DAY 3 → Advance Layout : Grid and media query

1. Grid Container:

- **Description: The parent element that holds all the grid items and defines the grid context.**
- **Example:**

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

2. Grid Columns and Rows:

- **Description: Specifies the number and size of columns and rows in the grid.**
- **Example:**

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

3. Grid Items:

- **Description: The child elements placed inside the grid container.**
- **Example:**

```css
.grid-item {
  grid-column: 2 / 4; /* Spanning from the second column to the fourth column */
  grid-row: 1; /* Placed in the first row */
}
```

4. Grid Lines:
   - Description: Imaginary lines that divide the grid into columns and rows.
   - Example:

```css
.grid-container {
  grid-template-columns: [col1] 100px [col2] 200px [col3]; /* Custom grid lines */
  grid-template-rows: [row1] 100px [row2] 200px [row3];
}
.grid-item {
  grid-column: col1 / col3; /* Spanning from col1 to col3 */
  grid-row: row1 / row2; /* Spanning from row1 to row2 */
}
```

5. Grid Gaps:
   - Description: Adds spacing between grid cells.
   - Example:

```css
.grid-container {
  gap: 10px; /* 10px gap between grid cells */
}
```

6. Grid Areas:
  - Description: Allows you to name and reference areas of the grid.
  - Example:
```css
.grid-container {
  grid-template-areas:
    "header header header"
    "sidebar main main"
    "footer footer footer";
}
.grid-item {
  grid-area: main; /* Placed in the 'main' area */
}
```

7. Grid Line-based Placement:
  - Description: Positions items based on grid lines rather than explicit grid positions.
  - Example:
```css
.grid-item {
  grid-column-start: 2; /* Starts at the second column line */
  grid-column-end: span 2; /* Spans across two columns */
  grid-row: 1; /* Placed in the first row */
}
```

8. Grid Auto Placement:
   - Description: Automatically positions items in the grid without explicitly defining their placement.
   - Example:
     ```css
     .grid-container {
       grid-auto-flow: row; /* Items are placed in rows */
     }
     ```

9. Grid Alignment:
   - Description: Controls the alignment of grid items within their cells.
   - Example:
     ```css
     .grid-container {
       justify-items: center; /* Centers items horizontally in their cells */
       align-items: center; /* Centers items vertically in their cells */
     }
     ```

10. Responsive Grids:
    - Description: Adapting the grid layout based on different screen sizes using media queries.
    - Example:
      ```css
      ```

```
    .grid-container {
      grid-template-columns: 1fr 1fr; /* Two columns */
    }

    @media screen and

 (max-width: 600px) {
      .grid-container {
        grid-template-columns: 1fr; /* One column on smaller
screens */
      }
    }
```

MEDIA QUERY → RESPONSIVE DESIGN

Media queries allow you to apply different CSS styles based on the characteristics of the device or browser viewport. This enables you to create responsive designs that adapt to different screen sizes, orientations, and other device properties.

Syntax:
```css
@media mediaType and (mediaFeature) {
  /* CSS styles to apply */
}
```

- `mediaType`: Specifies the type of media the query targets, such as `screen`, `print`, or `all`.
- `mediaFeature`: Defines the condition or characteristic on which the styles will be applied. Examples include `max-width`, `min-width`, `orientation`, `max-device-width`, etc.

Example:
```css
/* Styles applied when the viewport width is less than or equal to 600 pixels */
@media screen and (max-width: 600px) {
  /* CSS styles for smaller screens */
  body {
    font-size: 14px;
  }
  .container {
    width: 90%;
  }
}

/* Styles applied when the viewport width is greater than 600 pixels */
@media screen and (min-width: 601px) {
  /* CSS styles for larger screens */
  body {
    font-size: 16px;
  }
  .container {
    width: 70%;
  }
}
```

In the example above, we have two media queries. The first one targets screens with a maximum width of 600 pixels and applies specific styles. The second query targets screens with a minimum width of 601 pixels and applies different styles.

Media queries provide a powerful tool to create responsive designs and adapt the layout and styles based on the characteristics of the user's device or viewport. By using media queries effectively, you can optimize the user experience across various devices and screen sizes.