# CSS Grid Layout Module

## Grid Layout

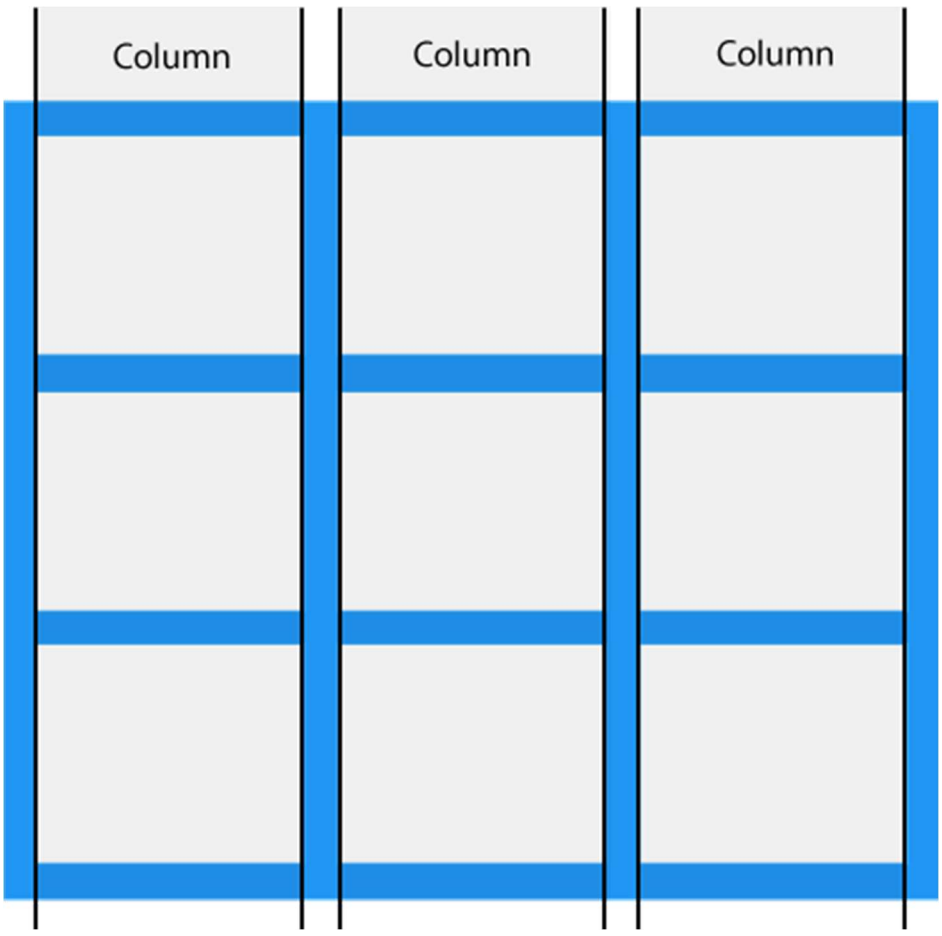
The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

```
<!DOCTYPE html>
<html>
<head>
<style>
.item1 { grid-area: header; }
.item2 { grid-area: menu; }
.item3 { grid-area: main; }
.item4 { grid-area: right; }
.item5 { grid-area: footer; }

.grid-container {
  display: grid;
  grid-template-areas:
    'header header header header header header'
    'menu main main main right right'
    'menu footer footer footer footer footer';
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
</style>
</head>
<body>

<h1>Grid Layout</h1>

<p>This grid layout contains six columns and three rows:</p>

<div class="grid-container">
  <div class="item1">Header</div>
  <div class="item2">Menu</div>
  <div class="item3">Main</div>
```

```
  <div class="item4">Right</div>
  <div class="item5">Footer</div>
</div>

</body>
</html>
```

