Grid in CSS

The CSS properties that allow you to use the CSS Grid capabilities

grid-area

In collection: **CSS Grid** Permalink Share Can I use MDN

Shorthand property for `grid-row-start` `grid-column-start` `grid-row-end` and `grid-column-end`.

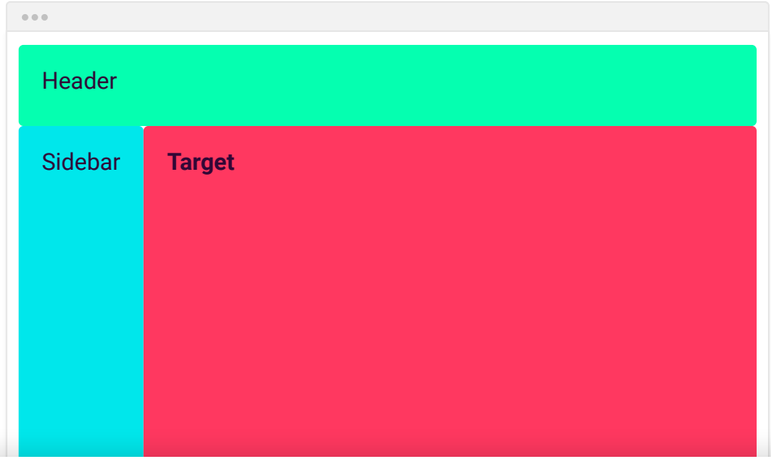
```
grid-area: auto;
```

default

The grid item's column and row starts and ends are automatically set.


```
grid-area: main;
```

You can use an **area name**.



grid-auto-columns

In collection: **CSS Grid** Permalink Share Can I use MDN

Defines the size of grid columns that were created *implicitly*: it means that `grid-auto-columns` targets the columns that were *not* defined with `grid-template-columns` or `grid-template-areas`.

```
grid-auto-columns: auto;
```

default

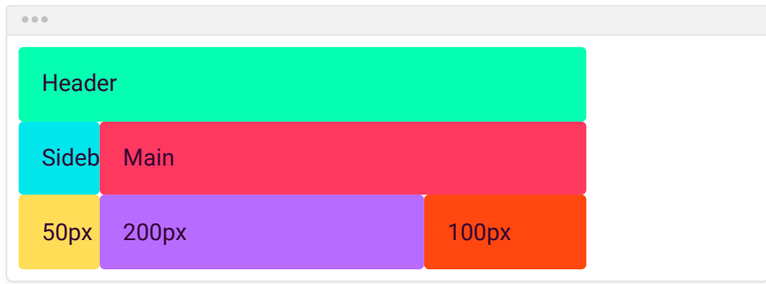
The implicity-created columns have an `auto` size.


```
grid-auto-columns: 100px;
```

Here we combine `grid-template-areas: "header header header" "sidebar main main"` with `grid-template-columns: 50px 200px`.

In this situation, the `grid-template-areas` defines **3 columns**, while the `grid-template-columns` only defines **2** column widths.

As a result, the third column width takes its value from the `grid-auto-columns` property: `100px`.

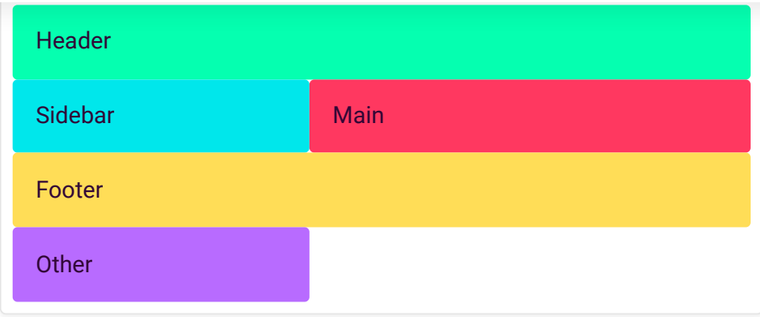


grid-auto-flow

In collection: **CSS Grid** Permalink Share Can I use MDN

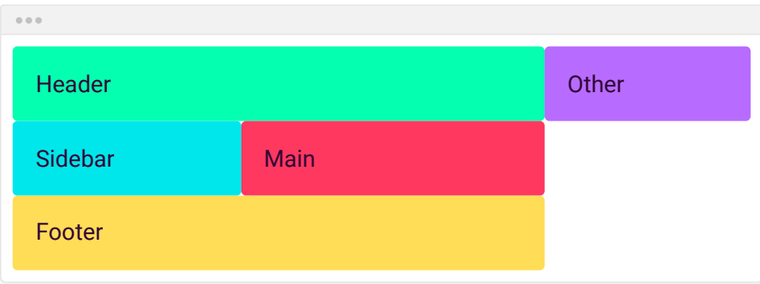
Defines the position of auto-generated **grid items**.

Here we have `grid-template-areas: "header header header" "sidebar main main" "footer footer footer"`, plus an additional "Other" grid item. The algorithm places it on its own **row**.



`grid-auto-flow: column;`

Here we have `grid-template-areas: "header header header" "sidebar main main" "footer footer footer"`, plus an additional "Other" grid item. The algorithm places it in its own **column**.

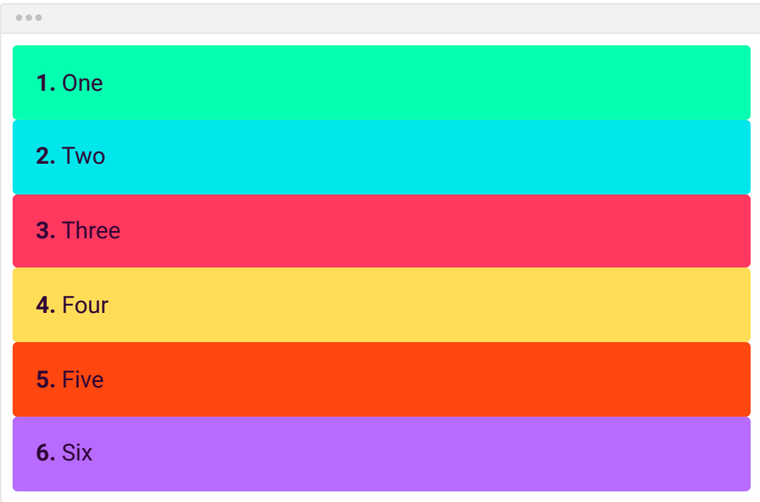


grid-auto-rows

Defines the size of grid rows that were created *implicitly*: it means that `grid-auto-rows` targets the rows that were *not* defined with `grid-template-rows` or `grid-template-areas`.

