Bootstrap 5

Bootstrap 5 Notes under layout category (Breakpoints - Gutters).

What is Bootstrap?

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains HTML, CSS and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

Get started by including Bootstrap’s production-ready CSS and JavaScript via CDN without the need for any build steps.

1. **Create a new `index.html` file in your project root.** Include the `<meta name="viewport">` tag as well for proper responsive behavior in mobile devices.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
  </head>
  <body>
    <h1>Hello, world!</h1>
  </body>
</html>
```

1. **Include Bootstrap’s CSS and JS.** Place the `<link>` tag in the `<head>` for CSS, and the `<script>` tag for JavaScript bundle (including Popper for positioning dropdowns, poppers, and tooltips) before the closing `</body>`.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel
="stylesheet" integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ" cr
ossorigin="anonymous">
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.j
s" integrity="sha384-ENjdO4Dr2bkBIFxQpeoTz1HIcje39Wm4jDKdf19U8gI4ddQ3GYNS7NTKfAdVQSZe" crossorigin
="anonymous"></script>
  </body>
</html>
```

You can also include Popper and JS separately. If you don't plan to use dropdowns, popovers, or tooltips, save some kilobytes by not including Popper.

```
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.7/dist/umd/popper.min.js" integrity="s
ha384-zYPOMqeu1DAVkHiLqWBUTcbYfZ8osu1Nd6Z89ify25QV9guujx43ITvfi12/QExE" crossorigin="anonymous"></sc
ript>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.min.js" integrity
="sha384-Y4oOpwW3duJdCWv5ly8SCFYWqFDsfob/3GkgExXKV4idmbt98QcxXYs9UoXAB7BZ" crossorigin="anonymous">
</script>
```

2. **Hello, world!** Open the page in your browser of choice to see your Bootstrapped page. Now you can start building with Bootstrap by creating your own layout, adding dozens of components, and utilizing examples.

CDN links

As reference, here are primary CDN links.

Description	URL
CSS	`https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css`
JS	`https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js`

1. Breakpoints

Breakpoints are customizable widths that determine how your responsive layout behaves across device or viewport sizes in Bootstrap.

Core concepts

- **Breakpoints are the building blocks of responsive design.** Use them to control when your layout can be adapted at a particular viewport or device size.
- **Use media queries to architect your CSS by breakpoint.** Media queries are a feature of CSS that allow you to conditionally apply styles based on a set of browser and operating system parameters. We most commonly use `min-width` in our media queries.

- **Mobile first, responsive design is the goal.** Bootstrap's CSS aims to apply the bare minimum of styles to make a layout work at the smallest breakpoint, and then layers on styles to adjust that design for larger devices. This optimizes your CSS, improves rendering time, and provides a great experience for your visitors.

Available breakpoints

Bootstrap includes six default breakpoints, sometimes referred to as *grid tiers*, for building responsively. These breakpoints can be customized if you're using source Sass files.

Breakpoint	Class infix	Dimensions
Extra small	*None*	<576px
Small	`sm`	≥576px
Medium	`md`	≥768px
Large	`lg`	≥992px
Extra large	`xl`	≥1200px
Extra extra large	`xxl`	≥1400px

Each breakpoint was chosen to comfortably hold containers whose widths are multiples of 12. Breakpoints are also representative of a subset of common device sizes and viewport dimensions—they don't specifically target every use case or device. Instead, the ranges provide a strong and consistent foundation to build on for nearly any device.

These breakpoints are customizable via Sass—you'll find them in a Sass map in our `_variables.scss` stylesheet.

```
$grid-breakpoints: (
xs: 0,
sm: 576px,
md: 768px,
lg: 992px,
xl: 1200px,
xxl: 1400px
);
```

Media queries

Since Bootstrap is developed to be mobile first, we use a handful of media queries to create sensible breakpoints for our layouts and interfaces. These breakpoints are mostly based on minimum viewport widths and allow us to scale up elements as the viewport changes.

Min-width

Bootstrap primarily uses the following media query ranges—or breakpoints—in our source Sass files for our layout, grid system, and components.

// Source mixins

```
// No media query necessary for xs breakpoint as it's effectively @media (min-width: 0) { ... }
@include media-breakpoint-up(sm) { ... }
@include media-breakpoint-up(md) { ... }
@include media-breakpoint-up(lg) { ... }
@include media-breakpoint-up(xl) { ... }
@include media-breakpoint-up(xxl) { ... }
```

// Usage

```
// Example: Hide starting at min-width: 0, and then show at the sm breakpoint
.custom-class {
display: none;
}
@include media-breakpoint-up(sm) {
.custom-class {
display: block;
}
}
```

These Sass mixins translate in our compiled CSS using the values declared in our Sass variables. For example:

// X-Small devices (portrait phones, less than 576px)
// No media query for `xs` since this is the default in Bootstrap

```
// Small devices (landscape phones, 576px and up)
@media (min-width: 576px) { ... }
```

```
// Medium devices (tablets, 768px and up)
@media (min-width: 768px) { ... }
```

```
// Large devices (desktops, 992px and up)
@media (min-width: 992px) { ... }
```

```
// X-Large devices (large desktops, 1200px and up)
@media (min-width: 1200px) { ... }
```

```
// XX-Large devices (larger desktops, 1400px and up)
@media (min-width: 1400px) { ... }
```

Max-width

We occasionally use media queries that go in the other direction (the given screen size *or smaller*):

```
// No media query necessary for xs breakpoint as it's effectively @media (max-width: 0) { ... }
@include media-breakpoint-down(sm) { ... }
@include media-breakpoint-down(md) { ... }
@include media-breakpoint-down(lg) { ... }
@include media-breakpoint-down(xl) { ... }
@include media-breakpoint-down(xxl) { ... }
```

```
// Example: Style from medium breakpoint and down
@include media-breakpoint-down(md) {
.custom-class {
display: block;
}
}
```

These mixins take those declared breakpoints, subtract `.02px` from them, and use them as our `max-width` values. For example:

```
// xs returns only a ruleset and no media query
// ... { ... }
```

```
// sm applies to x-small devices (portrait phones, less than 576px)
@media (max-width: 575.98px) { ... }
```

```
// md applies to small devices (landscape phones, less than 768px)
@media (max-width: 767.98px) { ... }
```

```
// lg applies to medium devices (tablets, less than 992px)
@media (max-width: 991.98px) { ... }
```

```
// xl applies to large devices (desktops, less than 1200px)
@media (max-width: 1199.98px) { ... }
```

```
// xxl applies to x-large devices (large desktops, less than 1400px)
@media (max-width: 1399.98px) { ... }
```

Single breakpoint

There are also media queries and mixins for targeting a single segment of screen sizes using the minimum and maximum breakpoint widths.

```
@include media-breakpoint-only(xs) { ... }
@include media-breakpoint-only(sm) { ... }
```

```
@include media-breakpoint-only(md) { ... }
@include media-breakpoint-only(lg) { ... }
@include media-breakpoint-only(xl) { ... }
@include media-breakpoint-only(xxl) { ... }
```

For example the `@include media-breakpoint-only(md) { ... }` will result in :

```
@media (min-width: 768px) and (max-width: 991.98px) { ... }
```

Between breakpoints

Similarly, media queries may span multiple breakpoint widths:

```
@include media-breakpoint-between(md, xl) { ... }
```

```
// Example
// Apply styles starting from medium devices and up to extra large devices
@media (min-width: 768px) and (max-width: 1199.98px) { ... }
```

2. Containers

Containers are a fundamental building block of Bootstrap that contain, pad, and align your content within a given device or viewport.

How they work

Containers are the most basic layout element in Bootstrap and are **required when using our default grid system**. Containers are used to contain, pad, and (sometimes) center the content within them. While containers *can* be nested, most layouts do not require a nested container.

Bootstrap comes with three different containers:

- `.container`, which sets a `max-width` at each responsive breakpoint
- `.container-{breakpoint}`, which is `width: 100%` until the specified breakpoint
- `.container-fluid`, which is `width: 100%` at all breakpoints

The table below illustrates how each container's `max-width` compares to the original `.container` and `.container-fluid` across each breakpoint.

