# CSS Grid Layout

## CSS Grid Layout Module

- The Grid Layout Module offers a grid-based layout system, with rows and columns.
- The Grid Layout Module allows developers to easily create complex web layouts.
- The Grid Layout Module makes it easy to design a responsive layout structure, without using float or positioning.

**A grid always consists of:**

- **A Grid Container** - The parent (container) element, where the display property is set to grid or inline-grid
- **One or more Grid Items** - The direct children of the grid container automatically becomes grid items

```
.container {
  display: grid;
  grid-template-columns: auto auto auto;
  background-color: dodgerblue;
  padding: 10px;
}
```

## Display Grid Property

The <div> element becomes a grid container when its display property is set to grid or inline-grid.

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

## CSS Grid Container Properties

- **CSS Grid Tracks (Rows and Colums):** Inside the grid container you define the number and size of the grid columns and rows.
    - **grid-template-columns** - Defines the number and width of the columns in the grid
    - **grid-template-rows** - Defines the number and height of the rows in the grid

- **grid-template-areas** - Defines how to display columns and rows, using named grid items

**The CSS grid-template-columns Property:**

- The grid-template-columns property defines the number and width of the columns in the grid.
- The value is a space-separated-list, where each value defines the width of the respective column.
- **Comman Values are:**
  - Fixed lengths (100px 300px 200px)
  - Percentages (20% 60% 20%)
  - fr unit (1fr 2fr 1fr)
  - auto (auto auto auto)
  - repeat() (repeat(3, 1fr))
  - minmax() (minmax(80px, 1fr) 150px 150px)
- 
```
.container{
        display: grid;
        grid-template-columns: auto auto auto;
}
```

**The CSS grid-template-rows Property:**

- The grid-template-rows property defines the number and height of the rows in the grid.
- The value is a space-separated-list, where each value defines the height of the respective row.
- **Common values are:**
  - Fixed lengths
  - Percentages
  - fr unit
  - auto
  - min-content
  - max-content
  - repeat()
  - minmax()
  - fit-content()

- .container{

  ```
  display: grid;
  grid-template-rows: 80px 200px;
  ```

  }

## CSS Grid Gaps

- The space between the rows and columns in a grid container are called gaps (or gutters).
- The gaps are created between the grid rows and columns, not on the outer edges of the grid container.

The size of the gap can be adjusted with the following properties:

- **column-gap** - Specifies the gap between grid columns
  - .container{

    ```
    display: grid;
    cloumn-gap: 50px;
    ```

    }
- **row-gap** - Specifies the gap between grid rows
  - .container{

    ```
    display: grid;
    row-gap: 50px;
    ```

    }
- **gap** - Shorthand property for row-gap and column-gap
  - .container{

    ```
    display: grid;
    gap: 50px;
    ```

    }

## CSS Grid Container - Align Grid Content

- **justify-content** - Aligns the grid content when it does not use all available space on the main-axis (horizontally)
  - space-evenly
  - space-around
  - space-between
  - center
  - start

    - end
- **align-content** - Aligns the grid content when it does not use all available space on the cross-axis (vertically)
    - space-evenly
    - space-around
    - space-between
    - center
    - start
    - end

- **place-content** - Shorthand property for align-content and justify-content
    - **place-content: start center;** - the align-content value is 'start' and justify-content value is 'center'

        -
          ```
          .container {
                  display: grid;
                  height: 300px;
                  place-content: center;
          }
          ```
    - **place-content: end;** - both align-content and justify-content values are 'end'

        -
          ```
          container {
                  display: grid;
                  height: 300px;
                  place-content: end space-between;
          }
          ```

## CSS Grid Items

- A grid container contains one or more grid items.
- All direct child elements of a grid container automatically become grid items.
- **grid-column-start** - Specifies on which column-line the grid item will start
- **grid-column-end** - Specifies on which column-line the grid item will end

