Сетка в Bootstrap 4. Подробное руководство

[](https://medium.com/@stasonmars?source=post_header_lockup)

[Stas Bagretsov](https://medium.com/@stasonmars)Follow

Jul 5, 2018

Понимание сеток в Bootstrap это самый важный момент работы с фреймворком, понимая который вы сможете делать на нем без препятствий буквально всё, что захотите. Советую почитать статьи [Узнаем Bootstrap 4 за 30 минут, создавая лендинг](https://medium.com/@stasonmars/%D1%83%D0%B7%D0%BD%D0%B0%D0%B5%D0%BC-bootstrap-4-%D0%B7%D0%B0-30-%D0%BC%D0%B8%D0%BD%D1%83%D1%82-%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%B2%D0%B0%D1%8F-%D0%BB%D0%B5%D0%BD%D0%B4%D0%B8%D0%BD%D0%B3-d268d52d6c84) и [Самые полезные хитрости в Bootstrap](https://medium.com/@stasonmars/%D1%81%D0%B0%D0%BC%D1%8B%D0%B5-%D0%BF%D0%BE%D0%BB%D0%B5%D0%B7%D0%BD%D1%8B%D0%B5-%D1%85%D0%B8%D1%82%D1%80%D0%BE%D1%81%D1%82%D0%B8-%D0%B2-bootstrap-f194058dc177)

[*Перевод статьи How the Bootstrap 4 Grid Works*](https://uxplanet.org/how-the-bootstrap-4-grid-works-a1b04703a3b7)

**Что такое система сеток в Bootstrap?**

Система сеток в Bootstrap используется для шаблона, а в частности для создания адаптивных шаблонов. Понимание того, как это работает — является жизненно важным моментом в понимании Bootstrap. Сетка сделана из группирования строк и колонок внутри одного или нескольких контейнеров.

Сетки Bootstrap могут использоваться отдельно, без Bootstrap JavaScript и других CSS компонентов. Вам надо только скачать и сослаться на "bootstrap-grid.css", который включает в себя flexbox классы и классы для сетки. Вообще более подробно для этого есть отдельная документация [на официальном сайте](http://getbootstrap.com/docs/4.1/getting-started/contents/#css-files).

Вот самый простой пример применения сетки:

<div class="container">

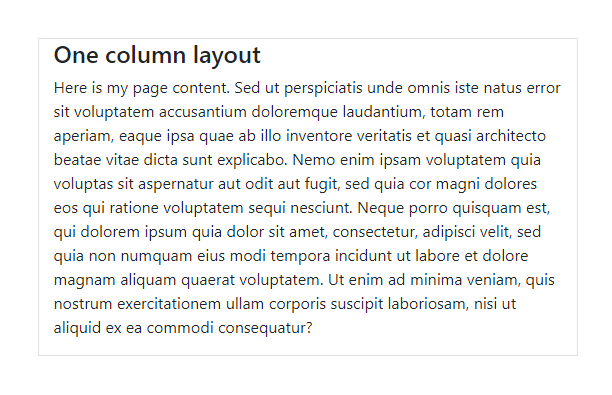
<div class="row">

<div class="col">I’m your content inside the grid!</div>

</div>

</div>

Что даст нам одну большую колонку на всю ширину вьюпорта.



А вот две колонки:

<div class="container">

<div class="row">

<div class="col">Left column</div>

<div class="col">Right column</div>

</div>

</div>



И вот на три колонки:

<div class="container">

<div class="row">

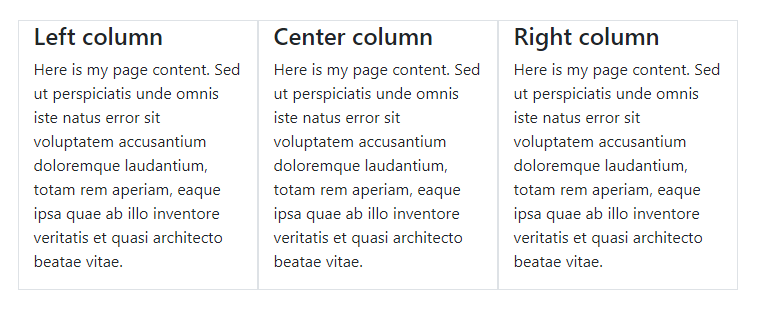
<div class="col">Left column</div>

<div class="col">Center column</div>

<div class="col">Right column</div>

</div>

</div>



Обратите внимание, что светлые серые линии границ вокруг колонок в примере были добавлены, чтобы вы визуально могли выделить каждую колонку. Вот рабочие [шаблоны на Codeply](https://www.codeply.com/go/tTDmka8GUe).

Простые концепты сетки очень быстро можно понять, но вы уже наверное начинаете удивляться, почему вся эта HTML разметка так необходима. У вас уже возможно есть несколько вопросов, таких как: Зачем мне нужен контейнер? Могу ли я сделать один контейнер шире, че другие?

Я коснусь этих вопросов немного позже. Но сначала, я хочу сделать шаг назад и объяснить кое-что важное по поводу использования сеток.

