

[There's a newer version of Bootstrap 4!](#)



# Grid system

Use our powerful mobile-first flexbox grid to build layouts of all shapes and sizes thanks to a twelve column system, five default responsive tiers, Sass variables and mixins, and dozens of predefined classes.



Code in Node.js, Java, Python, and other open-source languages. ads via Carbon

## How it works

Bootstrap's grid system uses a series of containers, rows, and columns to layout and align content. It's built with [flexbox](#) and is fully responsive. Below is an example and an in-depth look at how the grid comes together.

New to or unfamiliar with flexbox? [Read this CSS Tricks flexbox guide](#) for background, terminology, guidelines, and code snippets.

One of three columns

One of three columns

One of three columns

```
<div class="container">
  <div class="row">
    <div class="col-sm">
      One of three columns
    </div>
    <div class="col-sm">
      One of three columns
    </div>
    <div class="col-sm">
      One of three columns
    </div>
  </div>
</div>
```

The above example creates three equal-width columns on small, medium, large, and extra large devices using our predefined grid classes. Those columns are centered in the page with the parent `.container`.

Breaking it down, here's how it works:

- Containers provide a means to center and horizontally pad your site's contents. Use `.container` for a responsive pixel width or `.container-fluid` for `width: 100%` across all viewport and device sizes.
- Rows are wrappers for columns. Each column has horizontal `padding` (called a gutter) for controlling the space between them. This `padding` is then counteracted on the rows with negative margins. This way, all the content in your columns is visually aligned down the left side.
- In a grid layout, content must be placed within columns and only columns may be immediate children of rows.
- Thanks to flexbox, grid columns without a specified `width` will automatically layout as equal width columns. For example, four instances

of `.col-sm` will each automatically be 25% wide from the small breakpoint and up. See the [auto-layout columns](#) section for more examples.

- Column classes indicate the number of columns you'd like to use out of the possible 12 per row. So, if you want three equal-width columns across, you can use `.col-4`.
- Column **widths** are set in percentages, so they're always fluid and sized relative to their parent element.
- Columns have horizontal **padding** to create the gutters between individual columns, however, you can remove the **margin** from rows and **padding** from columns with `.no-gutters` on the `.row`.
- To make the grid responsive, there are five grid breakpoints, one for each [responsive breakpoint](#): all breakpoints (extra small), small, medium, large, and extra large.
- Grid breakpoints are based on minimum width media queries, meaning **they apply to that one breakpoint and all those above it** (e.g., `.col-sm-4` applies to small, medium, large, and extra large devices, but not the first `xs` breakpoint).
- You can use predefined grid classes (like `.col-4`) or [Sass mixins](#) for more semantic markup.

Be aware of the limitations and [bugs around flexbox](#), like the [inability to use some HTML elements as flex containers](#).

## Grid options

While Bootstrap uses `ems` or `rems` for defining most sizes, `pxs` are used for grid breakpoints and container widths. This is because the viewport width is in pixels and does not change with the [font size](#).

See how aspects of the Bootstrap grid system work across multiple devices with a handy table.

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	<code>.col-</code>	<code>.col-sm-</code>	<code>.col-md-</code>	<code>.col-lg-</code>	<code>.col-xl-</code>
# of columns	12				
Gutter width	30px (15px on each side of a column)				
Nestable	Yes				
Column ordering	Yes				

## Auto-layout columns

Utilize breakpoint-specific column classes for easy column sizing without an explicit numbered class like `.col-sm-6`.

## Equal-width

For example, here are two grid layouts that apply to every device and viewport, from `xs` to `xl`. Add any number of unit-less classes for each breakpoint you need and every column will be the same width.

1 of 2		2 of 2			
1 of 3		2 of 3		3 of 3	

```
<div class="container">
  <div class="row">
    <div class="col">
      1 of 2
    </div>
    <div class="col">
      2 of 2
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col">
      2 of 3
    </div>
    <div class="col">
      3 of 3
    </div>
  </div>
</div>
```

Equal-width columns can be broken into multiple lines, but there was a [Safari flexbox bug](#) that prevented this from working without an explicit `flex-basis` or `border`. There are workarounds for older browser versions, but they shouldn't be necessary if you're up-to-date.

