

# Layout

Options for structuring your pages with Bootstrap, including global styles, required scaffolding, grid system, and more.



Toptal matches you with  
top developers who are  
guaranteed to succeed.  
ads via Carbon

## Grid system

Bootstrap includes a powerful mobile-first grid system for building layouts of all shapes and sizes. It's based on a 12 column layout and has multiple tiers, one for each [media query range](#). You can use it with Sass mixins or our predefined classes.

Search...

[Getting started](#)

[Layout](#)

[Overview](#)

[Grid](#)

[Flexbox grid](#)

[Media object](#)

[Responsive utilities](#)

[Content](#)

[Components](#)

[Utilities](#)

[About](#)

[Migration](#)

## Contents

- [How it works](#)
- [Quick start example](#)
- [Grid options](#)
- [Sass mixins](#)
  - [Variables](#)
  - [Mixins](#)
  - [Example usage](#)

:s

cked-to-horizontal

obile and desktop

obile, tablet, desktop

lumn wrapping

sponsive column resets

setting columns

sting columns

lumn ordering

grid

gutters

## CS

ow the grid system works:

ajor components—containers, rows, and

ntainer for fixed width or

id for full width—center your site's contents

ur grid content.

ital groups of columns that ensure your  
l up properly.

re placed within columns, and only columns  
te children of rows.

ndicate the number of columns you'd like to  
ossible 12 per row. So if you want three  
mns, you'd use `.col-sm-4`.

are set in percentages, so they're always fluid  
o their parent element.

orizontal padding to create the gutters  
al columns.

d tiers, one for each responsive breakpoint:  
, medium, large, and extra large.

ed on minimum widths, meaning they apply to  
all those above it (e.g., `.col-sm-4` applies to  
rge, and extra large devices).

efined grid classes or Sass mixins for more

.

Let's move on to seeing all that in an example.

## example

ap's compiled CSS, this is the example you'll want

One of three  
columns      One of three  
columns

Copy

```
iner">  
">  
ol-sm-4">  
e columns  
  
ol-sm-4">  
e columns  
  
ol-sm-4">  
e columns
```

Creates three equal-width columns on small,  
extra large devices using our [predefined grid](#)  
; are centered in the page with the parent

## S

`em` s or `rem` s for defining most sizes, `px` s are  
nts and container widths. This is because the  
els and does not change with the [font size](#).

Bootstrap grid system work across multiple  
able.

	<b>Extra small</b> <576px	<b>Small</b> ≥576px	<b>Medium</b> ≥768px
Horizontal at all times	Collapsed to start, horizontal		
<b>h</b>	None (auto)	540px	720px
	.col-xs-	.col-sm-	.col-md-
12			
	30px (15px on each side of a column)		
Yes			
Yes			
Yes			

In source Sass files, you have the option of defining mixins to create custom, semantic, and units. Our [predefined grid classes](#) use these same mixins to provide a whole suite of ready-to-use classes and units.

Determine the number of columns, the gutter width, the trigger point at which to begin floating columns. You can also use the predefined grid classes documented above or create your own custom mixins listed below.

[Copy](#)

```
12;  
h-base: 30px;  
  
hs: (  
r-width-base, // 30px  
r-width-base, // 30px  
r-width-base, // 30px  
r-width-base, // 30px  
r-width-base // 30px
```

```
: (  
screen / phone  
  
/ phone
```

```
n / tablet
```

```
/ desktop
```

```
screen / wide desktop
```

```
dths: (
```

junction with the grid variables to generate dual grid columns.

