

[Ctrl + /](#)

v5.0 ▾

[View on GitHub](#)

Grid system

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



Your new development career awaits. Check out the latest listings.

ads via Carbon

On this page

[Example](#)

[How it works](#)

[Grid options](#)

[Auto-layout columns](#)

[Equal-width](#)

[Setting one column width](#)

[Variable width content](#)

[Responsive classes](#)

[All breakpoints](#)

[Stacked to horizontal](#)

[Mix and match](#)

[Row columns](#)

[Nesting](#)

[Sass](#)

[Variables](#)

[Mixins](#)

[Example usage](#)

[Customizing the grid](#)

Columns and gutters

Grid tiers

Example

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 explanation for how the grid system comes together.

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

Column

Column

Column

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

The above example creates three equal-width columns across all devices and viewports using our predefined grid classes. Those columns are centered in the page with the parent `.container`.

How it works

Breaking it down, here's how the grid system comes together:

- **Our grid supports [six responsive breakpoints](#).** Breakpoints are based on `min-width` media queries, meaning they affect that breakpoint and all those above it (e.g., `.col-sm-4` applies to

`sm`, `md`, `lg`, `xl`, and `xxl`). This means you can control container and column sizing and behavior by each breakpoint.

- **Containers center and horizontally pad your content.** Use `.container` for a responsive pixel width, `.container-fluid` for `width: 100%` across all viewports and devices, or a responsive container (e.g., `.container-md`) for a combination of fluid and pixel widths.
- **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 to ensure the content in your columns is visually aligned down the left side. Rows also support modifier classes to [uniformly apply column sizing](#) and [gutter classes](#) to change the spacing of your content.
- **Columns are incredibly flexible.** There are 12 template columns available per row, allowing you to create different combinations of elements that span any number of columns. Column classes indicate the number of template columns to span (e.g., `col-4` spans four). `widths` are set in percentages so you always have the same relative sizing.
- **Gutters are also responsive and customizable.** [Gutter classes are available](#) across all breakpoints, with all the same sizes as our [margin and padding spacing](#). Change horizontal gutters with `.gx-*` classes, vertical gutters with `.gy-*`, or all gutters with `.g-*` classes. `.g-0` is also available to remove gutters.
- **Sass variables, maps, and mixins power the grid.** If you don't want to use the predefined grid classes in Bootstrap, you can use our [grid's source Sass](#) to create your own with more semantic markup. We also include some CSS custom properties to consume these Sass variables for even greater flexibility for you.

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

Grid options

Bootstrap's grid system can adapt across all six default breakpoints, and any breakpoints you customize. The six default grid tiers are as follow:

- Extra small (xs)
- Small (sm)
- Medium (md)
- Large (lg)
- Extra large (xl)
- Extra extra large (xxl)

As noted above, each of these breakpoints have their own container, unique class prefix, and modifiers. Here's how the grid changes across these breakpoints:

	xs <576px	sm ≥576px	md ≥768px	lg ≥992px	xl ≥1200px	xxl ≥1400px
Container <i>max-width</i>	None (auto)	540px	720px	960px	1140px	1320px
Class prefix	<i>.col-</i>	<i>.col-sm-</i>	<i>.col-md-</i>	<i>.col-lg-</i>	<i>.col-xl-</i>	<i>.col-xxl-</i>
# of columns	12					
Gutter width	1.5rem (.75rem on left and right)					
Custom gutters	Yes					
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 *xxl*. 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>

```

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>
```

Responsive classes

Bootstrap's grid includes six 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`.



```
<div class="container">
  <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>
</div>
```

Stacked to horizontal

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

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

```
<div class="container">
  <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>
</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-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

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

```
...
```

Row columns

Use the responsive `.row-cols-*` classes to quickly set the number of columns that best render your content and layout. Whereas normal `.col-*` classes apply to the individual columns (e.g., `.col-md-4`), the row columns classes are set on the parent `.row` as a default for contained columns. With `.row-cols-auto` you can give the columns their natural width.

Use these row columns classes to quickly create basic grid layouts or to control your card layouts and override when needed at the column level.

Column	Column
Column	Column

```
<div class="container">
```

```
  <div class="row row-cols-2">
```

```
    <div class="col">Column</div>
```

```
    <div class="col">Column</div>
```

```
    <div class="col">Column</div>
```

```
    <div class="col">Column</div>
```

```
  </div>
```

```
</div>
```

Column	Column	Column
Column		

```
<div class="container">
  <div class="row row-cols-3">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

Column	Column	Column	Column
--------	--------	--------	--------

```
<div class="container">
  <div class="row row-cols-auto">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

Column	Column	Column	Column
--------	--------	--------	--------

```
<div class="container">
  <div class="row row-cols-4">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

Column	Column	Column
Column		

```
<div class="container">
  <div class="row row-cols-4">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col-6">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

Column	Column
Column	Column

```
<div class="container">
  <div class="row row-cols-1 row-cols-sm-2 row-cols-md-4">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

Column		Column	
Column		Column	
Column		Column	
Column	Column		Column

Column	Column	Column
--------	--------	--------

```
<div class="container">
  <div class="row row-cols-2 row-cols-lg-3">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col-4 col-lg-2">Column</div>
    <div class="col-4 col-lg-2">Column</div>
    <div class="col-4 col-lg-2">Column</div>
    <div class="col-4 col-lg-2">Column</div>
    <div class="col-4 col-lg-2">Column</div>
    <div class="col-4 col-lg-2">Column</div>
  </div>
</div>
```

You can also use the accompanying Sass mixin, `row-cols()`:

```
.element {
  // Three columns to start
  @include row-cols(3);

  // Five columns from medium breakpoint up
  @include media-breakpoint-up(md) {
    @include row-cols(5);
  }
}
```

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-3	Level 2: .col-8 .col-sm-6	Level 2: .col-4 .col-sm-6
--------------------	---------------------------	---------------------------

```
<div class="container">
  <div class="row">
    <div class="col-sm-3">
      Level 1: .col-sm-3
    </div>
    <div class="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>
</div>
```

Sass

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: 1.5rem;
```

```
$grid-breakpoints: (
  xs: 0,
  sm: 576px,
  md: 768px,
  lg: 992px,
  xl: 1200px,
```

```
xxl: 1400px
);
```

```
$container-max-widths: (
  sm: 540px,
  md: 720px,
  lg: 960px,
  xl: 1140px,
  xxl: 1320px
);
```

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();

// Without optional size values, the mixin will create equal columns (similar to use
@include make-col();
@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 {
  @include make-container();
  // Make sure to define this width after the mixin to override
  // `width: 100%` generated by `make-container()`
  width: 800px;
```

```

}

.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` sets the width for the column gutters.

```
$grid-columns: 12 !default;  
$grid-gutter-width: 1.5rem !default;
```

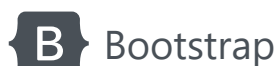
Grid tiers

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 `%`).



Designed and built with all the love in the world by the Bootstrap team with the help of our contributors.

Code licensed MIT, docs CC BY 3.0.

Currently v5.0.2.

Analytics by Fathom.

Links

[Home](#)

[Docs](#)

[Examples](#)

[Themes](#)

[Blog](#)

[Swag Store](#)

Projects

[Bootstrap 5](#)

[Bootstrap 4](#)

[Icons](#)

[RFS](#)

[npm starter](#)

Guides

[Getting started](#)

[Starter template](#)

[Webpack](#)

[Parcel](#)

Community

[Issues](#)

[Discussions](#)

[Corporate sponsors](#)

[Open Collective](#)

[Stack Overflow](#)