Column	Column
Column	Column

```
<div class="container">
  <div class="row">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="w-100"></div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

## Setting one column width

Auto-layout for flexbox grid columns also means you can set the width of one column and have the sibling columns automatically resize around it. You may use predefined grid classes (as shown below), grid mixins, or inline widths. Note that the other columns will resize no matter the width of the center column.

1 of 3	2 of 3 (wider)	3 of 3
1 of 3	2 of 3 (wider)	3 of 3

```
<div class="container">
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col-6">
      2 of 3 (wider)
    </div>
    <div class="col">
      3 of 3
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col-5">
      2 of 3 (wider)
    </div>
    <div class="col">
      3 of 3
    </div>
  </div>
</div>
```

## Variable width content

Use `col-{breakpoint}-auto` classes to size columns based on the natural width of their content.

1 of 3

Variable width content

3 of 3

1 of 3

Variable width content

3 of 3

```
<div class="container">
  <div class="row justify-content-md-center">
    <div class="col col-lg-2">
      1 of 3
    </div>
    <div class="col-md-auto">
      Variable width content
    </div>
    <div class="col col-lg-2">
      3 of 3
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col-md-auto">
      Variable width content
    </div>
    <div class="col col-lg-2">
      3 of 3
    </div>
  </div>
</div>
```

## Equal-width multi-row

Create equal-width columns that span multiple rows by inserting a `.w-100` where you want the columns to break to a new line. Make the breaks responsive by mixing the `.w-100` with some [responsive display utilities](#).

col	col
col	col

```
<div class="row">
  <div class="col">col</div>
  <div class="col">col</div>
  <div class="w-100"></div>
  <div class="col">col</div>
  <div class="col">col</div>
</div>
```

## Responsive classes

Bootstrap's grid includes five tiers of predefined classes for building complex responsive layouts. Customize the size of your columns on extra small, small, medium, large, or extra large devices however you see fit.

## All breakpoints

For grids that are the same from the smallest of devices to the largest, use the `.col` and `.col-*` classes. Specify a numbered class when you need a particularly sized column; otherwise, feel free to stick to `.col`.

col	col	col	col
-----	-----	-----	-----

col-8	col-4
-------	-------

```
<div class="row">
  <div class="col">col</div>
  <div class="col">col</div>
  <div class="col">col</div>
  <div class="col">col</div>
</div>
<div class="row">
  <div class="col-8">col-8</div>
  <div class="col-4">col-4</div>
</div>
```

## Stacked to horizontal

Using a single set of `.col-sm-*` classes, you can create a basic grid system that starts out stacked before becoming horizontal with at the small breakpoint (`sm`).

col-sm-8		col-sm-4
col-sm	col-sm	col-sm

```
<div class="row">
  <div class="col-sm-8">col-sm-8</div>
  <div class="col-sm-4">col-sm-4</div>
</div>
<div class="row">
  <div class="col-sm">col-sm</div>
  <div class="col-sm">col-sm</div>
  <div class="col-sm">col-sm</div>
</div>
```

## Mix and match

Don't want your columns to simply stack in some grid tiers? Use a combination of different classes for each tier as needed. See the example below for a better idea of how it all works.

.col-12 .col-md-8	
.col-6 .col-md-4	
.col-6 .col-md-4	.col-6 .col-md-4
.col-6 .col-md-4	
.col-6	.col-6

```
<!-- Stack the columns on mobile by making one full-width and the other half-width -->
<div class="row">
  <div class="col-12 col-md-8">.col-12 .col-md-8</div>
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
</div>

<!-- Columns start at 50% wide on mobile and bump up to 33.3% wide on desktop -->
<div class="row">
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
</div>

