

Grid

We are using a standard 12 column fluid responsive grid system. The grid helps you layout your page in an ordered, easy fashion.



Get 10 Wishes
From Genshin Impact
[Start Now](#)

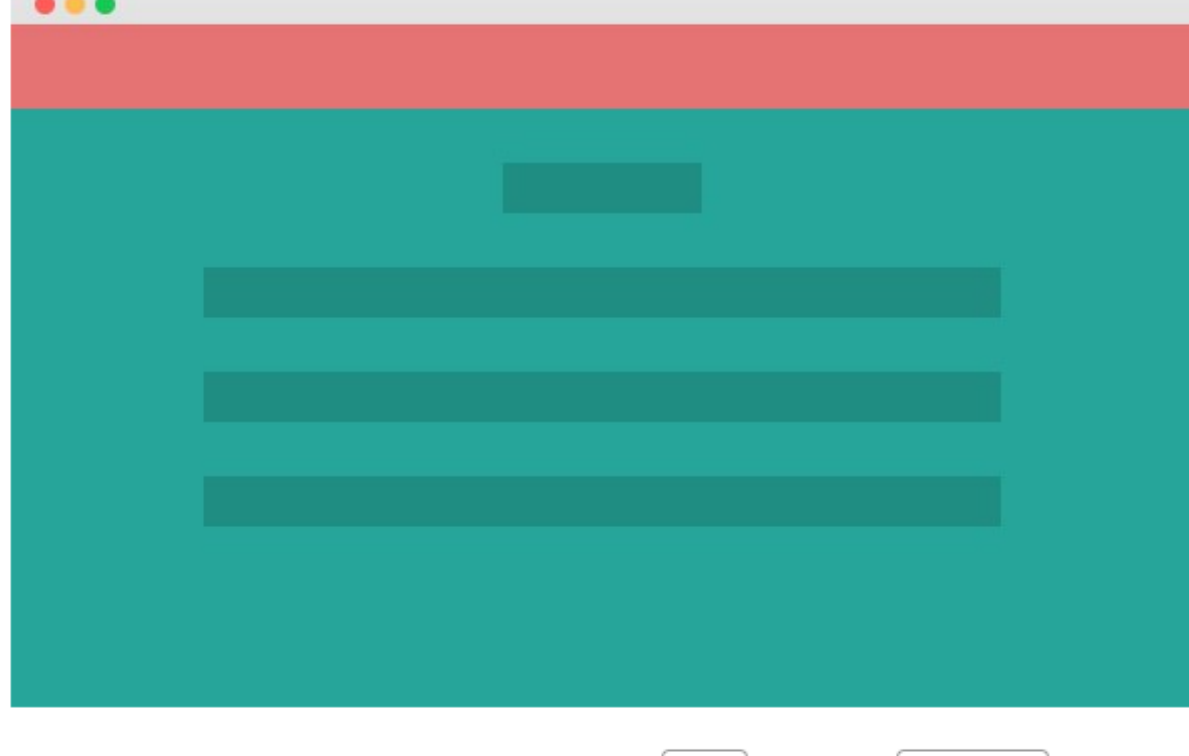
Container

The container class is not strictly part of the grid but is important in laying out content. It allows you to center your page content. The `container` class is set to ~70% of the window width. It helps you center and contain your page content. We use the container to contain our body content.

Demo

Try the button below to see what the page looks like without containers.

TURN OFF CONTAINER



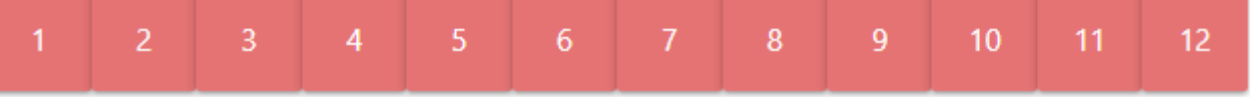
To add a container just put your content inside a `<div>` tag with a `container` class. Here's an example of how your page might be set up.

Introduction

Take a look at this section to quickly understand how the grid works!

12 Columns

Our standard grid has 12 columns. No matter the size of the browser, each of these columns will always have an equal width.

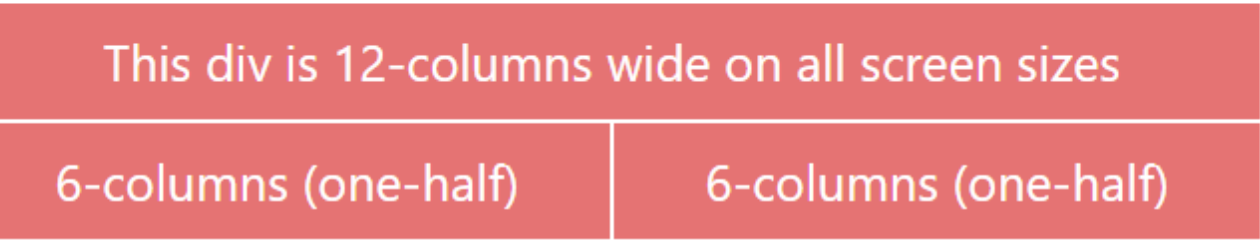


To get a feel of how the grid is used in HTML, take a look at the code below which will produce a similar result to the one above.