Copy

```
per for a series of columns
gutters: $grid-gutter-widths) {
x {
;
ap;

rfix();
```

```
nt in map-keys($gutters) {
a-breakpoint-up($breakpoint) {
p-get($gutters, $breakpoint);
t: ($gutter / -2);
: ($gutter / -2);
```

```
nt grid-ready (applying everything but
```

```
eady($gutters: $grid-gutter-widths) {
ive;
; // Prevent collapsing
```

```
mns from becoming too narrow when at
s by
ng `width: 100%;`. This works because
ues
override this initial width.
x {
```

```
nt in map-keys($gutters) {
a-breakpoint-up($breakpoint) {
p-get($gutters, $breakpoint);
ht: ($gutter / 2);
t: ($gutter / 2);
```

```
size, $columns: $grid-columns) {
x {
centage($size / $columns);
-width` to ensure content within each
low out
```

```
of the column. Applies to IE10+ and
nd Safari
ear to require this.
percentage($size / $columns);

tag($size / $columns);

ffsetting, or changing the sort order
ffset($size, $columns: $grid-columns) {
percentage($size / $columns);

ush($size, $columns: $grid-columns) {
> 0, percentage($size / $columns),
```

## ge

variables to your own custom values, or just use default values. Here's an example of using the `grid-column-gap` utility to create a two-column layout with a gap between.

[View rendered example.](#)

Copy

```
;  
ontainer();  
  
ow();  
  
ol-ready();  
  
th: 32em) {  
-col(6);  
  
th: 32.1em) {  
-col(8);  
  
y {  
ol-ready();  
  
th: 32em) {  
-col(6);  
  
th: 32.1em) {  
-col(4);  
  
iner">  
">  
ontent-main">...</div>  
ontent-secondary">...</div>
```

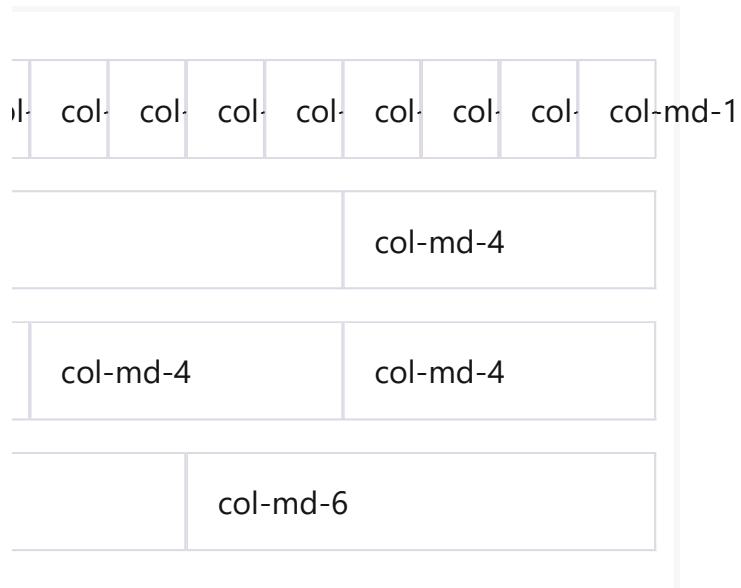
Copy

## classes

antic mixins, Bootstrap includes an extensive for quickly creating grid columns. It includes ed column sizing, reordering columns, and

## Stacked-to-horizontal

With `col-md-*` grid classes, you can create a basic layout stacked on mobile devices and tablet (or small range) before becoming horizontal on larger devices. Place grid columns within any `.row`.



Copy

```
-md-1">col-md-1</div>
-  
-md-1">col-md-1</div>
```

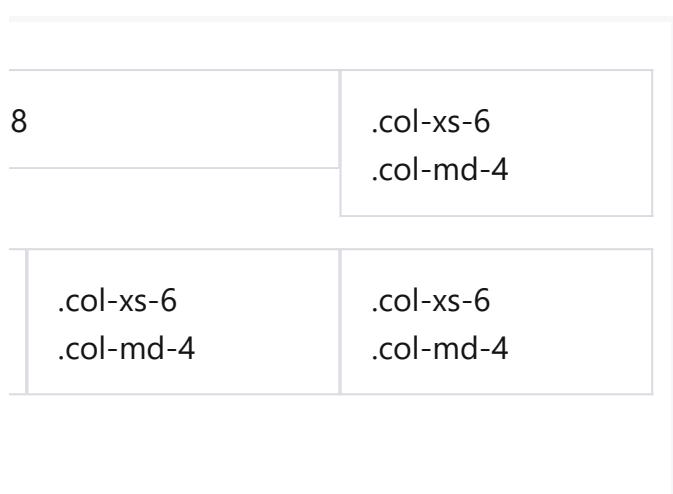
```
-  
-md-8">col-md-8</div>
-  
-md-4">col-md-4</div>
```

```
-  
-md-4">col-md-4</div>
-  
-md-4">col-md-4</div>
-  
-md-4">col-md-4</div>
```

```
-  
-md-6">col-md-6</div>
-  
-md-6">col-md-6</div>
```