<!-- Columns are always 50% wide, on mobile and desktop -->
<div class="row">
  <div class="col-6">.col-6</div>
  <div class="col-6">.col-6</div>
</div>
```

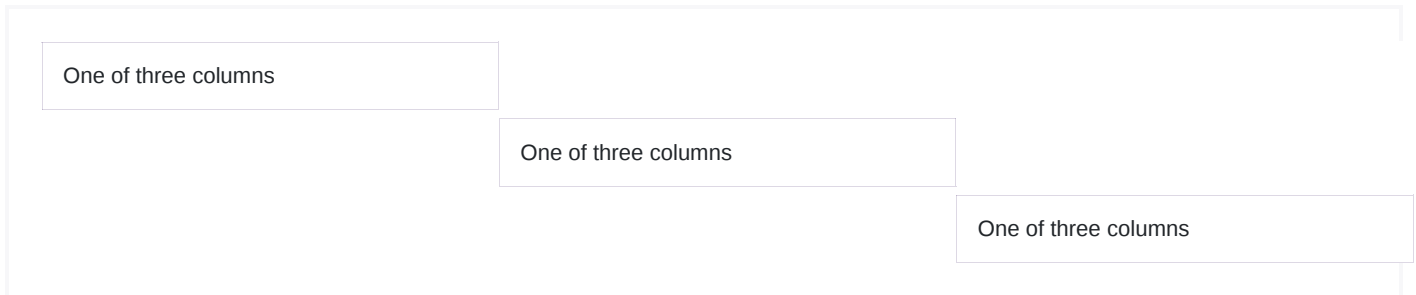
## Alignment

Use flexbox alignment utilities to vertically and horizontally align columns.

### Vertical alignment

One of three columns	One of three columns	One of three columns
One of three columns	One of three columns	One of three columns
One of three columns	One of three columns	One of three columns

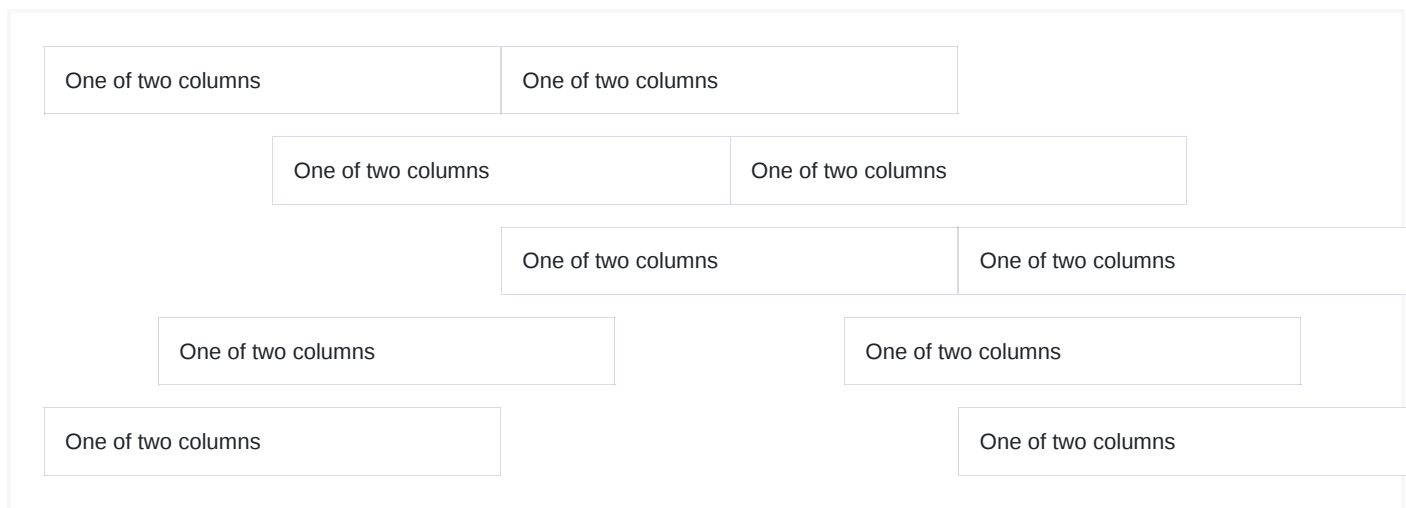
```
<div class="container">
  <div class="row align-items-start">
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
  </div>
  <div class="row align-items-center">
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
  </div>
  <div class="row align-items-end">
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
  </div>
</div>
```



