Training Report - Web Development Training

Day 13: CSS Grid

Date: 21/6/24

Summary of the Day: On the thirteenth day of our web development training, we delved into CSS Grid. CSS Grid is a powerful two-dimensional layout system that allows developers to create complex and responsive grid-based layouts with ease. The session focused on understanding the basics of CSS Grid, including container properties and grid item properties.

Detailed Notes:

1. Introduction to CSS Grid: CSS Grid is designed to handle both rows and columns, providing a more robust layout system compared to Flexbox, which is primarily one-dimensional.

2. CSS Grid Container Properties: The following properties are used to define a grid container and specify its behavior.

- **display: grid;** Turns an element into a grid container.

```
.grid-container {
  display: grid;
}
```

- **grid-template-columns and grid-template-rows:** Define the number and size of the columns and rows in the grid.

```
.grid-container {
  grid-template-columns: 1fr 1fr 1fr; /* Three equal columns */
  grid-template-rows: 100px 200px; /* Two rows with specific heights */
}
```

- **grid-template-areas:** Names grid areas for easier reference and layout.

```
.grid-container {
  grid-template-areas:
    'header header header'
    'sidebar content content'
    'footer footer footer';
}
```

- **grid-gap (or gap):** Sets the spacing between rows and columns.

```
.grid-container {
  grid-gap: 10px; /* Space between rows and columns */
}
```

- **grid-auto-flow:** Controls how auto-placed items are inserted into the grid.

```
.grid-container {
  grid-auto-flow: row; /* Default */
}
/* Other values: column, dense, row dense, column dense */
```

3. CSS Grid Item Properties: The following properties are used to define the placement and behavior of grid items within a grid container.

- **grid-column-start, grid-column-end, grid-row-start, grid-row-end:** Specify the start and end positions of a grid item.

```
.grid-item {
  grid-column-start: 1;
  grid-column-end: 3; /* Span across the first two columns */
  grid-row-start: 1;
  grid-row-end: 2; /* Span across the first row */
}
```

- **grid-column and grid-row:** Shorthand for setting both the start and end positions.

```
.grid-item {
  grid-column: 1 / 3; /* Span across the first two columns */
  grid-row: 1 / 2; /* Span across the first row */
}
```

- **grid-area:** Assigns an item to a named grid area.

```
.grid-item {
  grid-area: header;}
```

- **justify-self:** Aligns the grid item within its column.

```
.grid-item {
  justify-self: center; /* Default is stretch */
```

```
}
/* Other values: start, end, stretch */
```

- **align-self:** Aligns the grid item within its row.

```
.grid-item {
  align-self: center; /* Default is stretch */
}
/* Other values: start, end, stretch */
```

Reflection: Today's session on CSS Grid was enlightening. Learning about the container and item properties of CSS Grid has opened up new possibilities for creating complex, responsive web layouts. The ability to control both rows and columns simultaneously makes CSS Grid an incredibly powerful tool for modern web design. I am excited to apply these concepts in my future projects to build sophisticated and well-organized web pages.