## bile and desktop

ins to simply stack in smaller devices? Use the  
n device grid classes by adding `.col-xs-*`  
ur columns. See the example below for a better  
s.





```
.col-xs-6
```

[Copy](#)

lumns on mobile by making one  
e other half-width -->

```
-xs-12 col-md-8">.col-xs-12 .col-md-  
-xs-6 col-md-4">.col-xs-6 .col-md-
```

t at 50% wide on mobile and bump up to  
ktop -->

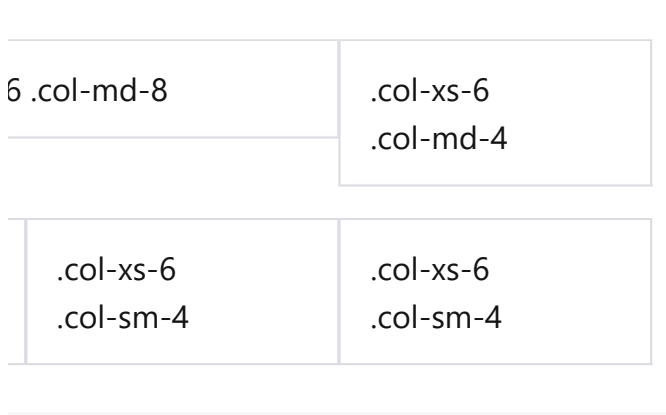
```
-xs-6 col-md-4">.col-xs-6 .col-md-  
-xs-6 col-md-4">.col-xs-6 .col-md-  
-xs-6 col-md-4">.col-xs-6 .col-md-
```

always 50% wide, on mobile and desktop

```
-xs-6">.col-xs-6</div>  
-xs-6">.col-xs-6</div>
```

## mobile, tablet, desktop

example by creating even more dynamic and  
ablet `.col-sm-*` classes.



Copy

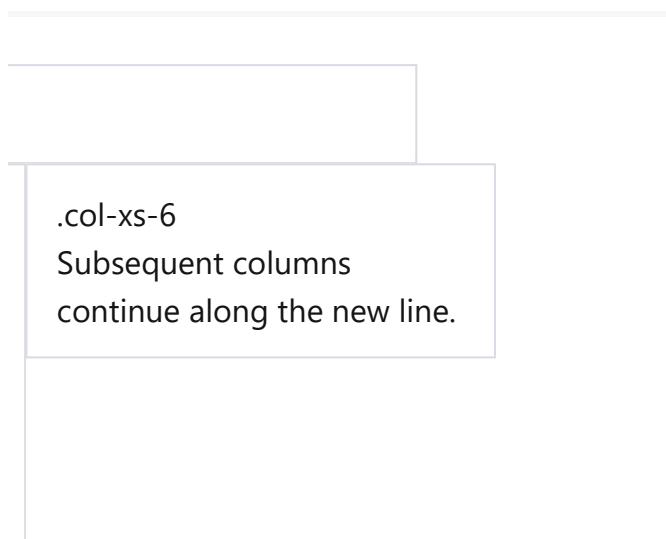
```
-xs-12 col-sm-6 col-md-8">.col-xs-12  
-8</div>  
-xs-6 col-md-4">.col-xs-6 .col-md-
```

```
-xs-6 col-sm-4">.col-xs-6 .col-sm-  
-xs-6 col-sm-4">.col-xs-6 .col-sm-
```

```
clear the XS cols if their content  
height -->  
arfix hidden-sm-up"></div>  
-xs-6 col-sm-4">.col-xs-6 .col-sm-
```

## umn wrapping

ns are placed within a single row, each group of  
one unit, wrap onto a new line.