```
<div class="container">
  <div class="row">
    <div class="col align-self-start">
      One of three columns
    </div>
    <div class="col align-self-center">
      One of three columns
    </div>
    <div class="col align-self-end">
      One of three columns
    </div>
  </div>
</div>
```

## Horizontal alignment





```
<div class="container">
  <div class="row justify-content-start">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
  <div class="row justify-content-center">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
  <div class="row justify-content-end">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
  <div class="row justify-content-around">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
  <div class="row justify-content-between">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
</div>
```

No gutters

The gutters between columns in our predefined grid classes can be removed with `.no-gutters`. This removes the negative `margins` from `.row` and the horizontal `padding` from all immediate children columns.

Here's the source code for creating these styles. Note that column overrides are scoped to only the first children columns and are targeted via [attribute selector](#). While this generates a more specific selector, column padding can still be further customized with [spacing utilities](#).

**Need an edge-to-edge design?** Drop the parent `.container` or `.container-fluid`.

```
.no-gutters {  
  margin-right: 0;  
  margin-left: 0;  
  
  > .col,  
  > [class*="col-"] {  
    padding-right: 0;  
    padding-left: 0;  
  }  
}
```

In practice, here's how it looks. Note you can continue to use this with all other predefined grid classes (including column widths, responsive tiers, reorders, and more).

.col-12 .col-sm-6 .col-md-8

.col-6 .col-md-4

```
<div class="row no-gutters">  
  <div class="col-12 col-sm-6 col-md-8">.col-12 .col-sm-6 .col-md-8</div>  
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>  
</div>
```

## Column wrapping

If more than 12 columns are placed within a single row, each group of extra columns will, as one unit, wrap onto a new line.

.col-9

.col-4

Since  $9 + 4 = 13 > 12$ , this 4-column-wide div gets wrapped onto a new line as one contiguous unit.

.col-6

Subsequent columns continue along the new line.

```
<div class="row">  
  <div class="col-9">.col-9</div>  
  <div class="col-4">.col-4<br>Since 9 + 4 = 13 > 12, this 4-column-wide div gets wrapped onto a new line as one contiguous unit.</div>  
  <div class="col-6">.col-6<br>Subsequent columns continue along the new line.</div>  
</div>
```

## Column breaks

Breaking columns to a new line in flexbox requires a small hack: add an element with `width: 100%` wherever you want to wrap your columns to a new line. Normally this is accomplished with multiple `.rows`, but not every implementation method can account for this.

.col-6 .col-sm-3	.col-6 .col-sm-3
.col-6 .col-sm-3	.col-6 .col-sm-3

```
<div class="row">
  <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
  <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>

  <!-- Force next columns to break to new line -->
  <div class="w-100"></div>

  <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
  <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
</div>
```

You may also apply this break at specific breakpoints with our [responsive display utilities](#).

.col-6 .col-sm-4	.col-6 .col-sm-4	.col-6 .col-sm-4
.col-6 .col-sm-4		

```
<div class="row">
  <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>
  <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>

  <!-- Force next columns to break to new line at md breakpoint and up -->
  <div class="w-100 d-none d-md-block"></div>

  <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>
  <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>
</div>
```

## Reordering

### Order classes

Use `.order-` classes for controlling the **visual order** of your content. These classes are responsive, so you can set the `order` by breakpoint (e.g., `.order-1.order-md-2`). Includes support for **1** through **12** across all five grid tiers.

First, but unordered	Third, but first	Second, but last
----------------------	------------------	------------------