Note: For now, just know that the `s1` stands for small-1 which in plain English means "1 column on small screens".

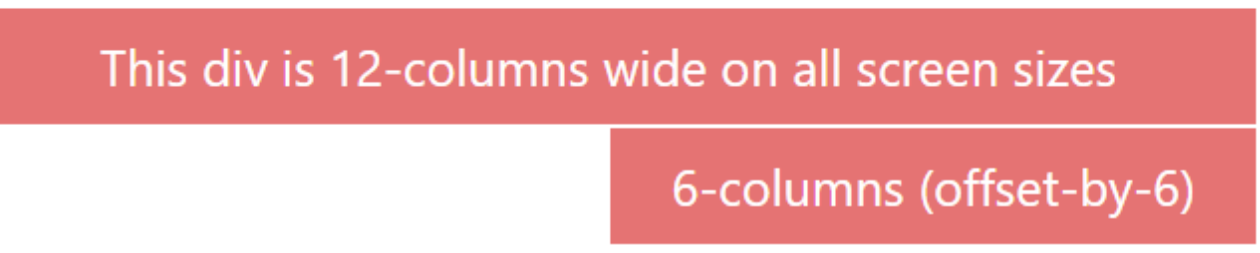
Columns live inside Rows

Remember when you are creating your layout that all columns must be contained inside a row and that you must add the `col` class to your inner divs to make them into columns



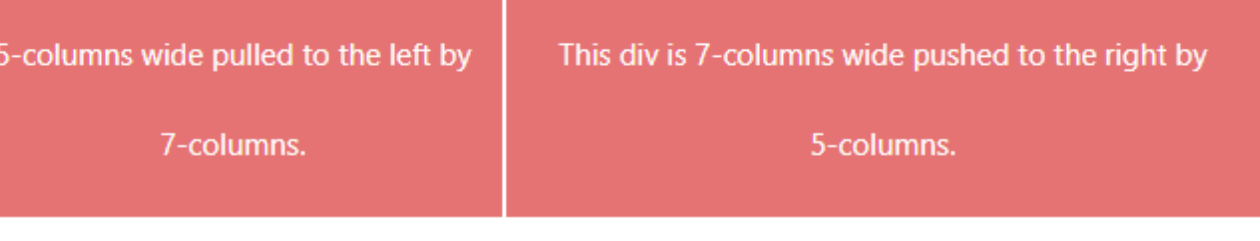
Offsets

To offset, simply add `offset-s2` to the class where `s` signifies the screen class-prefix (s = small, m = medium, l = large) and the number after is the number of columns you want to offset by.



Push and Pull

You can easily change the order of your columns with push and pull. Simply add `push-s2` or `pull-s2` to the class where `s` signifies the screen class-prefix (s = small, m = medium, l = large) and the number after is the number of columns you want to push or pull by.



Creating Layouts

Here we will show you how to create some commonly used layouts with our grid system. Hopefully these will get you more comfortable with laying out elements. To keep these demos simple, the ones here will not be responsive.

Section

The section class is used for simple top and bottom padding. Just add the `section` class to your div's containing large blocks of content.

Divider

Dividers are 1 pixel lines that help break up your content. Just add the `divider` to a div in between your content.

Example Sections and Dividers

Section 1

Stuff

Section 2


Stuff

Section 3

Stuff


Example Promotion Table

If we want 3 divs that are equal size, we define the divs with a width of 4-columns, as 4+4+4 nicely adds up to 12. Inside those divs, we can put our content. Take our front page content for example. We've modified it slightly for the sake of this example.




Speeds up development

We did most of the heavy lifting for you to provide a default stylings that incorporate our custom components.



User Experience Focused

By utilizing elements and principles of Material Design, we were able to create a framework that focuses on User Experience.



Easy to work with

We have provided detailed documentation as well as specific code examples to help new users get started.

Example Side Navigation Layout

You can see how easy it is to create layouts using the grid system. Just remember to make sure your column numbers add up to 12 for an even layout.



