GRID

The CSS Grid Layout Module introduces a two-dimensional grid-based layout system to CSS. It allows developers to create complex grid layouts for their web pages with ease. Here's an overview of some key properties associated with CSS Grid:

1. display: grid

Definition:

This property establishes a grid container, enabling a grid context for its direct children. It turns the element into a grid container, allowing you to use other grid properties to control the layout of its children.

Example:

HTML:

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
```

CSS:

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

Use Case:

Use display: grid; when you want to create a grid container to arrange its direct children in a grid layout.

Notes:

This property is applied to the parent container.
It enables Grid layout for its direct children.

2. grid-template-columns

Definition:

Specifies the size of the columns in the grid layout. It defines the number of columns and their sizes in the grid container.
Example:

```
.container {
  display: grid;
  grid-template-columns: 100px 200px 300px; /* Column sizes */
}
```

Use Case:

Use grid-template-columns to define the size of columns in a grid layout, providing flexibility in creating custom column layouts.

Values:

- 100px, 200px, 300px: Fixed size columns in pixels.
- <track-size>: Specifies the size of each column track in the grid layout.
- auto: Automatically sizes the column track based on its content.
- minmax(min, max): Specifies a size range for the column track, ensuring it remains within the specified minimum and maximum sizes.
- repeat(n, track-size): Generates a specified number of column tracks with a given size.

Notes:

Column sizes can be specified using various units like pixels, percentages, or fr units.

3. grid-template-rows

Definition:

Specifies the size of the rows in the grid layout. It defines the number of rows and their sizes in the grid container.
Example:

```
.container {
  display: grid;
  grid-template-rows: 50px 100px 150px; /* Row sizes */
}
```

Use Case:

Use grid-template-rows to define the size of rows in a grid layout, allowing for custom row layouts.

Values:

- 50px, 100px, 150px: Fixed size rows in pixels.
- <track-size>: Specifies the size of each row track in the grid layout.
- auto: Automatically sizes the row track based on its content.
- minmax(min, max): Specifies a size range for the row track, ensuring it remains within the specified minimum and maximum sizes.
- repeat(n, track-size): Generates a specified number of row tracks with a given size.

Notes:

Row sizes can be specified using various units like pixels, percentages, or fr units.

4. grid-template-areas

Definition:

Defines named grid areas, allowing you to visually lay out the grid by assigning names to different areas of the grid.
Example:

```
.container {
  display: grid;
  grid-template-areas:
    "header header header"
    "sidebar main main"
    "footer footer footer";
}
```

Use Case:

Use grid-template-areas to create a visual grid layout by assigning names to different areas of the grid, facilitating easy organization and maintenance of the layout.

Values:

Each string represents a row in the grid layout, with each area name separated by spaces.

Notes:

The layout defined by grid-template-areas can be visualized using the Firefox Grid Inspector or Chrome DevTools.

5. grid-gap

Definition:

Specifies the size of the gap between grid rows and columns. It defines the spacing between grid tracks (rows and columns) in the grid container.
Example:

```
.container {
  display: grid;
  grid-gap: 10px; /* Gap size */
}
```

Use Case:

Use grid-gap to add spacing between grid items, enhancing the readability and aesthetics of the grid layout.

Values:

10px: Gap size in pixels.

Notes:

grid-gap is a shorthand property for grid-row-gap and grid-column-gap, allowing you to set the gap size for rows and columns independently if needed.

6. justify-items

Definition:

Aligns grid items along the inline (row) axis within their grid areas. It specifies the alignment of grid items within the grid cells along the inline axis.
Example:

```
.container {
  display: grid;
  justify-items: center; /* or start, end, stretch */
}
```

Use Case:

Use justify-items to align grid items within their grid areas along the inline axis, ensuring consistent alignment across all items in the grid container.

Values:

- center: Items are centered within their grid areas.
- start: Items are aligned to the start of their grid areas.
- end: Items are aligned to the end of their grid areas.
- stretch: Items are stretched to fill their grid areas.

Notes:

This property applies to all grid items within the grid container.

7. align-items:

Definition:

Aligns grid items along the block (column) axis within their grid areas. It specifies the alignment of grid items within the grid cells along the block axis.
Example:

```
.container {
  display: grid;
  align-items: center; /* or start, end, stretch */
}
```

Use Case:

Use align-items to align grid items within their grid areas along the block axis, ensuring consistent alignment across all items in the grid container.

Values:

- center: Items are centered within their grid areas.
- start: Items are aligned to the start of their grid areas.
- end: Items are aligned to the end of their grid areas.
- stretch: Items are stretched to fill their grid areas.

Notes:

This property applies to all grid items within the grid container.

8. justify-content

Definition:

Aligns grid tracks (rows) within the grid container along the inline (row) axis. It specifies the alignment of grid tracks within the grid container along the inline axis.
Example:

```
.container {
  display: grid;
  justify-content: center; /* or start, end, space-between,
space-around, space-evenly */
}
```

Use Case:

Use justify-content to align grid tracks (rows) within the grid container along the inline axis, controlling the distribution of space between and around tracks.

Values:

- center: Tracks are centered within the grid container.
- start: Tracks are aligned to the start of the grid container.
- end: Tracks are aligned to the end of the grid container.
- space-between: Tracks are evenly distributed with the first track at the start and the last track at the end.
- space-around: Tracks are evenly distributed with equal space around them.
- space-evenly: Tracks are evenly distributed with equal space between them.

Notes:

This property applies to all grid tracks within the grid container.

9. align-content

Definition:

Aligns grid tracks (columns) within the grid container along the block (column) axis. It specifies the alignment of grid tracks within the grid container along the block axis.
Example:

```
.container {
  display: grid;
  align-content: center; /* or start, end, space-between,
space-around, space-evenly */
}
```

Use Case:

Use align-content to align grid tracks (columns) within the grid container along the block axis, controlling the distribution of space between and around tracks.

Values:

- center: Tracks are centered within the grid container.
- start: Tracks are aligned to the start of the grid container.
- end: Tracks are aligned to the end of the grid container.
- space-between: Tracks are evenly distributed with the first track at the start and the last track at the end.
- space-around: Tracks are evenly distributed with equal space around them.
- space-evenly: Tracks are evenly distributed with equal space between them.

Notes:

This property applies when there is extra space in the grid container along the block axis.

10. grid-auto-columns

Definition:

Specifies the size of implicitly created columns in the grid layout. It defines the size of columns that are created automatically to accommodate grid items that don't have an explicit grid placement.
Example:

```
.container {
  display: grid;
  grid-auto-columns: 100px; /* Column size */
}
```

Use Case:

Use grid-auto-columns to define the size of implicitly created columns in the grid layout, ensuring consistent sizing for automatically placed grid items.

Values:

100px: Fixed size for implicitly created columns in pixels.

Notes:

Implicitly created columns are those that are not explicitly defined using grid-template-columns.

11. grid-auto-rows

Definition:

Specifies the size of implicitly created rows in the grid layout. It defines the size of rows that are created automatically to accommodate grid items that don't have an explicit grid placement.
Example:

```
.container {
  display: grid;
  grid-auto-rows: 100px; /* Row size */ }
```

Use Case:

Use grid-auto-rows to define the size of implicitly created rows in the grid layout, ensuring consistent sizing for automatically placed grid items.

Values:

100px: Fixed size for implicitly created rows in pixels.

Notes:

Implicitly created rows are those that are not explicitly defined using grid-template-rows.

12. grid-auto-flow

Definition:

Specifies how implicitly created grid items are placed in the grid layout. It determines the direction in which implicitly created grid items are placed and whether they fill rows or columns first.
Example:

```
.container {
  display: grid;
  grid-auto-flow: column; /* or row, dense, column dense, row dense */
}
```

Use Case:

Use grid-auto-flow to control the placement of implicitly created grid items, ensuring flexibility and control over the grid layout.

Values:

- column: Implicit grid items are placed in columns, filling each column before moving to the next.
- row: Implicit grid items are placed in rows, filling each row before moving to the next.

- dense: Enables packing of grid items that are marked as "dense" into the grid, filling in any gaps left by previous items.

Notes:

This property affects the placement of grid items that do not have an explicit grid position.

13. justify-self

Definition:

Aligns a grid item along the inline (row) axis within its grid area. It specifies the alignment of a grid item within its grid cell along the inline axis.
Example:

```
.item {
  justify-self: center; /* or start, end, stretch */
}
```

Use Case:

Use justify-self to align a grid item within its grid area along the inline axis, providing fine-grained control over the alignment of individual grid items.

Values:

- center: The Item is centered within its grid area.
- start: The item is aligned to the start of its grid area.
- end: The item is aligned to the end of its grid area.
- stretch: The item is stretched to fill its grid area.

Notes:

This property applies to individual grid items within the grid container.

14. align-self

Definition:

Aligns a grid item along the block (column) axis within its grid area. It specifies the alignment of a grid item within its grid cell along the block axis.
Example:

```
.item {
  align-self: center; /* or start, end, stretch */
}
```

Use Case:

Use align-self to align a grid item within its grid area along the block axis, providing fine-grained control over the alignment of individual grid items.

Values:

- center: The Item is centered within its grid area.
- start: The item is aligned to the start of its grid area.
- end: The item is aligned to the end of its grid area.
- stretch: The item is stretched to fill its grid area.

Notes:

This property applies to individual grid items within the grid container.

15. grid-column

Definition:

This property is a shorthand for specifying a grid item's placement within the grid columns. It defines the starting and ending positions of a grid item within the grid columns.
Example:

```
.item {
  grid-column: 2 / span 3; /* Start at column line 2, span 3
columns */
}
```

Use Case:

Use grid-column to precisely position grid items within the grid columns, specifying both the start and end positions relative to the grid lines.

Values:

- start / end: Specifies the starting and ending positions of a grid item within the grid columns.
- span <number>: Indicates the number of columns the grid item should span. This value extends the grid item across multiple columns starting from the specified starting position.

Notes:

This shorthand property combines grid-column-start and grid-column-end.

16. grid-row-start

Definition:

Specifies the starting position of a grid item within the grid rows. It defines the line on which the grid item's top edge starts.
Example:

```
.item {
  grid-row-start: 2; /* Start at row line 2 */
}
```

Use Case:

Use grid-row-start to position the starting edge of a grid item within the grid rows, ensuring precise placement within the grid container.

Values:

<number>: Specifies the line on which the grid item's top edge starts within the grid rows. The number represents the line number in the grid row axis where the grid item begins.

Notes:

This property works in conjunction with grid-row-end to define the full extent of the grid item within the rows.

17. grid-row-end

Definition:

Specifies the ending position of a grid item within the grid rows. It defines the line on which the grid item's bottom edge ends.
Example:

```
.item {
  grid-row-end: span 2; /* End at row line 2, spanning 2 rows */
}
```

Use Case:

Use grid-row-end to position the ending edge of a grid item within the grid rows, defining the extent of the item's placement within the grid container.

Values:

- <number>: Specifies the line on which the grid item's bottom edge ends within the grid rows. The number represents the line number in the grid row axis where the grid item ends.
- span <number>: Indicates the number of rows the grid item should span. This value extends the grid item across multiple rows starting from the specified starting position.

Notes:

This property works in conjunction with grid-row-start to define the full extent of the grid item within the rows.

Conclusion

In conclusion, CSS Grid provides a robust and comprehensive layout system that empowers developers to create complex and flexible web designs. By mastering properties such as grid-template-columns, grid-template-rows, grid-template-areas, and others, developers gain fine-grained control over the structure and organization of elements within a grid container. With the ability to define column and row sizes, create named grid areas, and specify the placement of grid items, CSS Grid offers unparalleled flexibility in crafting visually appealing and responsive layouts.

Moreover, properties like grid-column, grid-row-start, and grid-row-end enable precise positioning of grid items within the grid layout, ensuring pixel-perfect alignment and distribution. The use of shorthand properties like grid-gap and grid-template further streamlines the process of creating complex grid layouts while enhancing code readability and maintainability. Ultimately, CSS Grid empowers developers to build modern, adaptive web layouts that seamlessly adapt to various screen sizes and devices, providing users with an optimal viewing experience across platforms. With its intuitive syntax and powerful features, CSS Grid remains a cornerstone of modern web design, revolutionizing the way developers approach layout creation on the web.

References

1. https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_grid_layout
2. https://developer.mozilla.org/en-US/docs/Web/CSS/align-content
3. https://developer.mozilla.org/en-US/docs/Web/CSS/justify-content