`grid-auto-rows: auto;` default

The implicity-created rows have an `auto` size.



`grid-auto-rows: 100px;`

cssreference.io

All properties Animations
Backgrounds Box model
Flexbox CSS Grid
Positioning Transitions
Typography

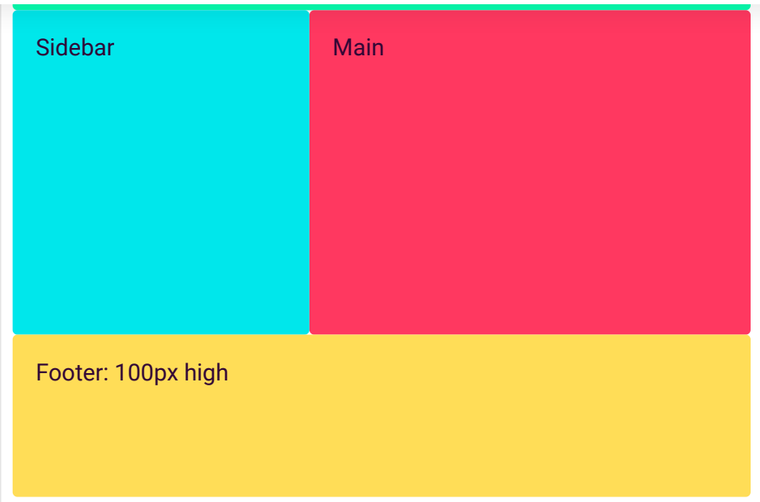
Search for a property

grid-area
grid-auto-columns
grid-auto-flow
grid-auto-rows
grid-column-end
grid-column-gap
grid-column-start
grid-column
grid-gap
grid-row-end
grid-row-gap
grid-row-start
grid-row
grid-template-areas
grid-template-columns
grid-template-rows
grid-template

footer footer" with `grid-template-rows: 50px 200px`.

In this situation, the `grid-template-areas` defines **3 rows**, while the `grid-template-rows` only defines **2** row heights.

As a result, the third row height (the footer) takes its value from the `grid-auto-rows` property: `100px`.



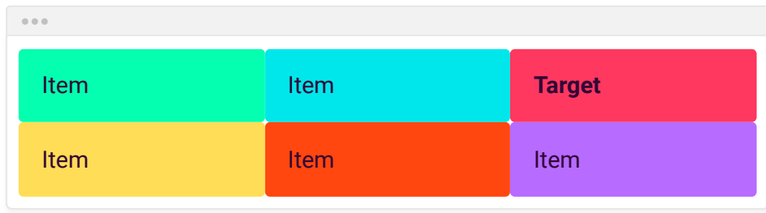
grid-column-end

In collection: **CSS Grid** Permalink Share Can I use MDN

Defines the **column end** position of a **grid item**.

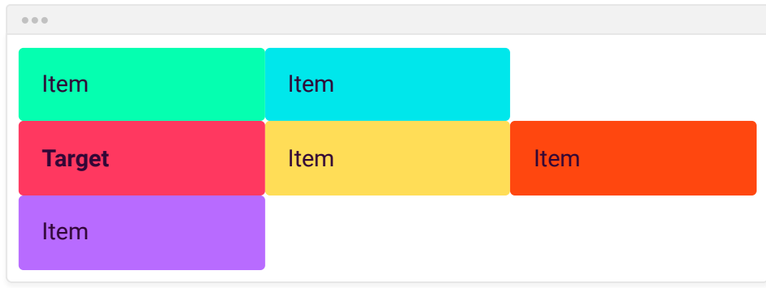
`grid-column-end: auto;` default

In this **3-column setup**, the grid item is automatically placed.



`grid-column-end: 2;`

The target grid item ends just before the **second** column.



`grid-column-end: 1;`

The target grid item ends just before the **first** column, which automatically creates a *fourth* column.

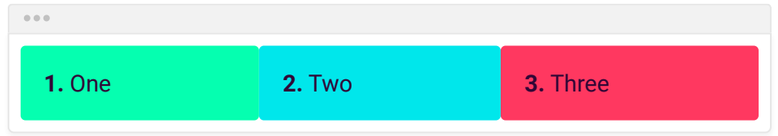
grid-column-gap

Defines the gutter between the columns of a **grid container**.

```
grid-column-gap: 0;
```

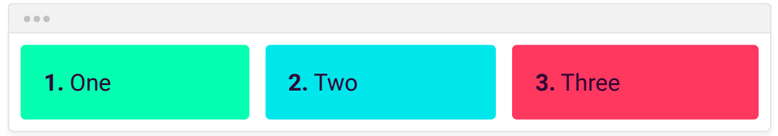
default

Removes the gap.



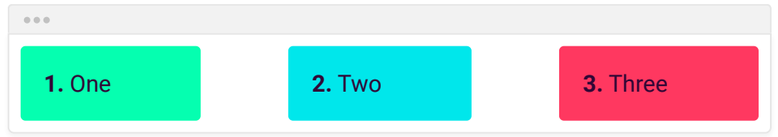

```
grid-column-gap: 10px;
```

You can use **pixel** values.



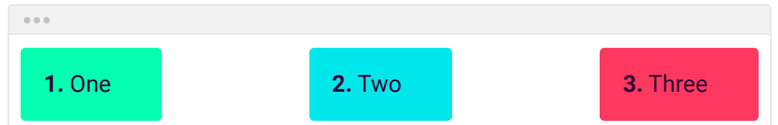

```
grid-column-gap: 3rem;
```

You can use **(r)em** values.




```
grid-column-gap: 20%;
```

You can use **percentage** values.



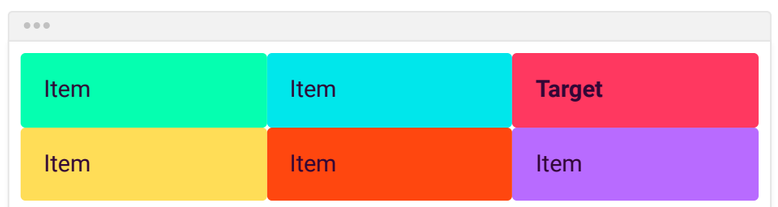
grid-column-start

Defines the **column start** position of a **grid item**.