Поверьте мне. Понимание **“Привил сетки”** сэкономит вам кучу времени и не даст расстроиться раньше времени. Читайте их очень внимательно.

**Правила сетки:**

*Колонки должны быть прямыми потомками строк*

*Строки используются только для того, чтобы включать в себя колонки и не для ничего больше.*

*Строки должны быть помещены внутри контейнера*

Эти правила **ОЧЕНЬ ВАЖНЫ**. Строки и колонки всегда работают вместе и вам надо смотреть за тем, что один не оставался без другого.

Ждите косяка, если вы не будете соблюдать эти три простые правила, определенно. Я ответил на бесчисленное количество вопросов на Stack Overflow, просто применяя эти правила. По началу, это может быть довольно сложным, но это реально станет легким, после того, как вы поймете то, как работает сетка.

Как использовать сетку Bootstrap. Правильный подход.

**Контейнер**

В простом примере до этого, вы возможно подметили, что я использовал .container, чтобы обернуть .row. Контейнер это вообще ключевой элемент сетки в Bootstrap.

<div class="container">

<div class="row">

<div class="col">I’m content inside the grid!</div>

</div>

</div>

Контейнер может использоваться для хранения любых элементов и контента. Он используется не только для строк и колонок сетки. Для примера, вот вот идеальная и правильная разметка Bootstrap:

<div class="container">

<h2>My Heading</h2>

<div class="row">

<div class="col">I’m content inside the grid!</div>

</div>

</div>

**Не игнорируйте контейнер**

С первого взгляда, контейнер может показаться пустяковым и не сильно уж нужным, но он очень важен для контроля ширины шаблона. Контейнер также используется для равномерного выравнивания граней шаблона внутри вьюпорта браузера. Контейнер используется, чтобы противодействовать отрицательным внешним отступам строки, о чем я расскажу немного попозже.

У Bootstrap 4 есть 2 типа контейнера. В моих примерах я использовал .container, но также есть и полноэкранный .container-fluid. Вы можете использовать любой из них:

1 — Контейнер с фиксированной шириной, для центровки контейнера по середине шаблона.

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

2 — Контейнер с шириной во весь экран.

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

.container масштабируется адаптивно по ширине экрана, так что в конце концов он может стать шириной на весь экран, как и .container-fluid, но на маленьких устройствах.

Помните, что контейнер может использоваться для содержания любого контента, а не только строк и колонок сетки. Но! Если вы используете последние два, то строки сетки должны быть размещены внутри контейнера. Попробуйте [демо для контейнера](https://www.codeply.com/go/TVQwGJrAFF).

Применяя сетку, более одной строки будет помещено внутри контейнера. Вы можете иметь сколько угодно их в контейнере и вы так же можете иметь сколько хотите контейнеров на странице. Все зависит от того, какой шаблон вы пытаетесь сверстать, но пока что не сильно беспокойтесь по этому поводу.

У строк (rows) есть отрицательные левые/правые внешние отступы в -15px. Внутренний отступ контейнера в 15px используется для пресечения срабатывания отрицательных внешних отступов в строке контейнера. Это делается для равномерного выравнивания по граням в шаблоне. Если вы не поместите строку (row) в контейнер, то она будет выходить за пределы своего контейнера, вызывая нежелательные горизонтальные прокрутки.

**Строки (Rows) и Колонки (Columns)**

Сейчас бы я хотел, чтобы .row в Bootstrap не были на самом деле именованы как “строки” (row). Название “строка”, зачастую запутывающее и скрывает настоящее предназначение .row.

Когда вы думаете о строке, вы возможно думаете о горизонтальной линии, что ок, НО, лучше думать о строке, как о родителе для колонок.

Думайте о строках, как о группе колонок (Columns)

Это потому, что колонки внутри .row не всегда располагаются горизонтально вдоль вьюпорта. Иногда нам надо, что колонки в шаблоне были горизотальны, а иногда нам надо, чтобы они располагались вертикально. Концепция горизонтального vs. Вертикального шаблона является сущностью адаптивного дизайна. Единственным предназначением “строки”, является содержание одной или более “колонки”.

Не вставляйте контент прямо в “строку”!

Так делать нельзя:

<div class="row">

This is very bad, wrong way, no bueno!!

</div>

<div class="row">

<p>This is also very bad, the wrong way!!</p>

</div>

<div class="row">

<h2>No headings either! This is the wrong way!!</h2>

</div>

“колонки” и только колонки, размещаются внутри “строк”.

А контент размещается внутри “колонок”.

<div class="row">

<div class="col">Happy :-) This is the right way.</div>

</div>

Так же очень важно упомянуть, что .row имеет display: flex. А как потомок в Flexbox, “колонка” в каждой строке одной и той же высоты.

Благодаря Flexbox, горизонтальное и вертикальное выравнивание легко делается с использованием рабочих [классов Bootstrap 4 — flex и auto-margin](http://getbootstrap.com/docs/4.1/utilities/flex/).