    -
      ```
      .item1 {
              grid-column-start: 1;
              grid-column-end: 3;
      }
      ```
- **grid-column** - Shorthand property for grid-column-start and grid-column-end

-
  ```
  .item1 {
              grid-column: 1 / span 2;
  }
  ```
- **grid-row-start** - Specifies on which row-line the grid item will start
- **grid-row-end** - Specifies on which row-line the grid item will end

  -
    ```
    .item1 {
                grid-row-start: 1;
                grid-row-end: 3;
    }
    ```
- **grid-row** - Shorthand property for grid-row-start and grid-row-end

  -
    ```
    .item1 {
                grid-row: 1 / span 2;
    }
    ```
- **Combine grid-column and grid-row:** we use both the grid-column and grid-row properties to let a grid item span both columns and rows

  -
    ```
    .item1 {
                grid-column: 1 / span 2;
                grid-row: 1 / span 2;
    }
    ```

## CSS Naming Grid Items

- The CSS **grid-template-areas** is a grid container property, and it specifies areas within the grid layout.
- You can name grid items by using the CSS grid-area property, and then reference to the name in the grid-template-areas property.
- Each area is defined within apostrophes. The named grid items in each area is defined inside the apostrophes, separated by a space.

-
  ```
  .container {
        display: grid;
         grid-template-areas: 'myHeader myHeader myHeader myHeader
  myHeader';
  }

  .item1 {
              grid-area: myHeader;
  }
  ```

- Full Example:
```
.item1 { grid-area: header; }
.item2 { grid-area: menu; }
.item3 { grid-area: main; }
.item4 { grid-area: right; }
.item5 { grid-area: footer; }

.container {
        display: grid;
        grid-template-areas:
        'header header header header header header'
        'menu main main main main right'
        'menu footer footer footer footer footer';
}
```

## CSS Grid Items – Alignment

- **justify-self** - Specifies the horizontal alignment within a cell
    - This property can have one of the following values:
        - auto (default)
        - normal
        - stretch
        - start
        - left
        - center
        - end
        - Right

    -
      ```
      .item1 {justify-self: right;}
      .item6 {justify-self: center;}
      ```
- **align-self** - Specifies the vertical alignment within a cell
    - This property can have one of the following values:
        - auto (default)
        - normal
        - stretch
        - start
        - end
        - center

    -
      ```
      .item1 {align-self: start;}
      .item6 {align-self: center;}
      ```

- **place-self** - Shorthand property for align-self and justify-self

    -
      ```
      .item {
              display: grid;
              place-self: center;
      }
      ```

## CSS Grid Items – Order

- The CSS order property can be used to define the visual order of the grid items.
- The first grid item in the HTML source code does not have to appear as the first item in the grid container.

-
  ```
  .item1 {order: 3;}
  .item2 {order: 6;}
  .item3 {order: 1;}
  .item4 {order: 2;}
  .item5 {order: 4;}
  .item6 {order: 5;}
  ```

| Property | Description |
|---|---|
| **align-content** | Vertically aligns the grid items inside the container |
| **align-items** | Specifies the default alignment for items inside a flexbox or grid container |
| **display** | Specifies the display behavior (the type of rendering box) of an element |
| **column-gap** | Specifies the gap between the columns |
| **gap** | A shorthand property for the row-gap and the column-gap properties |
| **grid** | A shorthand property for the grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns, and the grid-auto-flow properties |
| **grid-auto-columns** | Specifies a default column size |
| **grid-auto-flow** | Specifies how auto-placed items are inserted in the grid |
| **grid-auto-rows** | Specifies a default row size |
| **grid-template** | A shorthand property for the grid-template-rows, grid-template-columns and grid-areas properties |
| **grid-template-areas** | Specifies how to display columns and rows, using named grid items |

| | |
|---|---|
| **grid-template-columns** | Specifies the size of the columns, and how many columns in a grid layout |
| **grid-template-rows** | Specifies the size of the rows in a grid layout |
| **justify-content** | Horizontally aligns the grid items inside the container |
| **place-content** | A shorthand property for the align-content and the justify-content properties |
| **row-gap** | Specifies the gap between the grid rows |