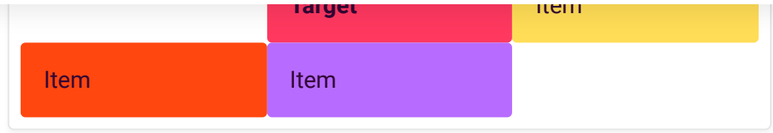
```
grid-column-start: auto;
```

default

In this **3-column setup**, the grid item is automatically placed.

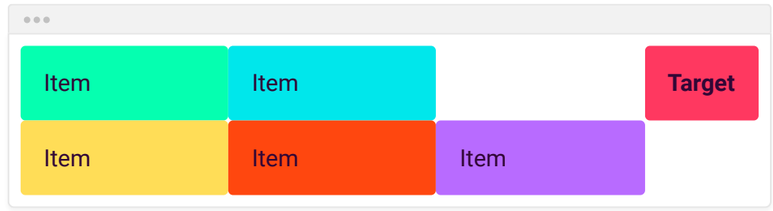



```
grid-column-start: 2;
```




```
grid-column-start: 4;
```

The target grid item is placed *outside* the grid, in an auto-generated **fourth** column.



grid-column

In collection: **CSS Grid** Permalink Share Can I use MDN

Shorthand property for `grid-column-start` and `grid-column-end`.

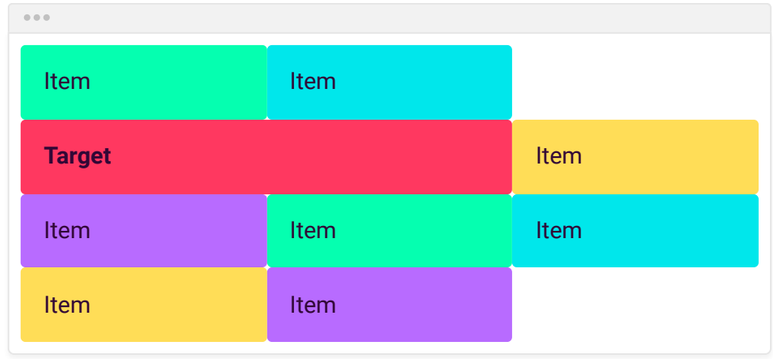
```
grid-column: auto auto;
```

default

The grid item's column start and end are automatically set.

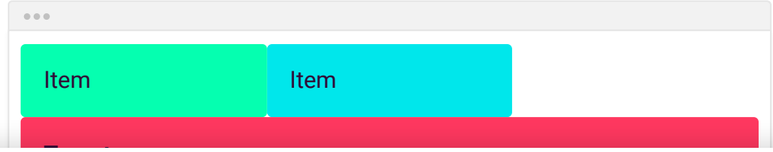

```
grid-column: 1 / 3;
```

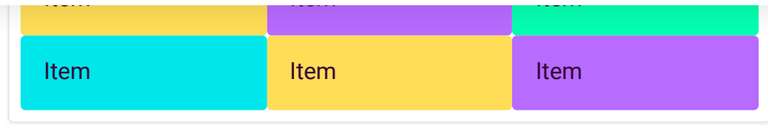
The grid item starts before the first column and ends just before the third one.




```
grid-column: span 3;
```

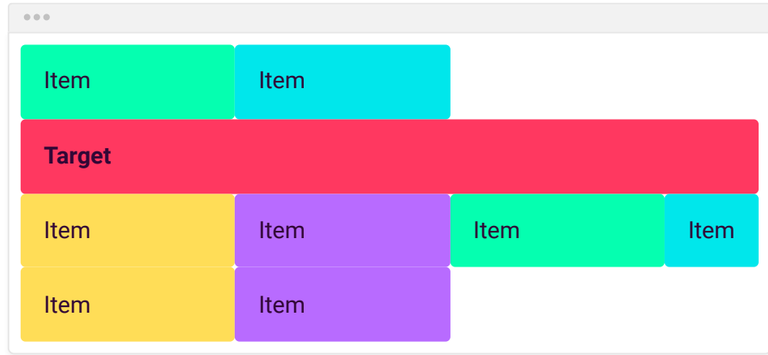
The grid item spans **3 columns**.






```
grid-column: 1 / span 4;
```

The grid items starts before the first column and spans for **4 columns**, creating a new one in the process.



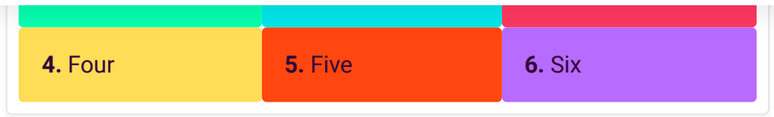

```
grid-column: main;
```

You can use an **area name** to "copy" its column start and end positions.

grid-gap

In collection: **CSS Grid** Permalink Share Can I use MDN

Shorthand property for `grid-row-gap` and `grid-column-gap`.




```
grid-gap: 10px;
```

You can set a **single value**.


```
grid-gap: 3rem 1rem;
```

You can set a value for **each** direction: rows first, columns second.

grid-row-end

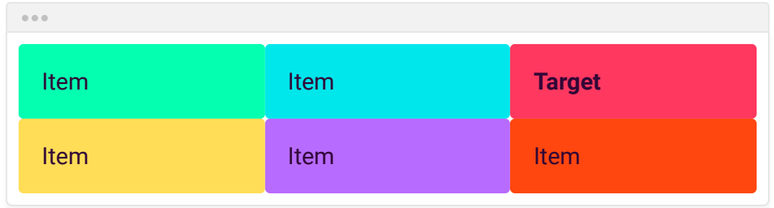
In collection: **CSS Grid** Permalink Share Can I use MDN

Defines the **row end** position of a **grid item**.

```
grid-row-end: auto;
```

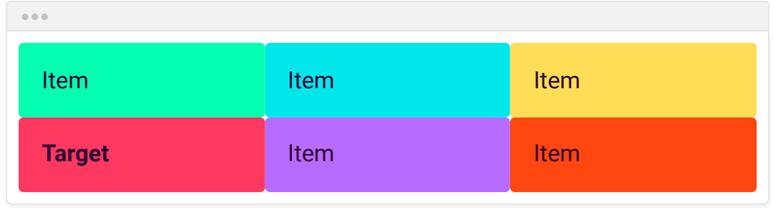
default

In this **3-column setup**, the grid item is automatically placed.