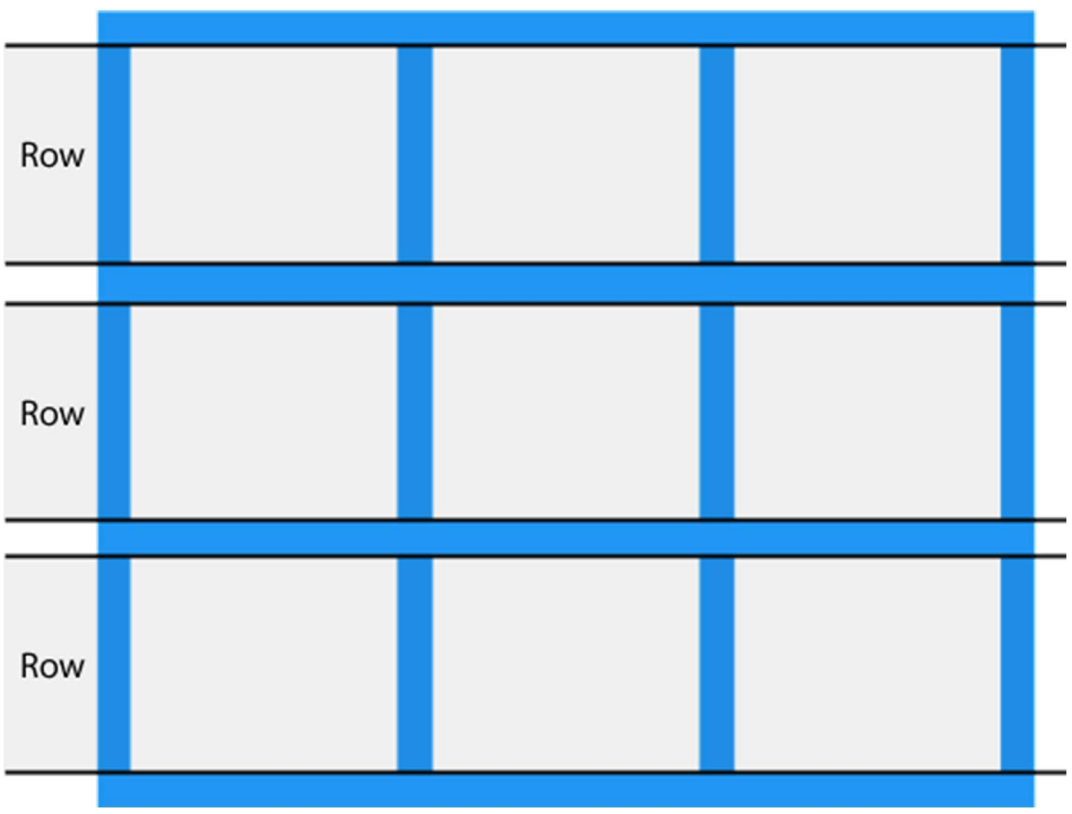
# Grid Columns

The vertical lines of grid items are called *columns*.