Creating Responsive Layouts

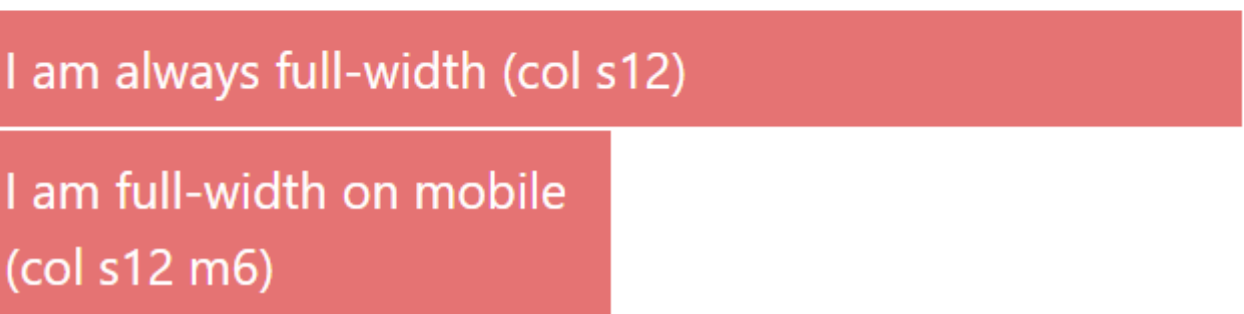
Above we showed you how to layout elements using our grid system. Now we'll show you how to design your layouts so that they look great on all screen sizes.

Screen Sizes

	Mobile Devices <= 600px	Tablet Devices > 600px	Desktop Devices > 992px	Large Desktop Devices > 1200px
Class Prefix	<code>.s</code>	<code>.m</code>	<code>.l</code>	<code>.xl</code>
Container Width	90%	85%	70%	70%
Number of Columns	12	12	12	12

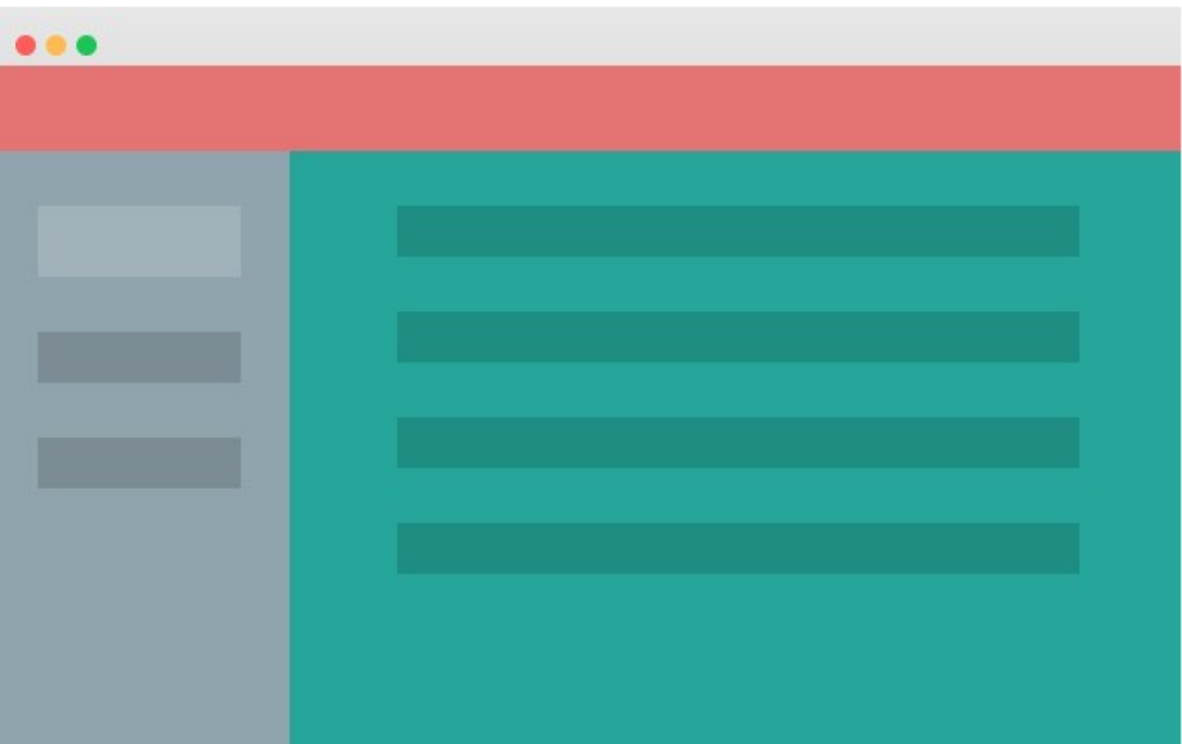
Adding Responsiveness

In the previous examples, we only defined the size for small screens using `"col s12"`. This is fine if we want a fixed layout since the rules propagate upwards. By just saying `s12`, we are essentially saying `"col s12 m12 l12"`. But by explicitly defining the size we can make our website more responsive.



Responsive Side Navigation Layout

In this example below, we take the same layout from above, but we make it responsive by defining how many columns the div should take up on each screen size. Try resizing your browser and watch the layout change below.



More Responsive Grid Examples