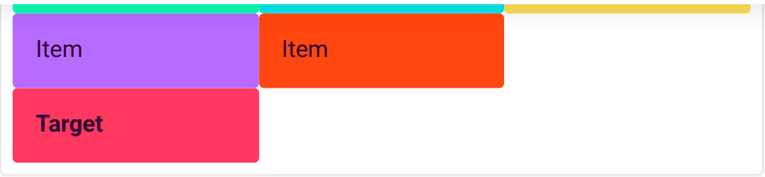



```
grid-row-end: 3;
```

The target grid item ends just before the **third** row.




```
grid-row-end: 4;
```



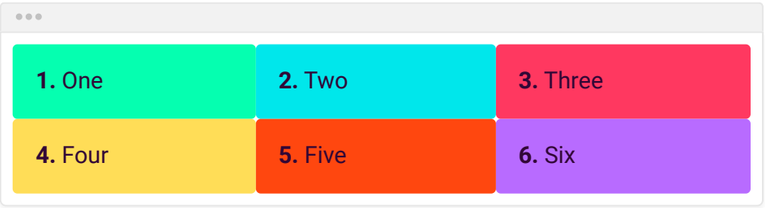
grid-row-gap

Defines the gutter between the rows of a **grid container**.

```
grid-row-gap: 0;
```

default

Removes the gap.




```
grid-row-gap: 10px;
```

You can use **pixel** values.


```
grid-row-gap: 3rem;
```

You can use **(r)em** values.

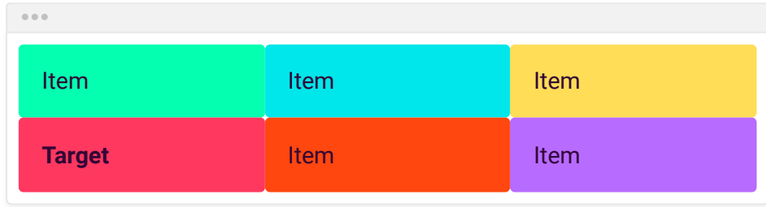
grid-row-start

Defines the **row start** position of a **grid item**.

In this **3-column setup**, the grid item is automatically placed on the first row.


```
grid-row-start: 2;
```

The target grid item is placed on the **second** row.




```
grid-row-start: 3;
```

The target grid item is placed *outside* the grid, in an auto-generated **third** row.

grid-row

In collection: **CSS Grid** Permalink Share Can I use MDN

Shorthand property for `grid-row-start` and `grid-row-end`.

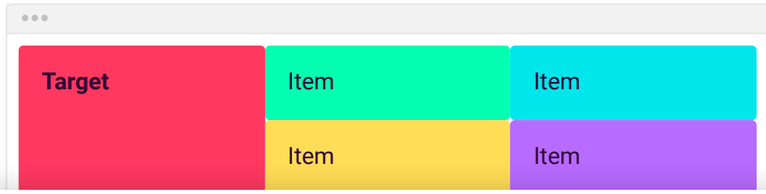
```
grid-row: auto auto;
```

default

The grid item's row start and end are automatically set.


```
grid-row: 1 / 3;
```

The grid item starts before the first row and ends just before the third one.



`grid-row: span 3;`

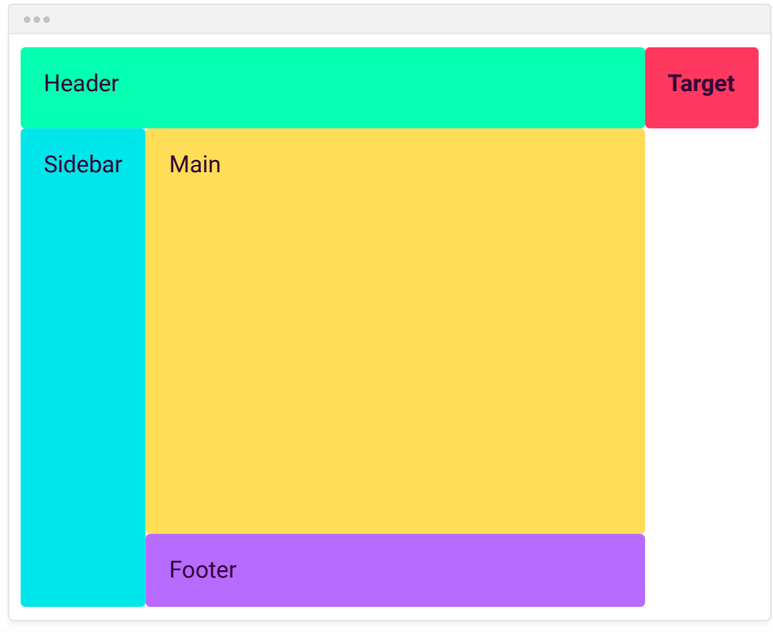
The grid item spans **3 rows**.

`grid-row: 1 / span 4;`

The grid items starts before the first row and spans for **4 rows**, creating a new one in the process.

`grid-row: header;`

You can use an **area name** to "copy" its row start and end positions.



grid-template-areas

Defines areas within a **grid container**. These areas can then be referenced when placing a **grid item**.

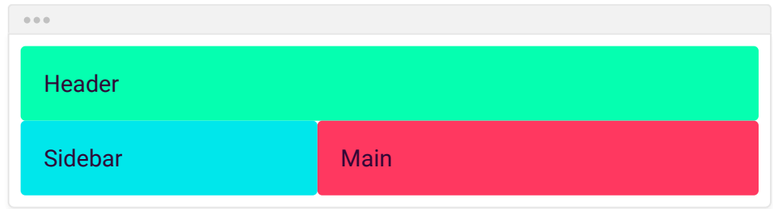
```
grid-template-areas: none;
```

default

No area is defined.


```
grid-template-areas: "header header header" "sidebar main main";
```

You can use **area names** to specify which cells each grid item should occupy.



grid-template-columns

Defines the columns of a **grid container**. You can specify the width of a column by using a keyword (like `auto`) or a length (like `10px`). The number of columns is determined by the number of **values** defined in the space-separated list.

```
grid-template-columns: none;
```

default

No columns are defined, so you only have one.

cssreference.io

All properties Animations
Backgrounds Box model
Flexbox CSS Grid
Positioning Transitions
Typography

