DISPLAY:FLEX

In HTML and CSS, `display: flex` is a value used for the `display` property to create a flex container. This allows you to use the Flexbox layout model, which makes it easier to design flexible and responsive layout structures.

Key Features of Flexbox (`display: flex`)

1. **Flex Container**: When you apply `display: flex` to an element, that element becomes a flex container. All of its direct children become flex items.
2. **Flexible Layout**: Flexbox enables flexible and predictable layouts by distributing space along a single axis (either horizontally or vertically).
3. **Alignment and Justification**: You can easily align and justify flex items within the container. This includes properties like `justify-content`, `align-items`, `align-self`, and `align-content`.
4. **Direction Control**: The direction of the flex items can be controlled with `flex-direction`, which can be `row`, `row-reverse`, `column`, or `column-reverse`.
5. **Wrapping**: Flexbox can handle wrapping of items with the `flex-wrap` property, allowing items to move to new lines if they overflow the container.
6. **Item Sizing**: Flex items can grow and shrink to fill available space with the `flex-grow`, `flex-shrink`, and `flex-basis` properties.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        .container {
            display: flex;
            justify-content: space-between; /* Distributes space between items */
            align-items: center; /* Aligns items vertically */
            border: 2px solid black;
            padding: 10px;
        }
        .item {
            background-color: lightblue;
            padding: 10px;
            border: 1px solid blue;
```

```
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="item">Item 1</div>
    <div class="item">Item 2</div>
    <div class="item">Item 3</div>
  </div>
</body>
</html>
```

- The `.container` div is a flex container because of `display: flex`.
- The `.item` divs are flex items and will be laid out according to the flexbox model.
- `justify-content: space-between` distributes the items with space between them.
- `align-items: center` centers the items vertically within the container.

Flexbox is highly versatile and useful for creating complex layouts with less code and more predictable results compared to older layout methods.

DISPLAY:GRID

`display: grid` is a CSS property that enables the Grid Layout system, which is a powerful tool for creating two-dimensional layouts on the web. With CSS Grid, you can design complex layouts with rows and columns, making it easier to align and distribute space among items within a container.

Key Features of CSS Grid

1. **Grid Container**: Applying `display: grid` to an element turns it into a grid container, and its direct children become grid items.
2. **Grid Tracks**: You can define the rows and columns of the grid using properties like `grid-template-rows` and `grid-template-columns`.
3. **Grid Gaps**: The space between grid items can be controlled with `gap` (or `grid-gap` for older syntax), `row-gap`, and `column-gap`.
4. **Placement**: Grid items can be placed explicitly in specific cells or areas using properties like `grid-column`, `grid-row`, and `grid-area`.
5. **Responsive Design**: Grid layouts are flexible and responsive, allowing for easier adjustments and rearrangements of items as the viewport size changes.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        .grid-container {
            display: grid;
            grid-template-columns: repeat(3, 1fr); /* Creates 3 equal-width columns */
            grid-template-rows: repeat(2, 100px); /* Creates 2 rows, each 100px tall */
            gap: 10px; /* Space between grid items */
            border: 2px solid black;
            padding: 10px;
        }
        .grid-item {
            background-color: lightcoral;
            border: 1px solid red;
            padding: 20px;
```

```
            text-align: center;
        }
    </style>
</head>
<body>
    <div class="grid-container">
        <div class="grid-item">1</div>
        <div class="grid-item">2</div>
        <div class="grid-item">3</div>
        <div class="grid-item">4</div>
        <div class="grid-item">5</div>
        <div class="grid-item">6</div>
    </div>
</body>
</html>
```

Explanation

- **Grid Container**: The `.grid-container` div is set up as a grid container with `display: grid`.
- **Grid Template Columns**: `grid-template-columns: repeat(3, 1fr)` creates three columns of equal width. `1fr` stands for one fraction of the available space.
- **Grid Template Rows**: `grid-template-rows: repeat(2, 100px)` creates two rows, each 100 pixels tall.
- **Gap**: `gap: 10px` adds a 10-pixel space between grid items.
- **Grid Items**: The `.grid-item` divs are the items within the grid, and they will be automatically placed into the grid cells created by the grid container.