Extra small<576px	Small≥576px	Medium≥768px	Large≥992px	X-Large≥1200px	XX-Large≥1400px
`.container`	100%	540px	720px	960px	1140px
`.container-sm`	100%	540px	720px	960px	1140px
`.container-md`					

	100%	100%	720px	960px	1140px
`.container-lg`	100%	100%	100%	960px	1140px
`.container-xl`	100%	100%	100%	100%	1140px
`.container-xxl`	100%	100%	100%	100%	100%
`.container-fluid`	100%	100%	100%	100%	100%

Default container

Default `.container` class is a responsive, fixed-width container, meaning its `max-width` changes at each breakpoint.

```
<div class="container">
  <!-- Content here -->
</div>
```

Responsive containers

Responsive containers allow you to specify a class that is 100% wide until the specified breakpoint is reached, after which we apply `max-width`s for each of the higher breakpoints. For example, `.container-sm` is 100% wide to start until the `sm` breakpoint is reached, where it will scale up with `md`, `lg`, `xl`, and `xxl`.

```
<div class="container-sm">100% wide until small breakpoint</div>
<div class="container-md">100% wide until medium breakpoint</div>
<div class="container-lg">100% wide until large breakpoint</div>
<div class="container-xl">100% wide until extra large breakpoint</div>
<div class="container-xxl">100% wide until extra extra large breakpoint</div>
```

Fluid containers

Use `.container-fluid` for a full width container, spanning the entire width of the viewport.

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

Sass

As shown above, Bootstrap generates a series of predefined container classes to help you build the layouts you desire. You may customize these predefined container classes by modifying the Sass map (found in `_variables.scss`) that powers them:

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

In addition to customizing the Sass, you can also create your own containers with our Sass mixin.

```
// Source mixin
@mixin make-container($padding-x: $container-padding-x) {
width: 100%;
padding-right: $padding-x;
padding-left: $padding-x;
margin-right: auto;
margin-left: auto;
}
```

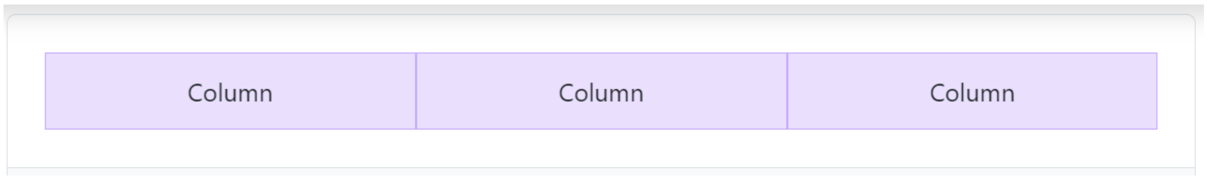
```
// Usage
.custom-container {
@include make-container();
}
```

3. 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.

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.



```
<div class="container text-center">
<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). `width`s 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 follows:

- 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:

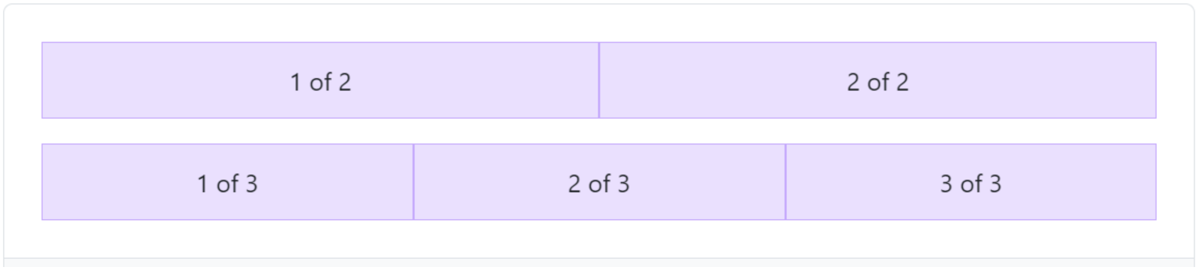
xs<576px	sm≥576px	md≥768px	lg≥992px	xl≥1200px	xxl≥1400px
Container `max-width`	None (auto)	540px	720px	960px	1140px
Class prefix	`.col-`	`.col-sm-`	`.col-md-`	`.col-lg-`	`.col-xl-`
# of columns	12				
Gutter width	1.5rem (.75rem on left and right)				
Custom gutters	Yes				
Nestable	Yes				
Column ordering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

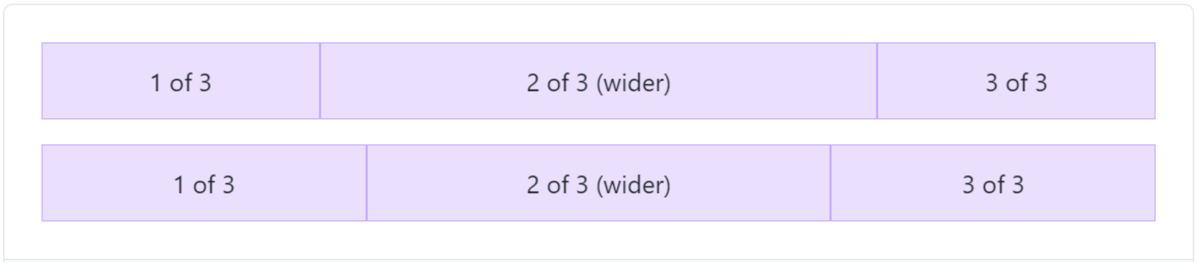


HTML

```
<div class="container text-center">
  <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.

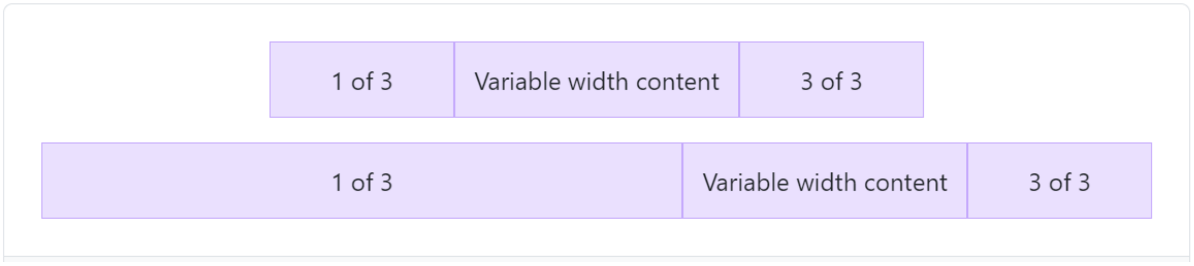


HTML

```
<div class="container text-center">
  <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.



HTML

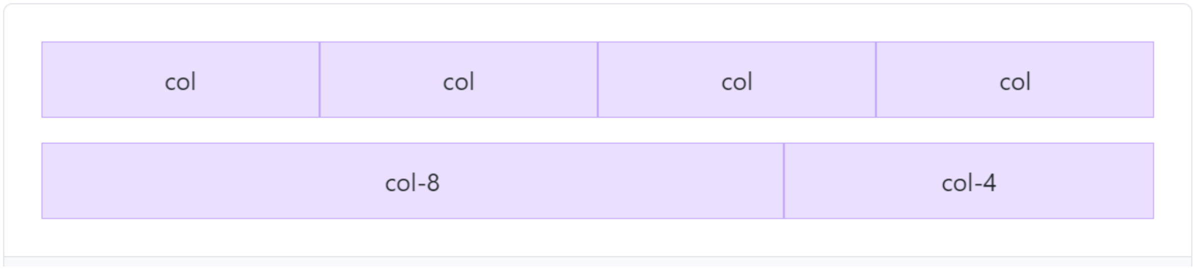
```
<div class="container text-center">
  <div class="row justify-content-md-center">
    <div class="col col-lg-2">
      1 of 3
    </div>
    <div class="col-md-auto">
      Variable width content
    </div>
    <div class="col col-lg-2">
      3 of 3
    </div>
  </div>
  <div class="row">
    <div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

