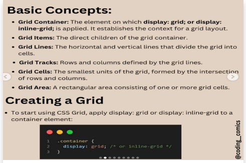


Basic Concepts:

- **Grid Container:** The element on which **display: grid; or display: inline-grid;** is applied. It establishes the context for a grid layout.
- **Grid Items:** The direct children of the grid container.
- **Grid Lines:** The horizontal and vertical lines that divide the grid into cells.
- **Grid Tracks:** Rows and columns defined by the grid lines.
- **Grid Cells:** The smallest units of the grid, formed by the intersection of rows and columns.
- **Grid Area:** A rectangular area consisting of one or more grid cells.

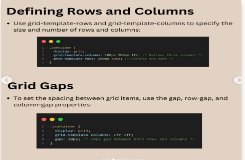
Creating a Grid

- To start using CSS Grid, apply display: grid or display: inline-grid to a container element:

```
.container {
  display: grid; /* or inline-grid */
}
```

@coding__comics

Grid Basics



Defining Rows and Columns

- Use grid-template-rows and grid-template-columns to specify the size and number of rows and columns:

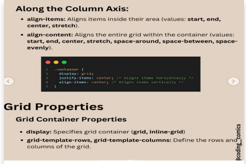
```
.container {
  display: grid;
  grid-template-columns: 100px 200px 1fr; /* Defines three columns */
  grid-template-rows: 100px auto; /* Defines two rows */
}
```

Grid Gaps

- To set the spacing between grid items, use the gap, row-gap, and column-gap properties:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr;
  gap: 10px; /* 10px gap between both rows and columns */
}
```

Grid Defining Rows and Columns



Along the Column Axis:

- **align-items:** Aligns items inside their area (values: **start, end, center, stretch**).
- **align-content:** Aligns the entire grid within the container (values: **start, end, center, stretch, space-around, space-between, space-evenly**).

```
.container {
  display: grid;
  justify-items: center; /* Aligns items horizontally */
  align-items: center; /* Aligns items vertically */
}
```

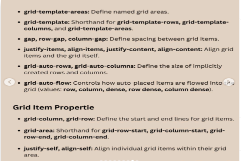
Grid Properties

Grid Container Properties

- **display:** Specifies grid container (**grid, inline-grid**)
- **grid-template-rows, grid-template-columns:** Define the rows and columns of the grid.

@coding__comics

Grid Properties and Align Items

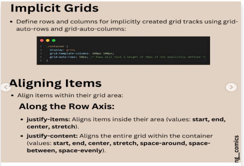


- **grid-template-areas:** Define named grid areas.
- **grid-template:** Shorthand for **grid-template-rows, grid-template-columns,** and **grid-template-areas**.
- **gap, row-gap, column-gap:** Define spacing between grid items.
- **justify-items, align-items, justify-content, align-content:** Align grid items and the grid itself.
- **grid-auto-rows, grid-auto-columns:** Define the size of implicitly created rows and columns.
- **grid-auto-flow:** Controls how auto-placed items are flowed into the grid (values: **row, column, dense, row dense, column dense**).

Grid Item Propertie

- **grid-column, grid-row:** Define the start and end lines for grid items.
- **grid-area:** Shorthand for **grid-row-start, grid-column-start, grid-row-end, grid-column-end**.
- **justify-self, align-self:** Align individual grid items within their grid area.

Grid Properties and Align Items 2



Implicit Grids

- Define rows and columns for implicitly created grid tracks using grid-auto-rows and grid-auto-columns:

```
.container {
  display: grid;
  grid-template-columns: 100px 100px;
  grid-auto-rows: 50px; /* Rows will have a height of 50px if not explicitly defined */
}
```

Aligning Items

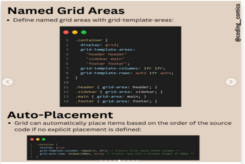
- Align items within their grid area:

Along the Row Axis:

- **justify-items:** Aligns items inside their area (values: **start, end, center, stretch**).
- **justify-content:** Aligns the entire grid within the container (values: **start, end, center, stretch, space-around, space-between, space-evenly**).

ng__comics

Grid - Implicit Grid Areas



Named Grid Areas

- Define named grid areas with grid-template-areas:

```
.container {
  display: grid;
  grid-template-areas:
    "header header"
    "sidebar main"
    "footer footer";
  grid-template-columns: 1fr 2fr;
  grid-template-rows: auto 1fr auto;
}

.header { grid-area: header; }
.sidebar { grid-area: sidebar; }
.main { grid-area: main; }
.footer { grid-area: footer; }
```

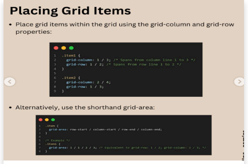
Auto-Placement

- Grid can automatically place items based on the order of the source code if no explicit placement is defined:

```
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr); /* Creates three equal-width columns */
  grid-auto-rows: minmax(100px, auto); /* Creates rows with a minimum height of 100px */
}
```

@coding__comics

Grid Named Areas



Placing Grid Items

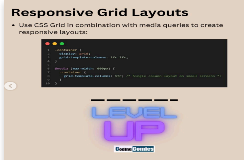
- Place grid items within the grid using the grid-column and grid-row properties:

```
.item1 {
  grid-column: 1 / 3; /* Spans from column line 1 to 3 */
  grid-row: 1 / 2; /* Spans from row line 1 to 2 */
}

.item2 {
  grid-column: 2 / 4;
  grid-row: 1 / 3;
}
```

- Alternatively, use the shorthand grid-area:

```
.item {
  grid-area: row-start / column-start / row-end / column-end;
}

/* Example */
.item1 {
  grid-area: 1 / 1 / 2 / 3; /* Equivalent to grid-row: 1 / 2; grid-column: 1 / 3; */
}
```

Grid - Placing Grid Items



Responsive Grid Layouts

- Use CSS Grid in combination with media queries to create responsive layouts:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr;
}

@media (max-width: 600px) {
  .container {
    grid-template-columns: 1fr; /* Single column layout on small screens */
  }
}
```

Grid - Responsive