Search for a property

grid-area
grid-auto-columns
grid-auto-flow
grid-auto-rows
grid-column-end
grid-column-gap
grid-column-start
grid-column
grid-gap
grid-row-end
grid-row-gap
grid-row-start
grid-row
grid-template-areas
grid-template-columns
grid-template-rows
grid-template


```
grid-template-columns: auto auto auto;
```

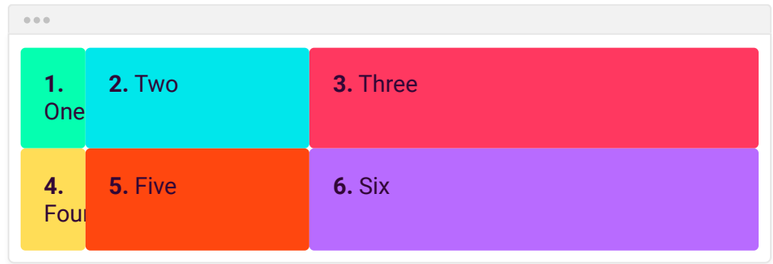
You can use the keyword `auto` so that each column resizes itself automatically.


```
grid-template-columns: 80px auto 1rem;
```

You can mix the **units**.


```
grid-template-columns: 40px 1fr 2fr;
```

You can use the `fr` **flex unit** to distribute the **remaining space** across all flex columns.



grid-template-rows

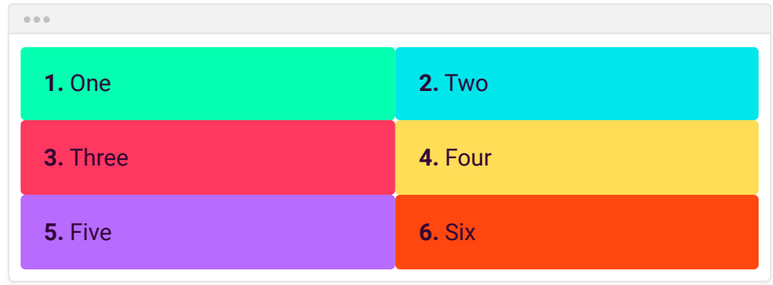
In collection: **CSS Grid** Permalink Share Can I use MDN

Defines the rows of a **grid container**. You can specify the width of a row by using a keyword (like `auto`) or a length (like `10px`). The number of rows is determined by the number of **values** defined in the space-separated list.

```
grid-template-rows: none;
```

default

No rows are defined.




```
grid-template-rows: 120px auto 3rem;
```


```
grid-template-rows: 20px 1fr 2fr;
```

You can use the `fr` **flex unit** to distribute the **remaining space** across all flex rows.

grid-template

In collection: **CSS Grid** Permalink Share Can I use MDN

Shorthand property for `grid-template-rows` `grid-template-columns` and `grid-template-area`.

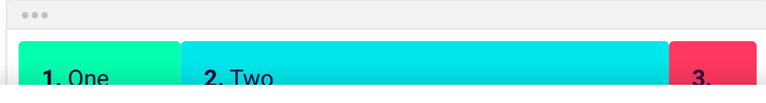
```
grid-template: none;
```

default

No rows, columns, or areas are defined.

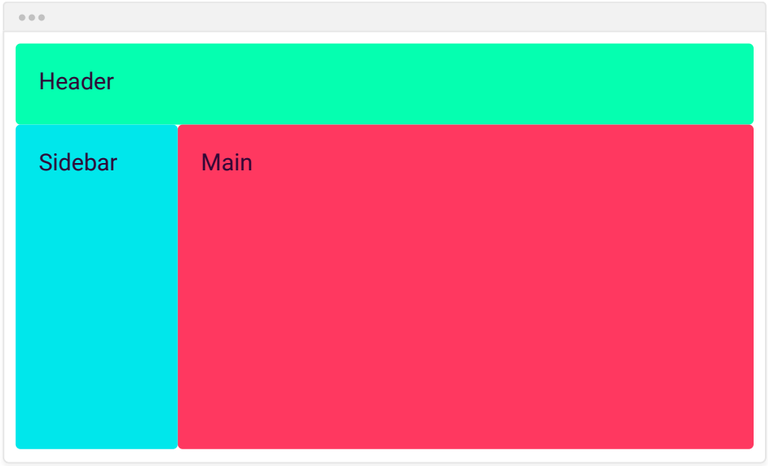

```
grid-template: 200px 1fr / 100px auto 3rem;
```

You can define **rows** first, **columns** second, by splitting them with a slash `/`




```
grid-template: "header header header" 50px "sidebar main main" 200px / 100px auto;
```

You can define both **areas** *and* **rows** and **columns** dimensions. In this case, each set of areas have a row size attached to it. The first row is `50px` high, the second one is `200px` high. The sidebar column is `100px` wide, while the main column's width is set to `auto` and takes up the remaining width.



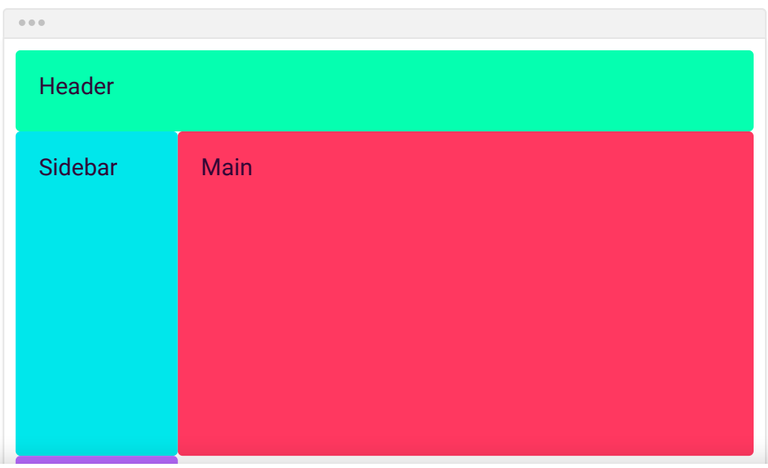
grid

In collection: **CSS Grid** Permalink Share Can I use MDN

Shorthand property for `grid-template-rows` `grid-template-columns` `grid-template-areas` `grid-auto-rows` `grid-auto-columns` and `grid-auto-flow`.

```
grid: "header header header" 50px "sidebar main main" 200px / 100px auto;
```

You can use it as `grid-template` by setting all **explicit** rows, columns, and areas.



```
grid: 200px 100px / auto-flow 30%;
```