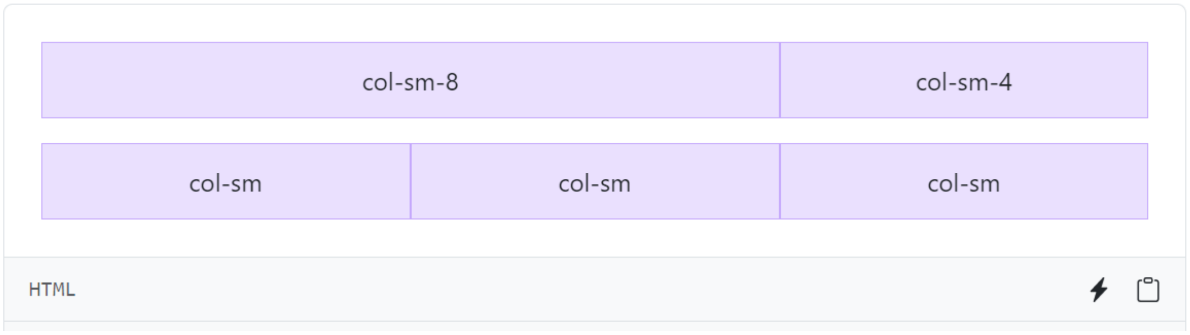


HTML

```
<div class="container text-center">
  <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`).

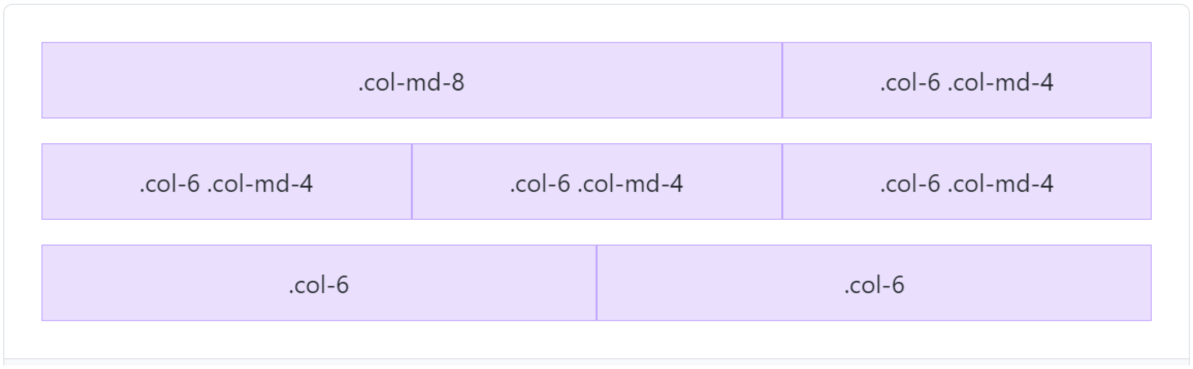


HTML

```
<div class="container text-center">
  <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.



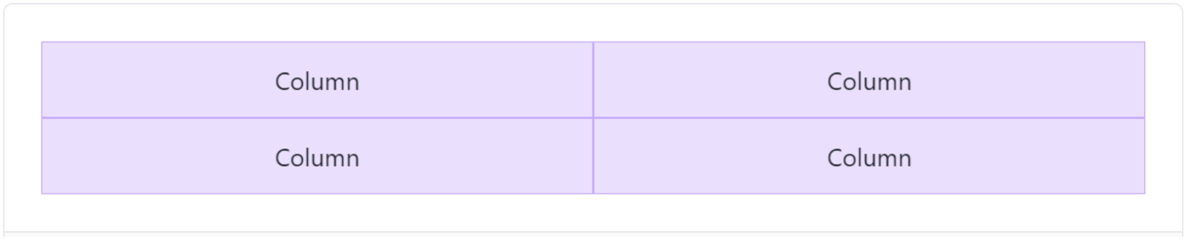
HTML

```
<div class="container text-center">
  <!-- 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>

  <!-- 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>
</div>
```

Row columns

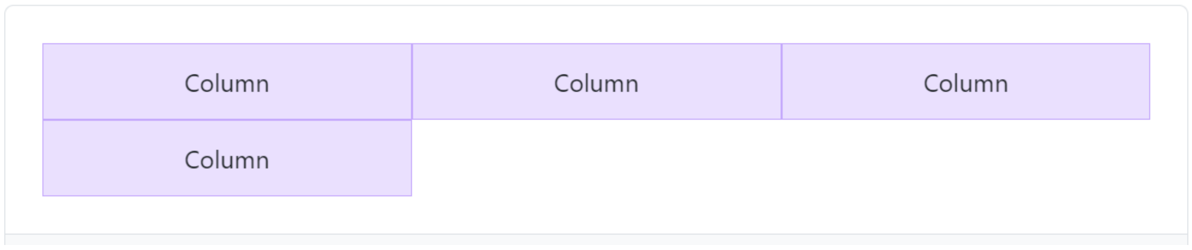
Use the responsive `.row-cols-*` classes to quickly set the number of columns that best render your content and layout. Whereas normal `.col-*` classes apply to the individual columns (e.g., `.col-md-4`), the row columns classes are set on the parent `.row` as a shortcut. With `.row-cols-auto` you can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

Use these row columns classes to quickly create basic grid layouts or to control your card layouts.



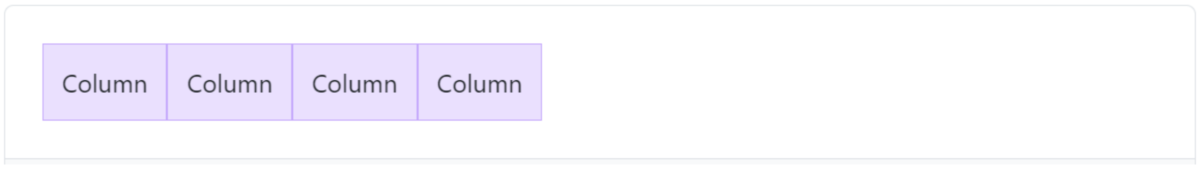
HTML

```
<div class="container text-center">
  <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>
```



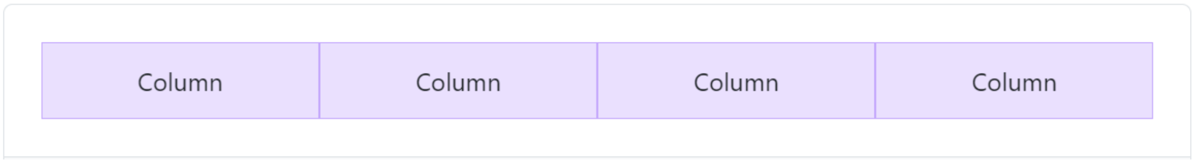
HTML

```
<div class="container text-center">
  <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>
```



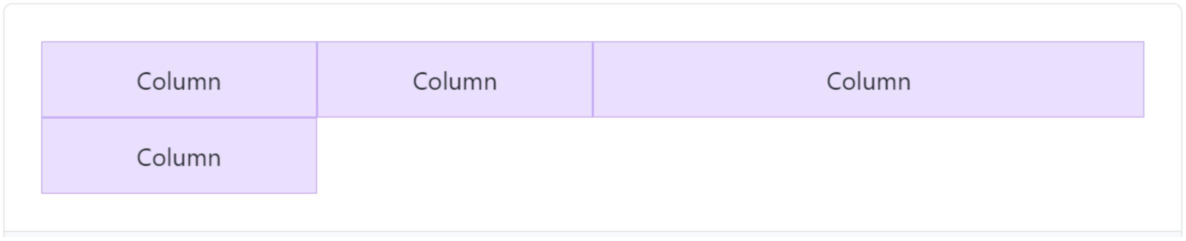
HTML

```
<div class="container text-center">
  <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>
```