```
<div class="container">
  <div class="row">
    <div class="col">
      First, but unordered
    </div>
    <div class="col order-12">
      Second, but last
    </div>
    <div class="col order-1">
      Third, but first
    </div>
  </div>
</div>
```

There are also responsive `.order-first` and `.order-last` classes that change the `order` of an element by applying `order: -1` and `order: 13` (`order: $columns + 1`), respectively. These classes can also be intermixed with the numbered `.order-*` classes as needed.

Third, but first	Second, but unordered	First, but last
------------------	-----------------------	-----------------

```
<div class="container">
  <div class="row">
    <div class="col order-last">
      First, but last
    </div>
    <div class="col">
      Second, but unordered
    </div>
    <div class="col order-first">
      Third, but first
    </div>
  </div>
</div>
```

## Offsetting columns

You can offset grid columns in two ways: our responsive `.offset-` grid classes and our [margin utilities](#). Grid classes are sized to match columns while margins are more useful for quick layouts where the width of the offset is variable.

### Offset classes

Move columns to the right using `.offset-md-*` classes. These classes increase the left margin of a column by `*` columns. For example, `.offset-md-4` moves `.col-md-4` over four columns.

.col-md-4
.col-md-4 .offset-md-4
.col-md-3 .offset-md-3
.col-md-3 .offset-md-3
.col-md-6 .offset-md-3

```
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4 offset-md-4">.col-md-4 .offset-md-4</div>
</div>
<div class="row">
  <div class="col-md-3 offset-md-3">.col-md-3 .offset-md-3</div>
  <div class="col-md-3 offset-md-3">.col-md-3 .offset-md-3</div>
</div>
<div class="row">
  <div class="col-md-6 offset-md-3">.col-md-6 .offset-md-3</div>
</div>
```

In addition to column clearing at responsive breakpoints, you may need to reset offsets. See this in action in [the grid example](#).

.col-sm-5 .col-md-6

.col-sm-5 .offset-sm-2 .col-md-6 .offset-md-0

.col-sm-6 .col-md-5 .col-lg-6

.col-sm-6 .col-md-5 .offset-md-2 .col-lg-6 .offset-lg-0

```
<div class="row">
  <div class="col-sm-5 col-md-6">.col-sm-5 .col-md-6</div>
  <div class="col-sm-5 offset-sm-2 col-md-6 offset-md-0">.col-sm-5 .offset-sm-2 .col-md-6 .offset-md-0</div>
</div>

<div class="row">
  <div class="col-sm-6 col-md-5 col-lg-6">.col-sm-6 .col-md-5 .col-lg-6</div>
  <div class="col-sm-6 col-md-5 offset-md-2 col-lg-6 offset-lg-0">.col-sm-6 .col-md-5 .offset-md-2 .col-lg-6
  .offset-lg-0</div>
</div>
```

## Margin utilities

With the move to flexbox in v4, you can use margin utilities like `.mr-auto` to force sibling columns away from one another.

.col-md-4

.col-md-4 .ml-auto

.col-md-3 .ml-md-auto

.col-md-3 .ml-md-auto

.col-auto .mr-auto

.col-auto

```
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4 ml-auto">.col-md-4 .ml-auto</div>
</div>
<div class="row">
  <div class="col-md-3 ml-md-auto">.col-md-3 .ml-md-auto</div>
  <div class="col-md-3 ml-md-auto">.col-md-3 .ml-md-auto</div>
</div>
<div class="row">
  <div class="col-auto mr-auto">.col-auto .mr-auto</div>
  <div class="col-auto">.col-auto</div>
</div>
```

## Nesting

To nest your content with the default grid, add a new `.row` and set of `.col-sm-*` columns within an existing `.col-sm-*` column. Nested rows should include a set of columns that add up to 12 or fewer (it is not required that you use all 12 available columns).

Level 1: .col-sm-9	
Level 2: .col-8 .col-sm-6	Level 2: .col-4 .col-sm-6