You can combine `grid-template-rows` with `grid-auto-columns`.


```
grid: auto-flow 50px / 200px 100px;
```

You can combine `grid-auto-rows` with `grid-template-columns`.

codecademy

Grid

Grid Template Columns

To specify the number of columns of the grid and the widths of each column, the CSS property `grid-template-columns` is used on the grid container. The number of width values determines the number of columns and each width value can be either in pixels(`px`) or percentages(%).

```
#grid-container {
  display: grid;
  width: 100px;
  grid-template-columns: 20px 20% 60%;
}
```

fr Relative Unit

The CSS grid relative sizing unit `fr` is used to split rows and/or columns into proportional distances. Each `fr` unit is a fraction of the grid's overall length and width. If a fixed unit is used along with `fr` (like pixels for example), then the `fr` units will only be proportional to the distance left over.

```
/*
In this example, the second column take
60px of the avaiable 100px so the first
and third columns split the remaining
available 40px into two parts (`1fr`
= 50% or 20px)
*/

.grid {
  display: grid;
  width: 100px;
  grid-template-columns: 1fr 60px 1fr;
}
```

Grid Gap

The CSS `grid-gap` property is a shorthand way of setting the two properties `grid-row-gap` and `grid-column-gap` . It is used to determine the size of the gap between each row and each column. The first value sets the size of the gap between rows and while the second value sets the size of the gap between columns.

```
// The distance between rows is 20px
// The distance between columns is 10px

#grid-container {
  display: grid;
  grid-gap: 20px 10px;
}
```

CSS Block Level Grid

CSS Grid is a two-dimensional CSS layout system. To set an HTML element into a block-level *grid container* use `display: grid` property/value. The nested elements inside this element are called *grid items*.

```
#grid-container {
  display: grid;
}
```

CSS grid-row

The CSS `grid-row` property is shorthand for the `grid-row-start` and `grid-row-end` properties specifying a grid item's size and location within the grid row. The starting and ending row values are separated by a `/`. There is a corresponding `grid-column` property shorthand that implements the same behavior for columns.

```
/*CSS Syntax */
grid-row: grid-row-start / grid-row-end;

/*Example*/
.item {
  grid-row: 1 / span 2;
}
```

CSS Inline Level Grid

CSS Grid is a two-dimensional CSS layout system. To set an HTML element into a inline-level *grid container* use `display: inline-grid` property/value. The nested elements inside this element are called *grid items*.

The difference between the values `inline-grid` and `grid` is that the `inline-grid` will make the element inline while `grid` will make it a block-level element.

```
#grid-container {
  display: inline-grid;
}
```

`minmax()` Function

The CSS Grid `minmax()` function accepts two parameters:

- The first parameter is the minimum size of a row or column.
- The second parameter is the maximum size.

The grid must have a variable width for the `minmax()` function.

If the maximum value is less than the minimum, then the maximum value is ignored and only the minimum value is used.

The function can be used in the values of the `grid-template-rows`, `grid-template-columns` and `grid-template` properties.

```
/* In this example, the second column
will vary in size between 100px and 500px
depending on the size of the web browser"
*/

.grid {
  display: grid;
  grid-template-columns: 100px
minmax(100px, 500px) 100px;
}
```

grid-row-start & grid-row-end

The CSS `grid-row-start` and `grid-row-end` properties allow single grid items to take up multiple rows. The `grid-row-start` property defines on which row-line the item will start. The `grid-row-end` property defines how many rows an item will span, or on which row-line the item will end. The keyword `span` can be used with either property to automatically calculate the ending value from the starting value or vice versa. There are complementary `grid-column-start` and `grid-column-end` properties that apply the same behavior to columns.

```
/* CSS syntax:
grid-row-start: auto|row-line;
grid-row-end: auto|row-line|span n;
*/
grid-row-start: 2;
grid-row-end: span 2;
```

CSS grid-row-gap

The CSS `grid-row-gap` property determines the amount of blank space between each row in a CSS grid layout or in other words, sets the size of the gap (gutter) between an element's grid rows. The `grid-column-gap` provides the same functionality for space between grid columns.

```
/*CSS Syntax */
grid-row-gap: length; /*Any legal length value, like px or %. 0 is the default value*/
```

CSS grid-area

The CSS `grid-area` property specifies a grid item's size and location in a grid layout and is a shorthand property for the `grid-row-start`, `grid-column-start`, `grid-row-end`, and `grid-column-end` in that order. Each value is separated by a `/`.

In the included example, `Item1` will start on row 2 and column 1, and span 2 rows and 3 columns

```
.item1 {
  grid-area: 2 / 1 / span 2 / span 3;
}
```

Align Self

The CSS `align-self` property is used to set how an individual grid item positions itself along the column or block axis. By default grid items inherit the value of the `align-items` property on the container. So if the `align-self` value is set, it would over-ride the inherited `align-items` value.

The value `start` positions grid items on the top of the grid area.

The value `end` aligns the grid on the bottom of the grid area.

The value `center` positions grid items on the center of the grid area.

The value `stretch` positions grid items to fill the grid area (default).

Justify Items

The `justify-items` property is used on a grid container. It's used to determine how the grid items are spread out along a row by setting the default `justify-self` property for all child boxes.

The value `start` aligns grid items to the left side of the grid area.

The value `end` aligns grid items to the right side of the grid area.

The value `center` aligns grid items to the center of the grid area.

The value `stretch` stretches all items to fill the grid area.

```
#container {
  display: grid;
  justify-items: center;
  grid-template-columns: 1fr;
  grid-template-rows: 1fr 1fr 1fr;
  grid-gap: 10px;
}
```

CSS grid-template-areas

The CSS `grid-template-areas` property allows the naming of sections of a webpage to use as values in the `grid-row-start`, `grid-row-end`, `grid-column-start`, `grid-column-end`, and `grid-area` properties. They specify named grid areas within a CSS grid.

```
/* Specify two rows, where "item" spans
the first two columns in the first two
rows (in a four column grid layout)*/
.item {
  grid-area: nav;
}
.grid-container {
  display: grid;
  grid-template-areas:
    'nav nav . .'
    'nav nav . .';
}
```

CSS grid-auto-flow

The CSS `grid-auto-flow` property specifies whether implicity-added elements should be added as rows or columns within a grid or, in other words, it controls how auto-placed items get inserted in the grid and this property is declared on the grid container.

The value `row` specifies the new elements should fill rows from left to right and create new rows when there are too many elements (default).