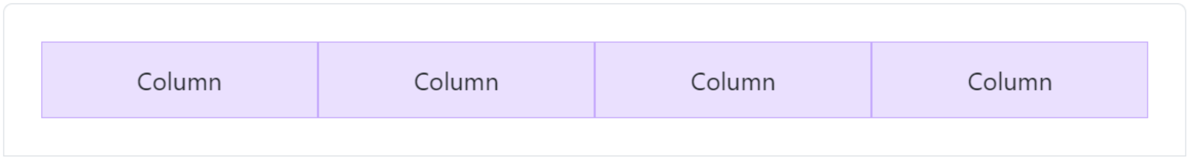
HTML

```
<div class="container text-center">
  <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>
```



HTML

```
<div class="container text-center">
  <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>
```



HTML

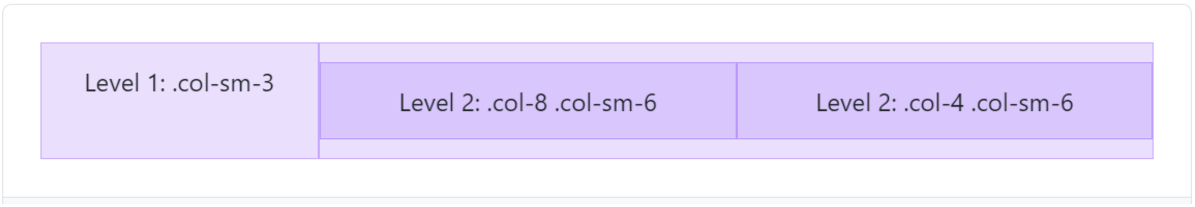
```
<div class="container text-center">
  <div class="row row-cols-1 row-cols-sm-2 row-cols-md-4">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

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).



HTML

```
<div class="container text-center">
  <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>
```

CSS

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.

Sass 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-row-columns:  6;
```

scss/_variables.scss

```
$grid-breakpoints: (
  xs: 0,
  sm: 576px,
  md: 768px,
  lg: 992px,
  xl: 1200px,
  xxl: 1400px
);
```

scss/_variables.scss

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

Sass 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 using .col)
@include make-col();
@include make-col($size, $columns: $grid-columns);

// Offset with margins
@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

HTML

```
<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-row-columns` is used to set the maximum number of columns of `.row-cols-*`, any number over this limit is ignored.

```
$grid-columns: 12 !default;
$grid-gutter-width: 1.5rem !default;
$grid-row-columns: 6 !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 `%`).

4. Columns

Learn how to modify columns with a handful of options for alignment, ordering, and offsetting thanks to our flexbox grid system. Plus, see how to use column classes to manage widths of non-grid elements.

How they work

- **Columns build on the grid's flexbox architecture.** Flexbox means we have options for changing individual columns and modifying groups of columns at the row level. You choose how columns grow, shrink, or otherwise change.
- **When building grid layouts, all content goes in columns.** The hierarchy of Bootstrap's grid goes from container to row to column to your content. On rare occasions, you may combine content and column, but be aware there can be unintended consequences.

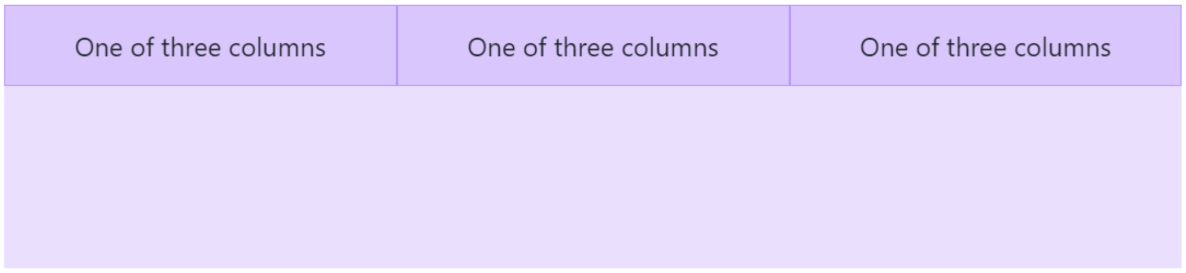
- **Bootstrap includes predefined classes for creating fast, responsive layouts.** With six breakpoints and a dozen columns at each grid tier, we have dozens of classes already built for you to create your desired layouts. This can be disabled via Sass if you wish.

Alignment

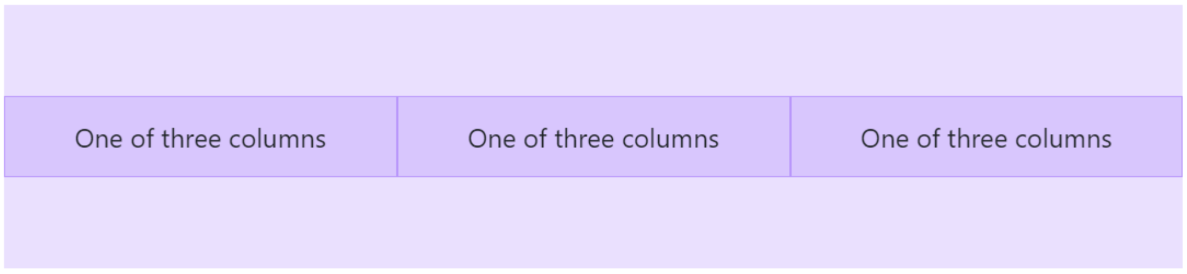
Use flexbox alignment utilities to vertically and horizontally align columns.

Vertical alignment

Change the vertical alignment with any of the responsive `align-items-*` classes.

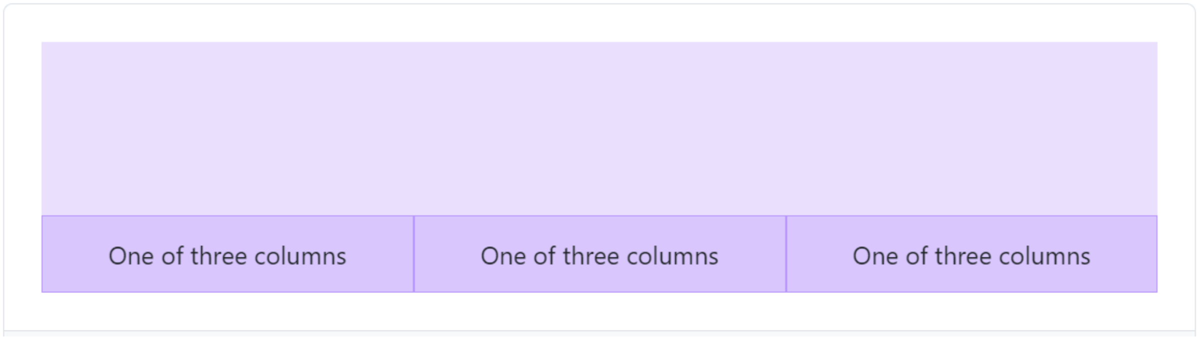


```
<div class="container text-center">
<div class="row align-items-start">
<div class="col">
One of three columns
</div>
<div class="col">
One of three columns
</div>
<div class="col">
One of three columns
</div>
</div>
</div>
```



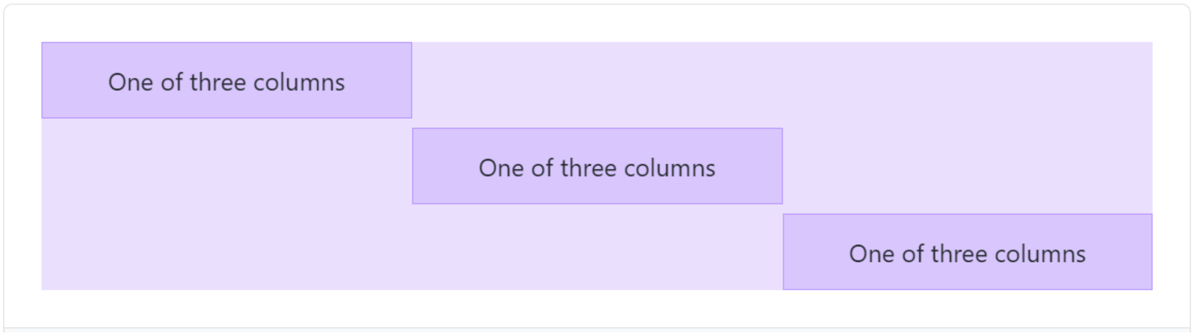
```
<div class="container text-center">
<div class="row align-items-center">
<div class="col">
One of three columns
</div>
<div class="col">
```

```
One of three columns
</div>
<div class="col">
One of three columns
</div>
</div>
</div>
```



```
<div class="container text-center">
<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>
```

Or, change the alignment of each column individually with any of the responsive `.align-self-*` classes.