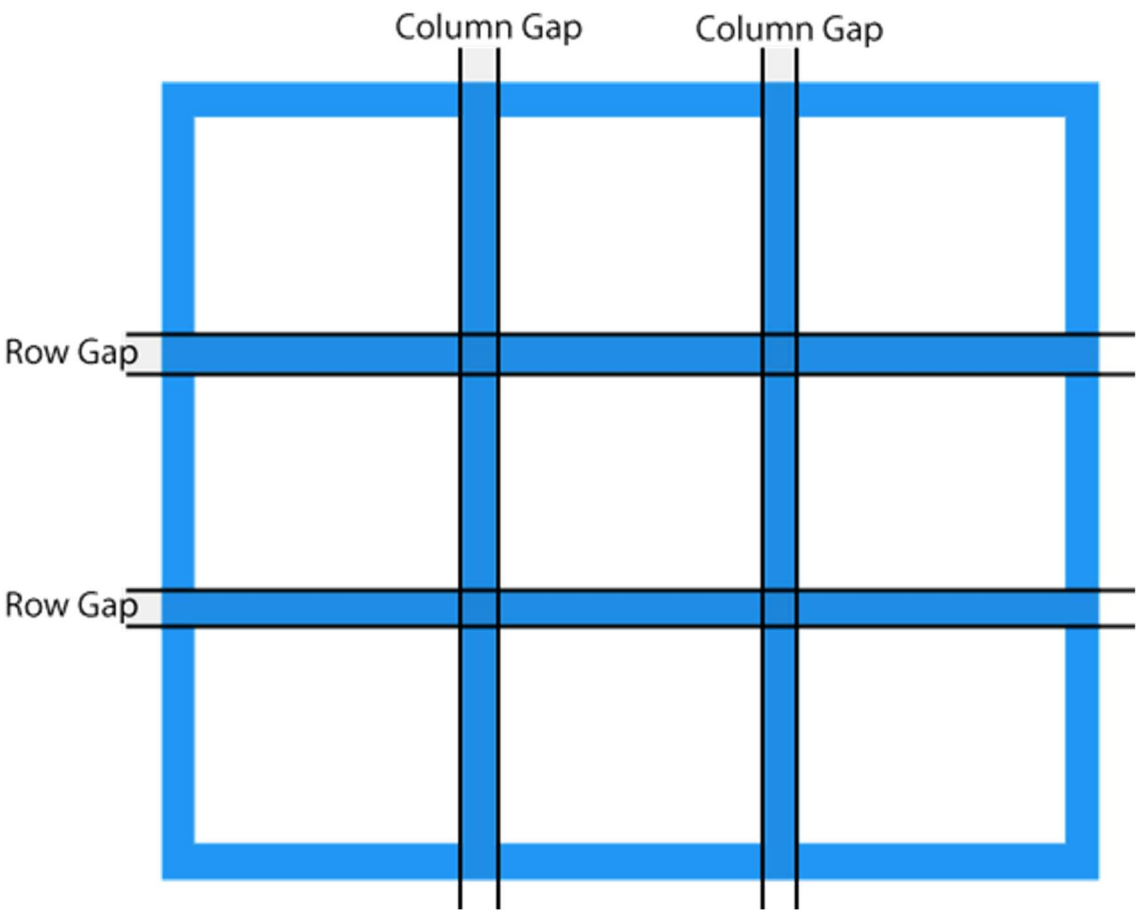

# Grid Rows

The horizontal lines of grid items are called *rows*.



# Grid Gaps

The spaces between each column/row are called *gaps*.