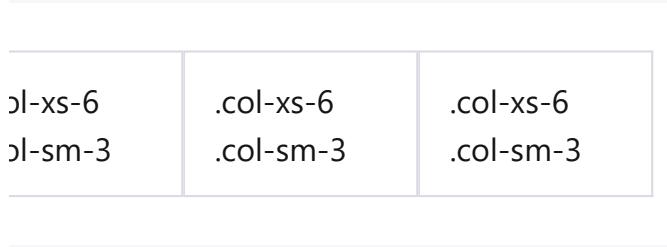


 Copy

```
-xs-9">.col-xs-9</div>
-xs-4">.col-xs-4<br>Since 9 + 4 = 13
column-wide div gets wrapped onto a new
guous unit.</div>
-xs-6">.col-xs-6<br>Subsequent columns
e new line.</div>
```

## Responsive column resets

If you stack columns available you're bound to run into issues at certain points, your columns don't clear quite right as they stack. To fix that, use a combination of a [responsive utility classes](#).



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

 Copy

```
-xs-6 col-sm-3">.col-xs-6 .col-sm-
```

```
-xs-6 col-sm-3">.col-xs-6 .col-sm-
```

tra clearfix for only the required

```
arfix hidden-sm-up"></div>
```

```
-xs-6 col-sm-3">.col-xs-6 .col-sm-
```

```
-xs-6 col-sm-3">.col-xs-6 .col-sm-
```

clearing at responsive breakpoints, you may

**pushes, or pulls.** See this in action in [the grid](#)

6

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

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

Copy

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

## setting columns

ight using `.offset-md-*` classes. These  
t margin of a column by `*` columns. For  
`4` moves `.col-md-4` over four columns.

ol-md-3  
ffset-md-3

.col-md-4  
.offset-md-4

.col-md-3  
.offset-md-3

```
col-md-6 .offset-md-3
```

Copy

```
-md-4">.col-md-4</div>  
-md-4 offset-md-4">.col-md-4 .offset-
```

```
-md-3 offset-md-3">.col-md-3 .offset-
```

```
-md-3 offset-md-3">.col-md-3 .offset-
```

```
-md-6 offset-md-3">.col-md-6 .offset-
```

## nesting columns

With the default grid, add a new `.row` and set `s` within an existing `.col-sm-*` column. Include a set of columns that add up to 12 or don't use all 12 available columns).

```
Level 2: .col-xs-4  
.col-sm-6
```

Copy

```
-sm-9">  
-sm-9  
ow">  
"col-xs-8 col-sm-6">  
.col-xs-8 .col-sm-6  
  
"col-xs-4 col-sm-6">  
.col-xs-4 .col-sm-6
```

## umn ordering

of our built-in grid columns with `11-md-*` modifier classes.

```
ol-md-9 .push-md-3
```

Copy

```
-md-9 push-md-3">.col-md-9 .push-
```

```
-md-3 pull-md-9">.col-md-3 .pull-
```

## g the grid

Sass variables and maps, it's possible to change the predefined grid classes. Change the media query dimensions, and the container size.

## l gutters

lumns and their horizontal padding (aka,

ed via Sass variables. `$grid-columns` is used (in percent) of each individual column while allows breakpoint-specific widths that are `padding-left` and `padding-right` for the

Copy

```
12 !default;  
h-base: 30px !default;  
hs: (  
r-width-base,  
r-width-base,  
r-width-base,  
r-width-base,  
r-width-base
```

lumns themselves, you may also customize the you wanted just three grid tiers, you'd update `s` and `$container-max-widths` to something

Copy

```
: (  
  
dths: (
```

nges to the Sass variables or maps, you'll need and recompile. Doing so will out a brand new classes for column widths, offsets, pushes, and lity utilities will also be updated to use the

[GitHub](#)   [Twitter](#)   [Examples](#)   [About](#)

Designed and built with all the love in the world by @mdo and @fat. Maintained by the core team with the help of our contributors.

Currently v4.0.0-alpha.5. Code licensed MIT, docs CC BY 3.0.