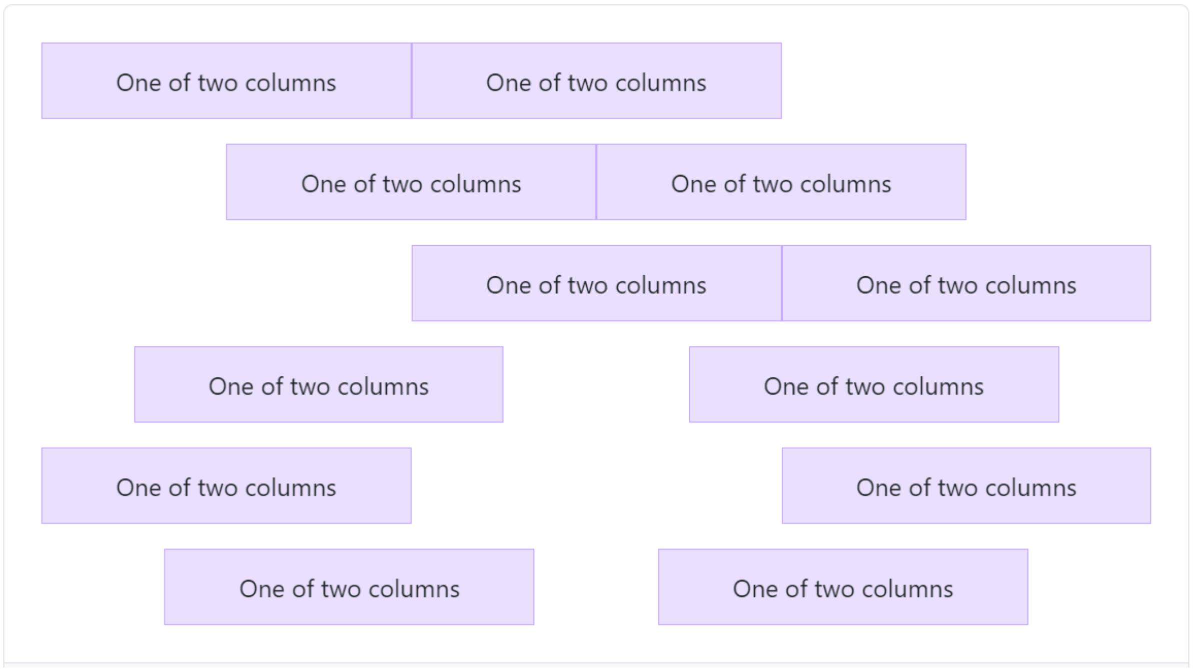


```
<div class="container text-center">
<div class="row">
<div class="col align-self-start">
One of three columns
</div>
<div class="col align-self-center">
```

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

Horizontal alignment

Change the horizontal alignment with any of the responsive `justify-content-*` classes.

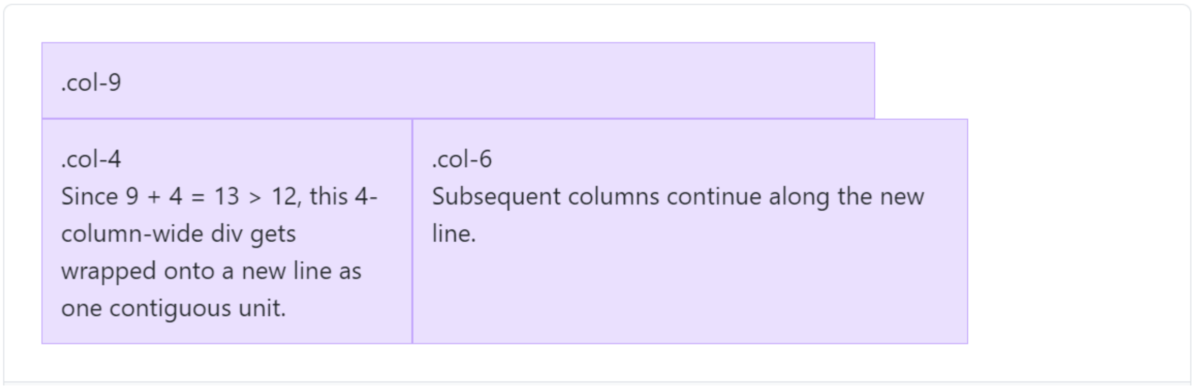


```
<div class="container text-center">
<div class="row justify-content-start">
<div class="col-4">
One of two columns
</div>
<div class="col-4">
One of two columns
</div>
</div>
<div class="row justify-content-center">
<div class="col-4">
One of two columns
</div>
<div class="col-4">
One of two columns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

Column wrapping

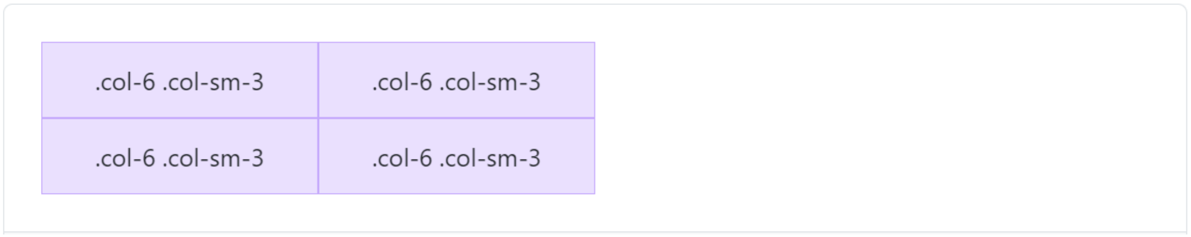
If more than 12 columns are placed within a single row, each group of extra columns will, as one unit, wrap onto a new line.


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

Column breaks

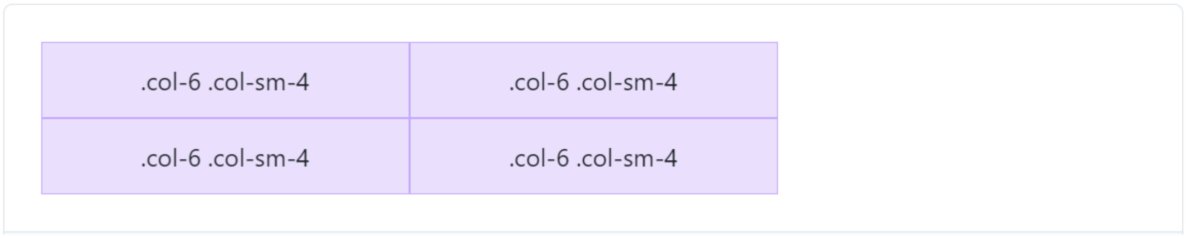
Breaking columns to a new line in flexbox requires a small hack: add an element with `width: 100%` wherever you want to wrap your columns to a new line. Normally this is accomplished with multiple `.row`s, but not every implementation method can account for this.



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

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

You may also apply this break at specific breakpoints with responsive display utilities.



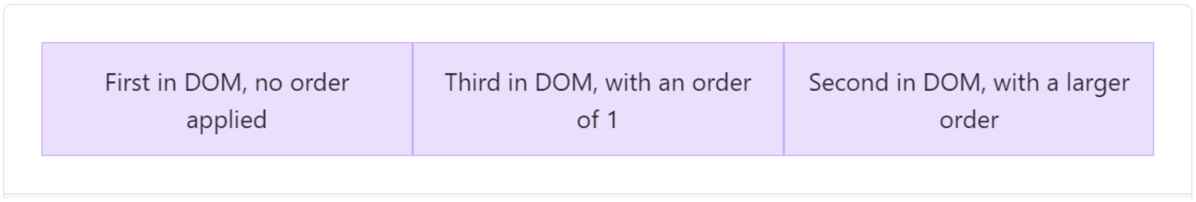
.col-6 .col-sm-4	.col-6 .col-sm-4
.col-6 .col-sm-4	.col-6 .col-sm-4