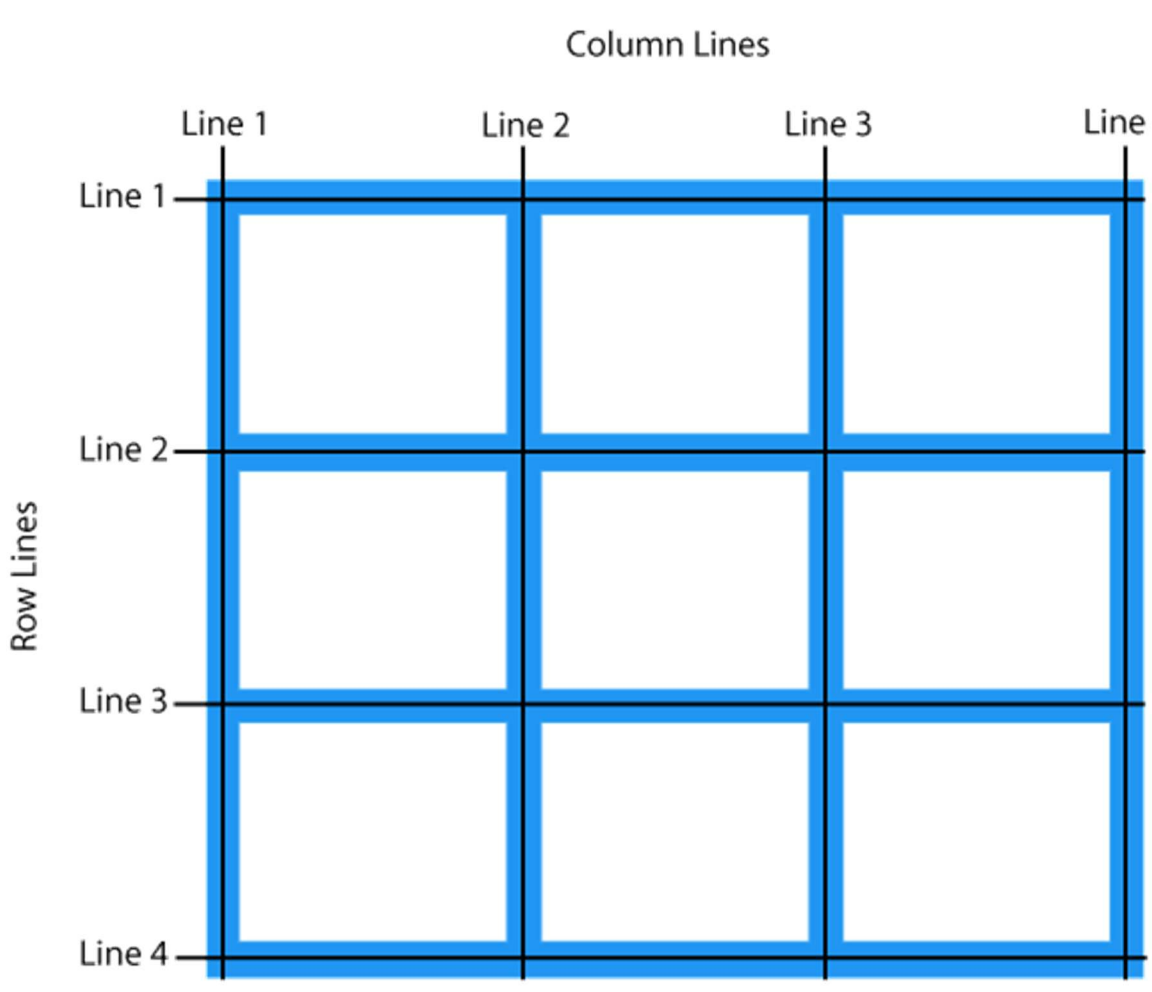

You can adjust the gap size by using one of the following properties:

- column-gap
- row-gap
- gap

# Grid Lines

The lines between columns are called *column lines*.

The lines between rows are called *row lines*.



# All CSS Grid Properties

| Property | Description |
|---|---|
| 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-area | Either specifies a name for the grid item, or this property is a shorthand property for the *grid-row-start*, *grid-column-start*, *grid-row-end*, and *grid-column-end* 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-column | A shorthand property for the *grid-column-start* and the *grid-column-end* properties |
| grid-column-end | Specifies where to end the grid item |
| grid-column-gap | Specifies the size of the gap between columns |
| grid-column-start | Specifies where to start the grid item |
| grid-gap | A shorthand property for the *grid-row-gap* and *grid-column-gap* properties |
| grid-row | A shorthand property for the *grid-row-start* and the *grid-row-end* properties |
| grid-row-end | Specifies where to end the grid item |
| grid-row-gap | Specifies the size of the gap between rows |

| | |
|---|---|
| grid-row-start | Specifies where to start the grid item |
| 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 |
| row-gap | Specifies the gap between the grid rows |

SOME EXAMPLES

```
<!DOCTYPE html>
<html>
<head>
<style>
.grid-container {
  display: grid;
  gap: 50px;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}

.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
</style>
</head>
<body>

<h1>The gap Property:</h1>

<p>Use the <em>gap</em> shorthand property to adjust the space between the columns and rows:</p>
```

```
<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 class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}

.item1 {
  grid-column-start: 1;
  grid-column-end: 3;
}
</style>
</head>
<body>

<h1>Grid Lines</h1>
```

```
<div class="grid-container">
  <div class="item1">1</div>
  <div class="item2">2</div>
  <div class="item3">3</div>
  <div class="item4">4</div>
  <div class="item5">5</div>
  <div class="item6">6</div>
  <div class="item7">7</div>
  <div class="item8">8</div>
</div>

<p>You can refer to line numbers when placing grid items.</p>

</body>
</html>
```

# Grid Container

To make an HTML element behave as a grid container, you have to set the display property to grid or inline-grid.

Grid containers consist of grid items, placed inside columns and rows.

# The grid-template-columns Property

The grid-template-columns property defines the number of columns in your grid layout, and it can define the width of each column.

The value is a space-separated-list, where each value defines the width of the respective column.

