responsive Grid with Media Queries**

We combined **CSS Grid with media queries** to make layouts **responsive**:

```
@media (max-width: 768px) {

  .container {

    grid-template-columns: 1fr; /* single column on smaller screens */
    grid-template-areas:
      "header"
      "main"
      "sidebar"
      "footer";
  }
}
```

- This allows the layout to **adapt automatically** for tablets and mobile devices.
- Practiced resizing the browser window to see **dynamic reflow of grid items**.

◆ 5. Practical Exercises

- Created a **3-column layout** for desktop screens.
- Implemented **named grid areas** for header, sidebar, main content, and footer.
- Made the grid **responsive for tablets and mobiles** using media queries.
- Experimented with **grid-gap, fractional units, and explicit placement** for better design control.

Learning Outcomes:

- Learned the fundamentals of **CSS Grid** for layout design.
- Practiced **placing items explicitly** and using **grid template areas**.

- Understood how to **create responsive layouts** using grid and media queries.
- Developed skills to build **modern, flexible, and well-structured webpages**.
- Realized that CSS Grid **simplifies complex layouts** compared to older methods.