```
<div class="container text-center">
<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 bre
akpoint 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>
</div>
```

Reordering

Order classes

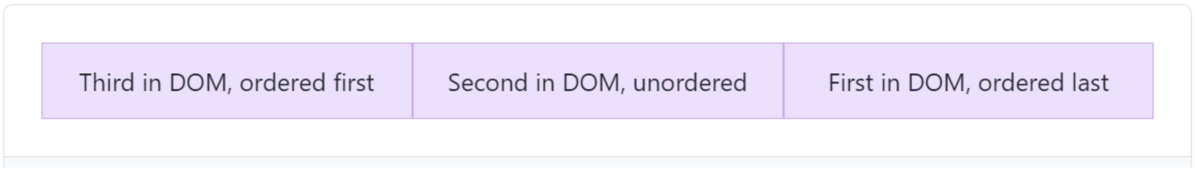
Use `.order-` classes for controlling the **visual order** of your content. These classes are responsive, so you can set the `order` by breakpoint (e.g., `.order-1.order-md-2`). Includes support for `1` through `5` across all six grid tiers. If you need more `.order-*` classes, you can modify the default number via Sass variable.



First in DOM, no order applied | Third in DOM, with an order of 1 | Second in DOM, with a larger order

```
<div class="container text-center">
<div class="row">
<div class="col">
First in DOM, no order applied
</div>
<div class="col order-5">
Second in DOM, with a larger order
</div>
<div class="col order-1">
Third in DOM, with an order of 1
</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: 6`, respectively. These classes can also be intermixed with the numbered `.order-*` classes as needed.



Third in DOM, ordered first | Second in DOM, unordered | First in DOM, ordered last

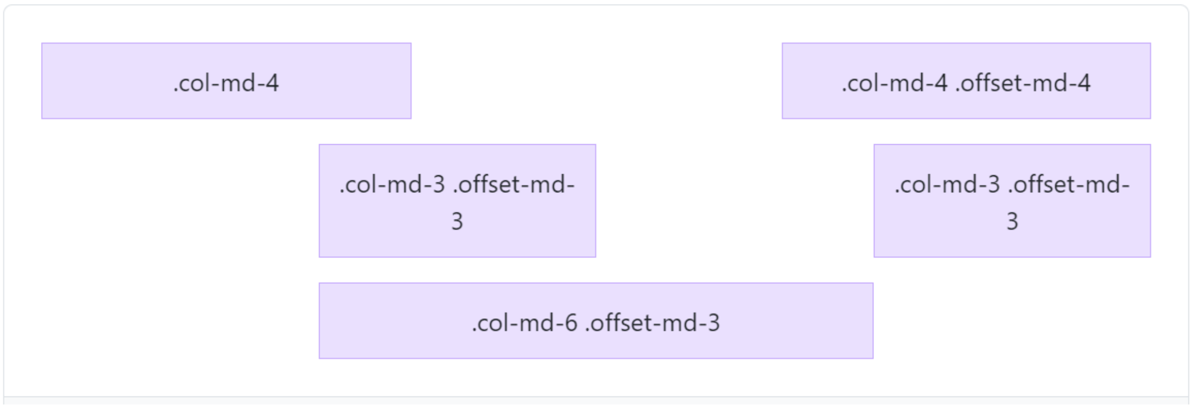
```
<div class="container text-center">
<div class="row">
<div class="col order-last">
First in DOM, ordered last
</div>
<div class="col">
Second in DOM, unordered
</div>
<div class="col order-first">
Third in DOM, ordered first
</div>
</div>
</div>
```

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.

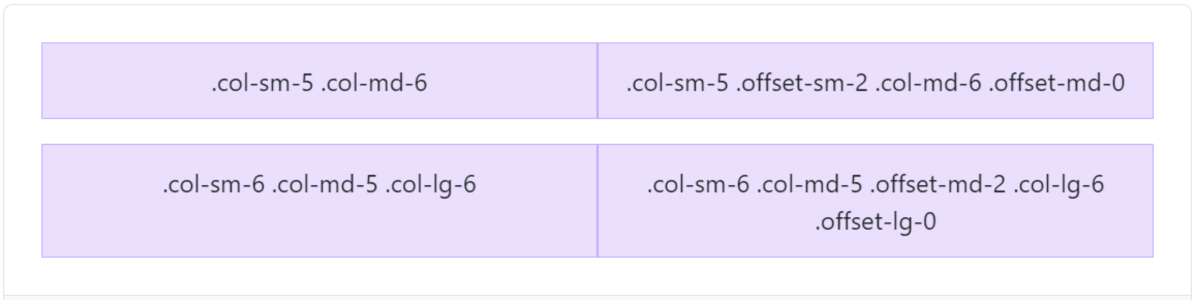
Offset classes

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.



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

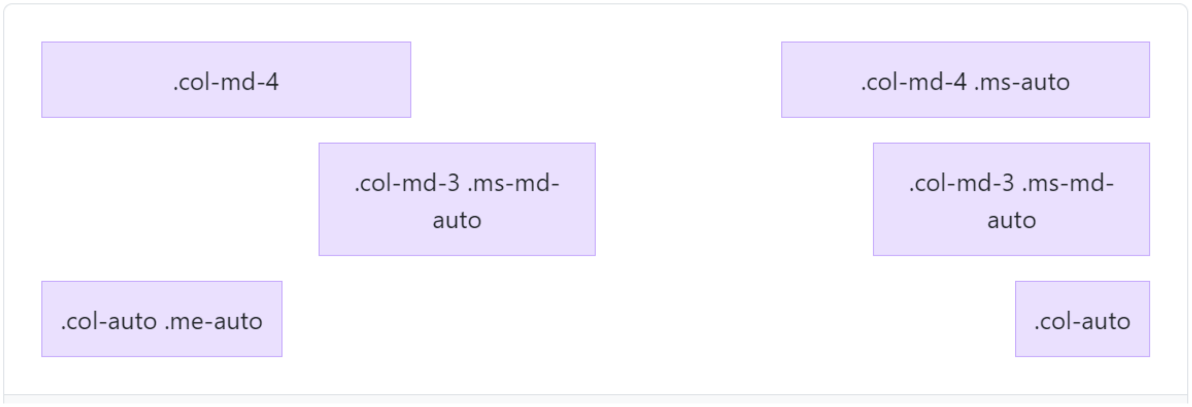
In addition to column clearing at responsive breakpoints, you may need to reset offsets.


.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="container text-center">
<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</di
v>
</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>
</div>
```

Margin utilities

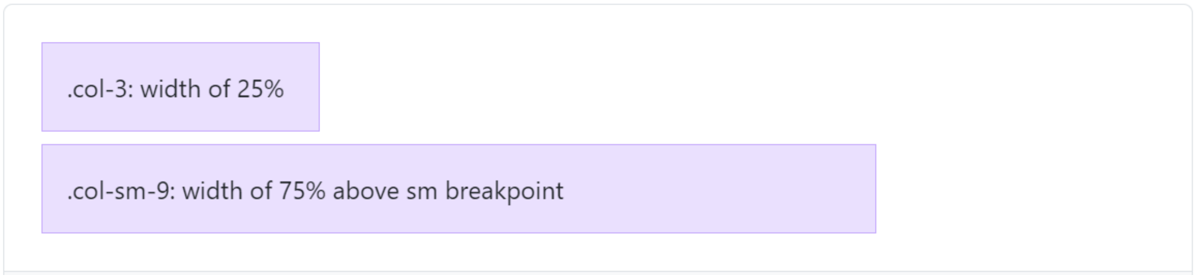
With the move to flexbox in v4, you can use margin utilities like `.me-auto` to force sibling columns away from one another.