If you want your grid layout to contain 4 columns, specify the width of the 4 columns, or "auto" if all columns should have the same width.

```
<!DOCTYPE html>
<html>
<head>
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
```

```
}
</style>
</head>
<body>

<h1>The grid-template-columns Property</h1>

<p>You can use the <em>grid-template-columns</em> property to specify the number of columns in your grid layout.</p>

<div class="grid-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
</div>

</body>
</html>
```

SOME MORE….

```
<!DOCTYPE html>
<html>
<head>
<style>
.grid-container {
  display: grid;
  justify-content: space-between;
  grid-template-columns: 50px 50px 50px; /*Make the grid smaller than the container*/
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
</style>
```

```
</head>
<body>

<h1>The justify-content Property</h1>

<p>Use the <em>justify-content</em> property to align the grid inside the container.</p>

<p>The value "space-between" will give the columns equal amount of space between them:</p>

<div class="grid-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
</div>

</body>
</html>
```

# Child Elements (Items)

A grid *container* contains grid *items*.

By default, a container has one grid item for each column, in each row, but you can style the grid items so that they will span multiple columns and/or rows.

# The grid-column Property:

The grid-column property defines on which column(s) to place an item.

You define where the item will start, and where the item will end.

```
<!DOCTYPE html>
<html>
<head>
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto auto auto;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
```

```
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}

.item2 {
  grid-column: 2 / span 3;
}
</style>
</head>
<body>

<h1>The grid-column Property</h1>

<p>Use the <em>grid-column</em> property to specify where to place an item.</p>

<p>Item2 will start on column line 2 and span 3 columns:</p>

<div class="grid-container">
  <div class="item1">1</div>
  <div class="item2">2</div>
  <div class="item3">3</div>
  <div class="item4">4</div>
  <div class="item5">5</div>
  <div class="item6">6</div>
  <div class="item7">7</div>
  <div class="item8">8</div>
  <div class="item9">9</div>
  <div class="item10">10</div>
  <div class="item11">11</div>
  <div class="item12">12</div>
  <div class="item13">13</div>
  <div class="item14">14</div>
  <div class="item15">15</div>
  <div class="item16">16</div>
</div>

</body>
</html>
```

## Get Going with Grid

The Grid module adds 18 new CSS properties in order to create a layout with rows and columns. Elements within the grid can be placed in any row/column, span multiple rows and/or columns, overlap other elements, and be aligned horizontally and/or vertically. There are similarities to Flexbox, but:

- Flexbox is one-dimensional. Elements come one after the other and may or may not wrap to a new “row”. Menus and photo galleries are a typical use case.
- Grid is two-dimensional and respects both rows and columns. If an element is too big for its cell, the row and/or column will grow accordingly. Grid is ideal for page and form layout.

It’s possibly better to compare CSS Grid with table-based layouts, but they’re considerably more flexible and require less markup. It has a steeper learning curve than other CSS concepts, but you’re unlikely to require all the properties and the minimum is demonstrated here. The most basic grid is defined on a containing element:

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

More practically, layouts also require the number of columns, their sizes, and the gap between rows and columns. For example:

```
.container {
  display: grid;
  grid-template-columns: 10% 1fr 2fr 12em;
  grid-gap: 0.3em 0.6em;
}
```

This defines four columns. Any measurement unit can be used as well as the fr fractional unit. This calculates the remaining space in a grid and distributes accordingly. The example above defines total of 3fr on columns two and three. If 600 pixels of horizontal space was available:

- `1fr` equates to `(1fr / 3fr) * 600px` = `200px`
- `2fr` equates to `(2fr / 3fr) * 600px` = `400px`

A gap of 0.3em is defined between rows and 0.6em between columns.

All child elements of the .container are now grid items. By default, the first child element will appear at row 1, column 1. The second in row 1, column 2, and the sixth in row 2, column 2. It’s possible to size rows using a property such as grid-template-rows, but heights will be inferred by the content.

Grid support is excellent. It’s not available in Opera Mini, but even IE11 offers an older implementation of the specification. In most cases, fallbacks are simple:

- Older browsers can use flexbox, floats, inline-blocks, or `display:table` layouts. All Grid properties are ignored.
- When a browser supports grid, all flexbox, floats, inline-blocks and table layout properties assigned to a grid item are disabled.