The value `column` specifies the new elements should fill columns from top to bottom and create new columns when there are too many elements.

The value `dense` invokes an algorithm that attempts to fill holes earlier in the grid layout if smaller elements are added.

```
/*CSS Syntax */
grid-auto-flow: row|column|dense|row
dense|column dense;
```

Justify Content

Sometimes the total size of the grid items can be smaller than the grid container. If this is the case, the CSS property `justify-content` can be used to position the entire grid along the row or inline axis of the grid container.

The value `start` aligns the grid to the left side of the grid container.

The value `end` aligns the grid to the right side of the grid container.

The value `center` centers the grid horizontally in the grid container.

The value `stretch` stretches the grid items to increase the size of the grid to expand horizontally across the container.

The value `space-around` includes an equal amount of space on each side of a grid element, resulting in double the amount of space between elements as there is before the first and after the last element.

The value `space-between` includes an equal amount of space between grid items and no space at either end.

The value `space-evenly` places an even amount of space between grid items and at either end.

Align Content

Some times the total size of the grid items can be smaller than the grid container. If this is the case, the CSS property `align-content` can be used to position the entire grid along the column axis of the grid container.
The property is declared on the grid container.

The value `start` aligns the grid to the top of the grid container.

The value `end` aligns the grid to the bottom of the grid container.

The value `center` centers the grid vertically in the grid container.

The value `stretch` stretches the grid items to increase the size of the grid to expand vertically across the container.

The value `space-around` includes an equal amount of space on each side of a grid element, resulting in double the amount of space between elements as there is before the first and after the last element.

The value `space-between` includes an equal amount of space between grid items and no space at either end.

The value `space-evenly` places an even amount of space between grid items and at either end.

CSS grid-auto-rows

The CSS `grid-auto-rows` property specifies the height of implicitly added grid rows or it sets a size for the rows in a grid container. This property is declared on the grid container. `grid-auto-columns` provides the same functionality for columns. Implicitly-added rows or columns occur when there are more grid items than cells available.

Justify Self

The CSS `justify-self` property is used to set how an individual grid item positions itself along the row or inline axis. By default grid items inherit the value of the `justify-items` property on the container. So if the `justify-self` value is set, it would override the inherited `justify-items` value.

The value `start` positions grid items on the left side of the grid area.

The value `end` positions the grid items on the right side of the grid area.

The value `center` positions grid items on the center of the grid area.

The value `stretch` positions grid items to fill the grid area (default).

```
// The grid items are positioned to the right (end) of the row.

#grid-container {
  display: grid;
  justify-items: start;
}

.grid-items {
  justify-self: end;
}
```

CSS grid-area

The CSS `grid-area` property allows for elements to overlap each other by using the `z-index` property on a particular element which tells the browser to render that element on top of the other elements.

Align Items

The `align-items` property is used on a grid container. It's used to determine how the grid items are spread out along the column by setting the default `align-self` property for all child grid items.

The value `start` aligns grid items to the top side of the grid area.

The value `end` aligns grid items to the bottom side of the grid area.

The value `center` aligns grid items to the center of the grid area.

The value `stretch` stretches all items to fill the grid area.

```
#container {
  display: grid;
  align-items: start;
  grid-template-columns: 1fr;
  grid-template-rows: 1fr 1fr 1fr;
  grid-gap: 10px;
}
```

Grid

CSS Block Level Grid

CSS Grid is a two-dimensional CSS layout system. To set an HTML element into a block-level *grid container* use `display: grid` property/value. The nested elements inside this element are called *grid items*.

```
#grid-container {
  display: grid;
}
```

CSS Inline Level Grid

CSS Grid is a two-dimensional CSS layout system. To set an HTML element into a inline-level *grid container* use `display: inline-grid` property/value. The nested elements inside this element are called *grid items*.
The difference between the values `inline-grid` and `grid` is that the `inline-grid` will make the element inline while `grid` will make it a block-level element.

```
#grid-container {
  display: inline-grid;
}
```

Grid Template Columns

To specify the number of columns of the grid and the widths of each column, the CSS property `grid-template-columns` is used on the grid container. The number of width values determines the number of columns and each width value can be either in pixels(`px`) or percentages(%).

```
#grid-container {
  display: grid;
  width: 100px;
  grid-template-columns: 20px 20% 60%;
}
```

fr Relative Unit

The CSS grid relative sizing unit `fr` is used to split rows and/or columns into proportional distances. Each `fr` unit is a fraction of the grid's overall length and width. If a fixed unit is used along with `fr` (like pixels for example), then the `fr` units will only be proportional to the distance left over.

```
/*
In this example, the second column take 60px of the avaiable 100px so the first and
third columns split the remaining available 40px into two parts (`1fr` = 50% or 20px)
*/

.grid {
  display: grid;
  width: 100px;
  grid-template-columns: 1fr 60px 1fr;
}
```

`minmax()` Function

The CSS Grid `minmax()` function accepts two parameters:

- The first parameter is the minimum size of a row or column.
- The second parameter is the maximum size.

The grid must have a variable width for the `minmax()` function.
If the maximum value is less than the minimum, then the maximum value is ignored and only the minimum value is used.
The function can be used in the values of the `grid-template-rows`, `grid-template-columns` and `grid-template` properties.

```
/* In this example, the second column will vary in size between 100px and 500px
depending on the size of the web browser" */

.grid {
  display: grid;
  grid-template-columns: 100px minmax(100px, 500px) 100px;
}
```

CSS grid-row-gap

The CSS `grid-row-gap` property determines the amount of blank space between each row in a CSS grid layout or in other words, sets the size of the gap (gutter) between an element's grid rows. The `grid-column-gap` provides the same functionality for space between grid columns.

```
/*CSS Syntax */
grid-row-gap: length; /*Any legal length value, like px or %. 0 is the default value*/
```

Grid Gap

The CSS `grid-gap` property is a shorthand way of setting the two properties `grid-row-gap` and `grid-column-gap`. It is used to determine the size of the gap between each row and each column. The first value sets the size of the gap between rows and while the second value sets the size of the gap between columns.

```
// The distance between rows is 20px
// The distance between columns is 10px

#grid-container {
  display: grid;
  grid-gap: 20px 10px;
}
```

grid-row-start & grid-row-end

The CSS `grid-row-start` and `grid-row-end` properties allow single grid items to take up multiple rows. The `grid-row-start` property defines on which row-line the item will start. The `grid-row-end` property defines how many rows an item will span, or on which row-line the item will end. The keyword `span` can be used with either property to automatically calculate the ending value from the starting value or vice versa. There are complementary `grid-column-start` and `grid-column-end` properties that apply the same behavior to columns.

```
/* CSS syntax:
grid-row-start: auto|row-line;
grid-row-end: auto|row-line|span n;
*/
grid-row-start: 2;
grid-row-end: span 2;
```