```
<div class="container text-center">
<div class="row">
<div class="col-md-4">.col-md-4</div>
<div class="col-md-4 ms-auto">.col-md-4 .ms-auto</div>
</div>
<div class="row">
<div class="col-md-3 ms-md-auto">.col-md-3 .ms-md-auto</div>
<div class="col-md-3 ms-md-auto">.col-md-3 .ms-md-auto</div>
</div>
<div class="row">
<div class="col-auto me-auto">.col-auto .me-auto</div>
<div class="col-auto">.col-auto</div>
</div>
</div>
```

Standalone column classes

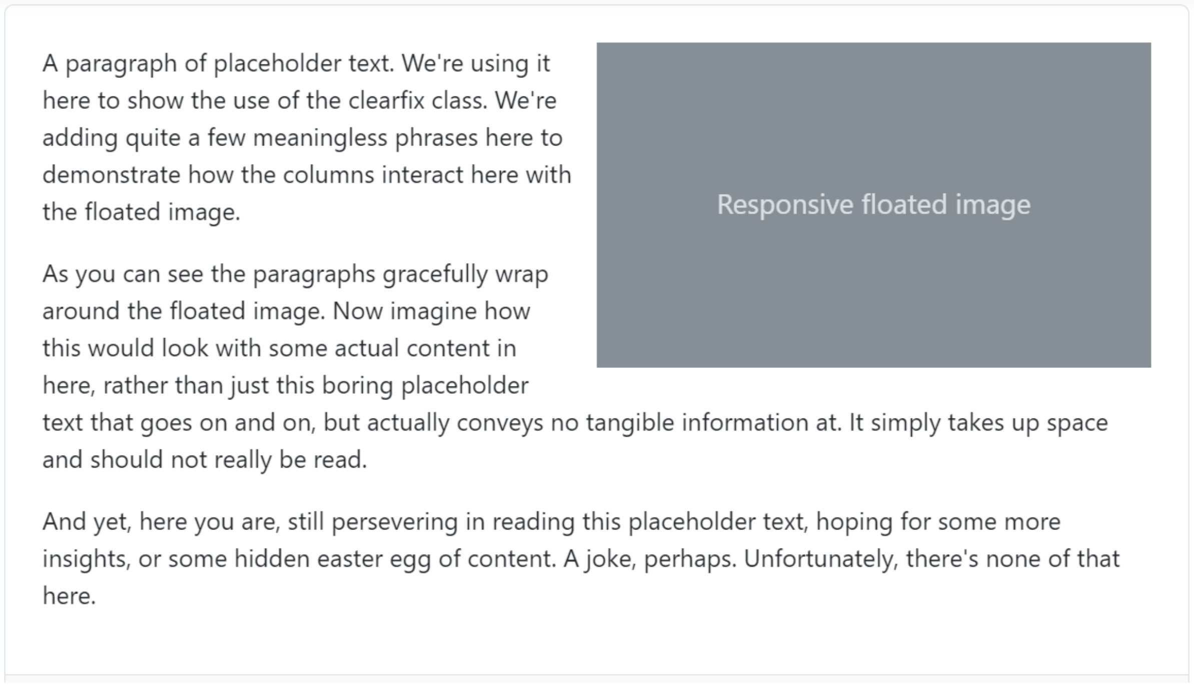
The `.col-*` classes can also be used outside a `.row` to give an element a specific width. Whenever column classes are used as non-direct children of a row, the paddings are omitted.


.col-3: width of 25%

.col-sm-9: width of 75% above sm breakpoint

```
<div class="col-3 p-3 mb-2">
.col-3: width of 25%
</div><div class="col-sm-9 p-3">
.col-sm-9: width of 75% above sm breakpoint
</div>
```

The classes can be used together with utilities to create responsive floated images. Make sure to wrap the content in a `.clearfix` wrapper to clear the float if the text is shorter.



A paragraph of placeholder text. We're using it here to show the use of the clearfix class. We're adding quite a few meaningless phrases here to demonstrate how the columns interact here with the floated image.

As you can see the paragraphs gracefully wrap around the floated image. Now imagine how this would look with some actual content in here, rather than just this boring placeholder text that goes on and on, but actually conveys no tangible information at. It simply takes up space and should not really be read.

And yet, here you are, still persevering in reading this placeholder text, hoping for some more insights, or some hidden easter egg of content. A joke, perhaps. Unfortunately, there's none of that here.

```
<div class="clearfix">
<img src="..." class="col-md-6 float-md-end mb-3 ms-md-3" alt="...">
<p>
A paragraph of placeholder text. We're using it here to show the use of the clearfix class. We're adding
quite a few meaningless phrases here to demonstrate how the columns interact here with the floated imag
e.
</p>
<p>
As you can see the paragraphs gracefully wrap around the floated image. Now imagine how this would look
 with some actual content in here, rather than just this boring placeholder text that goes on and on, bu
t actually conveys no tangible information at. It simply takes up space and should not really be read.
```

```
</p>
<p>
And yet, here you are, still persevering in reading this placeholder text, hoping for some more insight
s, or some hidden easter egg of content. A joke, perhaps. Unfortunately, there's none of that here.
</p>
</div>
```

5. Gutters

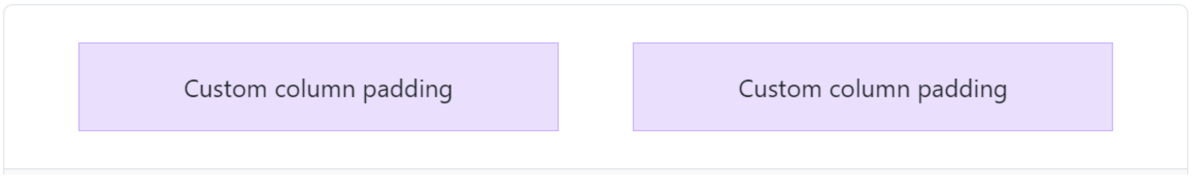
Gutters are the padding between your columns, used to responsively space and align content in the Bootstrap grid system.

How they work

- **Gutters are the gaps between column content, created by horizontal `padding`.** We set `padding-right` and `padding-left` on each column, and use negative `margin` to offset that at the start and end of each row to align content.
- **Gutters start at `1.5rem` (`24px`) wide.** This allows us to match our grid to the padding and margin spacers scale.
- **Gutters can be responsively adjusted.** Use breakpoint-specific gutter classes to modify horizontal gutters, vertical gutters, and all gutters.

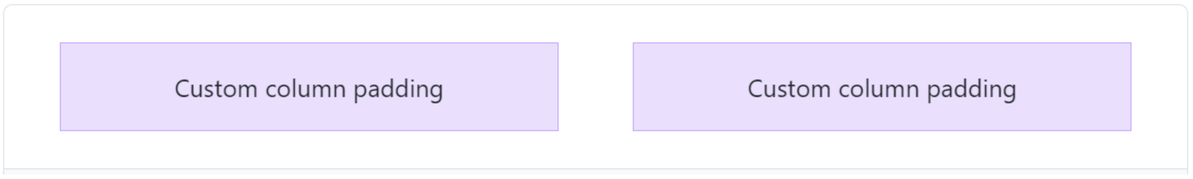
Horizontal gutters

`.gx-*` classes can be used to control the horizontal gutter widths. The `.container` or `.container-fluid` parent may need to be adjusted if larger gutters are used too to avoid unwanted overflow, using a matching padding utility. For example, in the following example we've increased the padding with `.px-4`:



```
<div class="container px-4 text-center">
<div class="row gx-5">
<div class="col">
<div class="p-3">Custom column padding</div>
</div>
<div class="col">
<div class="p-3">Custom column padding</div>
</div>
</div>
</div>
```

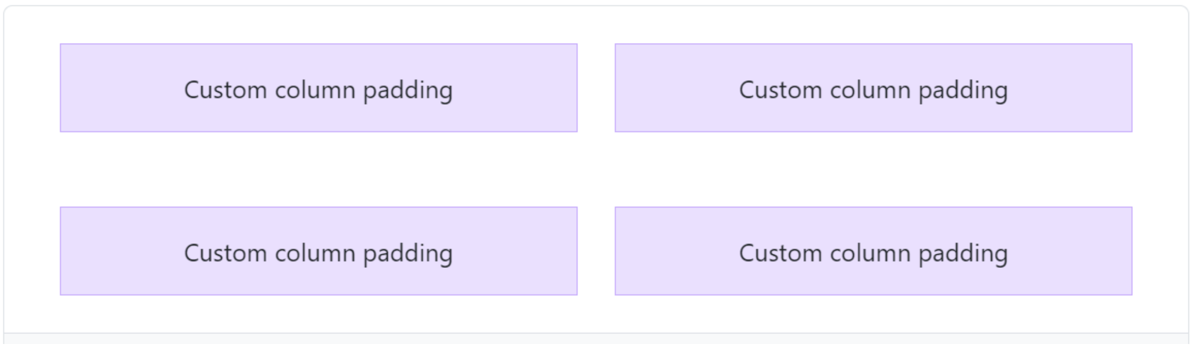
An alternative solution is to add a wrapper around the `.row` with the `.overflow-hidden` class:



```
<div class="container overflow-hidden text-center">
<div class="row gx-5">
<div class="col">
<div class="p-3">Custom column padding</div>
</div>
<div class="col">
<div class="p-3">Custom column padding</div>
</div>
</div>
</div>
```

Vertical gutters

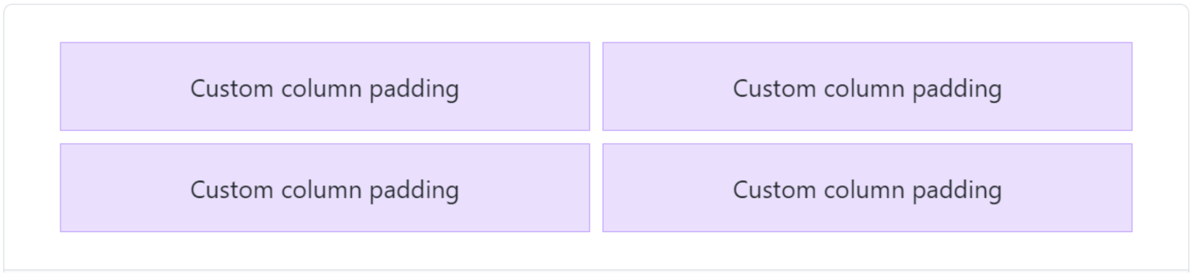
`.gy-*` classes can be used to control the vertical gutter widths within a row when columns wrap to new lines. Like the horizontal gutters, the vertical gutters can cause some overflow below the `.row` at the end of a page. If this occurs, you add a wrapper around `.row` with the `.overflow-hidden` class:



```
<div class="container overflow-hidden text-center">
<div class="row gy-5">
<div class="col-6">
<div class="p-3">Custom column padding</div>
</div>
<div class="col-6">
<div class="p-3">Custom column padding</div>
</div>
<div class="col-6">
<div class="p-3">Custom column padding</div>
</div>
<div class="col-6">
<div class="p-3">Custom column padding</div>
</div>
</div>
</div>
```

Horizontal & vertical gutters

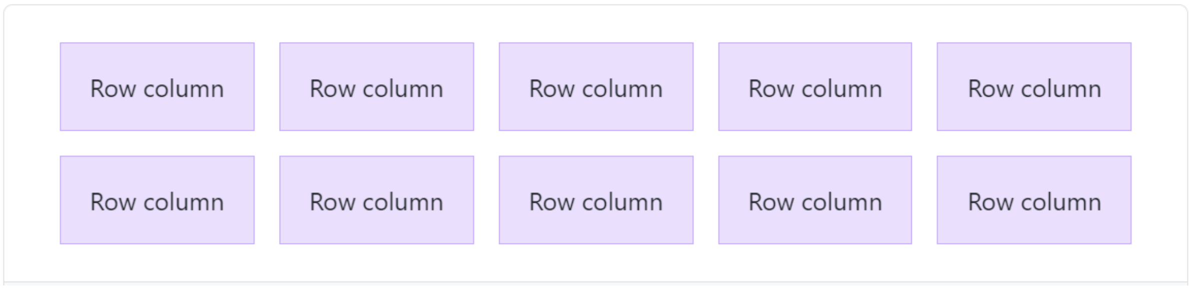
Use `.g-*` classes to control the horizontal and vertical grid gutters. In the example below, we use a smaller gutter width, so there isn't a need for the `.overflow-hidden` wrapper class.



```
<div class="container text-center">
<div class="row g-2">
<div class="col-6">
<div class="p-3">Custom column padding</div>
</div>
<div class="col-6">
<div class="p-3">Custom column padding</div>
</div>
<div class="col-6">
<div class="p-3">Custom column padding</div>
</div>
<div class="col-6">
<div class="p-3">Custom column padding</div>
</div>
</div>
</div>
```

Row columns gutters

Gutter classes can also be added to row columns. In the following example, we use responsive row columns and responsive gutter classes.



```
<div class="container text-center">
<div class="row row-cols-2 row-cols-lg-5 g-2 g-lg-3">
<div class="col">
<div class="p-3">Row column</div>
</div>
<div class="col">
```

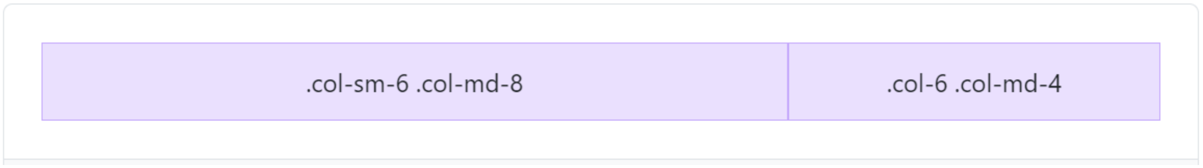
```
<div class="p-3">Row column</div>
</div>
<div class="col">
<div class="p-3">Row column</div>
</div>
<div class="col">
<div class="p-3">Row column</div>
</div>
<div class="col">
<div class="p-3">Row column</div>
</div>
<div class="col">
<div class="p-3">Row column</div>
</div>
<div class="col">
<div class="p-3">Row column</div>
</div>
<div class="col">
<div class="p-3">Row column</div>
</div>
<div class="col">
<div class="p-3">Row column</div>
</div>
<div class="col">
<div class="p-3">Row column</div>
</div>
</div>
</div>
```

No gutters

The gutters between columns in our predefined grid classes can be removed with `.g-0`. This removes the negative `margin`s from `.row` and the horizontal `padding` from all immediate children columns.

Need an edge-to-edge design? Drop the parent `.container` or `.container-fluid` and add `.mx-0` to the `.row` to prevent overflow.

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).



```
<div class="row g-0 text-center">
<div class="col-sm-6 col-md-8">.col-sm-6 .col-md-8</div>
<div class="col-6 col-md-4">.col-6 .col-md-4</div>
</div>
```

Change the gutters

Classes are built from the `$gutters` Sass map which is inherited from the `$spacers` Sass map.

```
$grid-gutter-width: 1.5rem;
$gutters: (
0: 0,
1: $spacer * .25,
2: $spacer * .5,
3: $spacer,
4: $spacer * 1.5,
5: $spacer * 3,
);
```

Source: getbootstrap.com