```
<div class="row">
  <div class="col-sm-9">
    Level 1: .col-sm-9
    <div class="row">
      <div class="col-8 col-sm-6">
        Level 2: .col-8 .col-sm-6
      </div>
      <div class="col-4 col-sm-6">
        Level 2: .col-4 .col-sm-6
      </div>
    </div>
  </div>
</div>
```

## Sass mixins

When using Bootstrap's source Sass files, you have the option of using Sass variables and mixins to create custom, semantic, and responsive page layouts. Our predefined grid classes use these same variables and mixins to provide a whole suite of ready-to-use classes for fast responsive layouts.

## Variables

Variables and maps determine the number of columns, the gutter width, and the media query point at which to begin floating columns. We use these to generate the predefined grid classes documented above, as well as for the custom mixins listed below.

```
$grid-columns: 12;
$grid-gutter-width: 30px;

$grid-breakpoints: (
  // Extra small screen / phone
  xs: 0,
  // Small screen / phone
  sm: 576px,
  // Medium screen / tablet
  md: 768px,
  // Large screen / desktop
  lg: 992px,
  // Extra large screen / wide desktop
  xl: 1200px
);

$container-max-widths: (
  sm: 540px,
  md: 720px,
  lg: 960px,
  xl: 1140px
);
```

## Mixins

Mixins are used in conjunction with the grid variables to generate semantic CSS for individual grid columns.

```
// Creates a wrapper for a series of columns
@include make-row();

// Make the element grid-ready (applying everything but the width)
@include make-col-ready();
@include make-col($size, $columns: $grid-columns);

// Get fancy by offsetting, or changing the sort order
@include make-col-offset($size, $columns: $grid-columns);
```

## Example usage

You can modify the variables to your own custom values, or just use the mixins with their default values. Here's an example of using the default settings to create a two-column layout with a gap between.

```

.example-container {
  width: 800px;
  @include make-container();
}

.example-row {
  @include make-row();
}

.example-content-main {
  @include make-col-ready();

  @include media-breakpoint-up(sm) {
    @include make-col(6);
  }
  @include media-breakpoint-up(lg) {
    @include make-col(8);
  }
}

.example-content-secondary {
  @include make-col-ready();

  @include media-breakpoint-up(sm) {
    @include make-col(6);
  }
  @include media-breakpoint-up(lg) {
    @include make-col(4);
  }
}

```

Main content

Secondary content

```

<div class="example-container">
  <div class="example-row">
    <div class="example-content-main">Main content</div>
    <div class="example-content-secondary">Secondary content</div>
  </div>
</div>

```

## Customizing the grid

Using our built-in grid Sass variables and maps, it's possible to completely customize the predefined grid classes. Change the number of tiers, the media query dimensions, and the container widths—then recompile.

### Columns and gutters

The number of grid columns can be modified via Sass variables. `$grid-columns` is used to generate the widths (in percent) of each individual column while `$grid-gutter-width` allows breakpoint-specific widths that are divided evenly across `padding-left` and `padding-right` for the column gutters.

```

$grid-columns: 12 !default;
$grid-gutter-width: 30px !default;

```

### Grid tiers



Moving beyond the columns themselves, you may also customize the number of grid tiers. If you wanted just four grid tiers, you'd update the `$grid-breakpoints` and `$container-max-widths` to something like this:

```
$grid-breakpoints: (  
  xs: 0,  
  sm: 480px,  
  md: 768px,  
  lg: 1024px  
);  
  
$container-max-widths: (  
  sm: 420px,  
  md: 720px,  
  lg: 960px  
);
```

When making any changes to the Sass variables or maps, you'll need to save your changes and recompile. Doing so will output a brand new set of predefined grid classes for column widths, offsets, and ordering. Responsive visibility utilities will also be updated to use the custom breakpoints. Make sure to set grid values in `px` (not `rem`, `em`, or `%`).