CSS grid-row

The CSS `grid-row` property is shorthand for the `grid-row-start` and `grid-row-end` properties specifying a grid item's size and location within the grid row. The starting and ending row values are separated by a `/`. There is a corresponding `grid-column` property shorthand that implements the same behavior for columns.

```
/*CSS Syntax */
grid-row: grid-row-start / grid-row-end;

/*Example*/
.item {
  grid-row: 1 / span 2;
}
```

CSS grid-area

The CSS `grid-area` property specifies a grid item's size and location in a grid layout and is a shorthand property for the `grid-row-start`, `grid-column-start`, `grid-row-end`, and `grid-column-end` in that order. Each value is separated by a `/`. In the included example, `Item1` will start on row 2 and column 1, and span 2 rows and 3 columns

```
.item1 {
  grid-area: 2 / 1 / span 2 / span 3;
}
```

CSS grid-template-areas

The CSS `grid-template-areas` property allows the naming of sections of a webpage to use as values in the `grid-row-start`, `grid-row-end`, `grid-column-start`, `grid-column-end`, and `grid-area` properties. They specify named grid areas within a CSS grid.

```
/* Specify two rows, where "item" spans the first two columns in the first two rows (in a four column grid layout)*/
.item {
  grid-area: nav;
}
.grid-container {
  display: grid;
  grid-template-areas:
    'nav nav . .'
    'nav nav . .';
}
```

CSS grid-area

The CSS `grid-area` property allows for elements to overlap each other by using the `z-index` property on a particular element which tells the browser to render that element on top of the other elements.

Justify Items

The `justify-items` property is used on a grid container. It's used to determine how the grid items are spread out along a row by setting the default `justify-self` property for all child boxes.
The value `start` aligns grid items to the left side of the grid area.
The value `end` aligns grid items to the right side of the grid area.
The value `center` aligns grid items to the center of the grid area.
The value `stretch` stretches all items to fill the grid area.

```
#container {
  display: grid;
  justify-items: center;
  grid-template-columns: 1fr;
  grid-template-rows: 1fr 1fr 1fr;
  grid-gap: 10px;
}
```

Justify Content

Sometimes the total size of the grid items can be smaller than the grid container. If this is the case, the CSS property `justify-content` can be used to position the entire grid along the row or inline axis of the grid container.
The value `start` aligns the grid to the left side of the grid container.
The value `end` aligns the grid to the right side of the grid container.
The value `center` centers the grid horizontally in the grid container.
The value `stretch` stretches the grid items to increase the size of the grid to expand horizontally across the container.
The value `space-around` includes an equal amount of space on each side of a grid element, resulting in double the amount of space between elements as there is before the first and after the last element.
The value `space-between` includes an equal amount of space between grid items and no space at either end.
The value `space-evenly` places an even amount of space between grid items and at either end.

Align Items

The `align-items` property is used on a grid container. It's used to determine how the grid items are spread out along the column by setting the default `align-self` property for all child grid items.
The value `start` aligns grid items to the top side of the grid area.
The value `end` aligns grid items to the bottom side of the grid area.
The value `center` aligns grid items to the center of the grid area.
The value `stretch` stretches all items to fill the grid area.

```
#container {
  display: grid;
  align-items: start;
  grid-template-columns: 1fr;
  grid-template-rows: 1fr 1fr 1fr;
  grid-gap: 10px;
}
```

Align Content

Some times the total size of the grid items can be smaller than the grid container. If this is the case, the CSS property `align-content` can be used to position the entire grid along the column axis of the grid container.
The property is declared on the grid container.
The value `start` aligns the grid to the top of the grid container.
The value `end` aligns the grid to the bottom of the grid container.
The value `center` centers the grid vertically in the grid container.
The value `stretch` stretches the grid items to increase the size of the grid to expand vertically across the container.
The value `space-around` includes an equal amount of space on each side of a grid element, resulting in double the amount of space between elements as there is before the first and after the last element.
The value `space-between` includes an equal amount of space between grid items and no space at either end.
The value `space-evenly` places an even amount of space between grid items and at either end.

Justify Self

The CSS `justify-self` property is used to set how an individual grid item positions itself along the row or inline axis. By default grid items inherit the value of the `justify-items` property on the container. So if the `justify-self` value is set, it would over-ride the inherited `justify-items` value.
The value `start` positions grid items on the left side of the grid area.
The value `end` positions the grid items on the right side of the grid area.
The value `center` positions grid items on the center of the grid area.
The value `stretch` positions grid items to fill the grid area (default).

```
// The grid items are positioned to the right (end) of the row.

#grid-container {
  display: grid;
  justify-items: start;
}

.grid-items {
  justify-self: end;
}
```

Align Self

The CSS `align-self` property is used to set how an individual grid item positions itself along the column or block axis. By default grid items inherit the value of the `align-items` property on the container. So if the `align-self` value is set, it would over-ride the inherited `align-items` value.
The value `start` positions grid items on the top of the grid area.
The value `end` aligns the grid on the bottom of the grid area.
The value `center` positions grid items on the center of the grid area.
The value `stretch` positions grid items to fill the grid area (default).

CSS grid-auto-rows

The CSS `grid-auto-rows` property specifies the height of implicitly added grid rows or it sets a size for the rows in a grid container. This property is declared on the grid container. `grid-auto-columns` provides the same functionality for columns. Implicitly-added rows or columns occur when there are more grid items than cells available.

CSS grid-auto-flow

The CSS `grid-auto-flow` property specifies whether implicity-added elements should be added as rows or columns within a grid or, in other words, it controls how auto-placed items get inserted in the grid and this property is declared on the grid container.
The value `row` specifies the new elements should fill rows from left to right and create new rows when there are too many elements (default).
The value `column` specifies the new elements should fill columns from top to bottom and create new columns when there are too many elements.
The value `dense` invokes an algorithm that attempts to fill holes earlier in the grid layout if smaller elements are added.

```
/*CSS Syntax */
grid-auto-flow: row|column|dense|row dense|column dense;
```