А теперь настало время взглянуть глубже на строки и колонки и то как они на самом деле работают вместе. Есть разные типы колонок и разные способы использования их в шаблоне. Они как магия.



**Счастье это, колонки!**

Создают горизонтальные разделители по вьюпорту.

Могут иметь разные определения ширины.

Шаблон горизонтально слева на право, вертикально сверху вниз.

Могут изменять позицию (порядок) относительно родственных элементов в той же строке.

Имеют туже высоту, что и другие родственные элементы в той же строке.

Могут “расти” или “урезаться” по ширине.

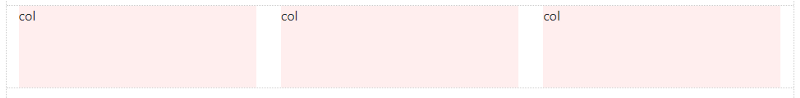
Могут автоматически обертываться или “сталкиваться” вертикально, если надо или при другой ширине экрана.

Могут содержать больше строк и колонок при вложении.

**Всё что нужно знать о колонках Bootstrap**

Снова не забываем, что колонки должны быть прямыми потомками строк.

И так, колонки создают горизонтальные деления по вьюпорту. Пространство между колонками называется “gutter”.



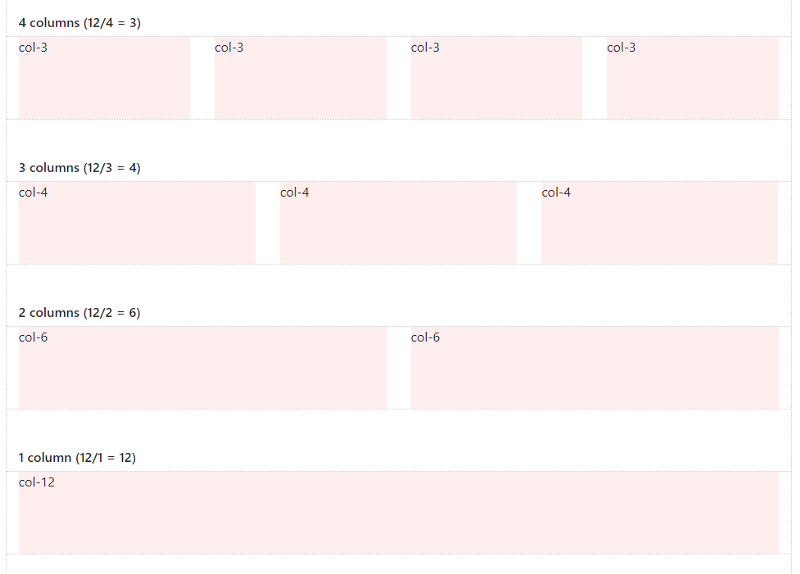
Классическая сетка Bootstrap имеет 12 колонок:



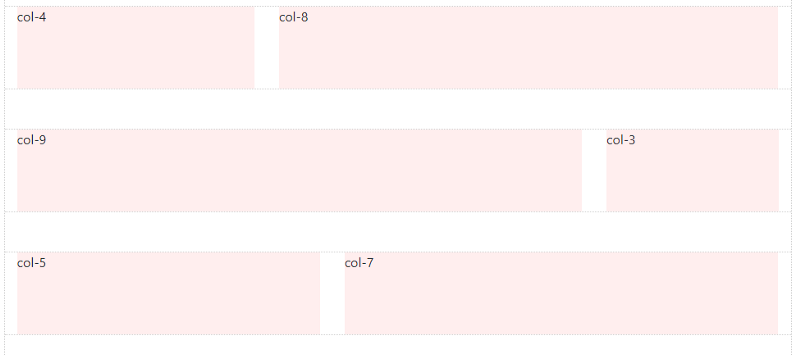
Таким образом, колонки могут быть равномерно разделены на 12 частей. Вот пример, 6 колонок (12/6=2) :



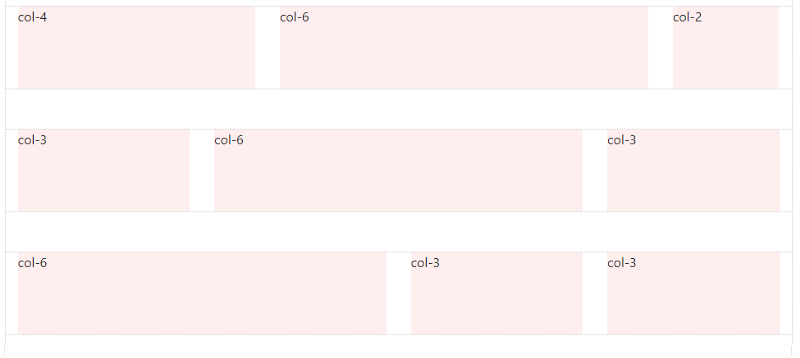
И вы можете делать вычисления:



Колонки могут быть разделены с использованием любой части из 12 элементов. И это нормально — использовать меньше 12. И также это нормально — использовать больше 12-ти, это я покажу вам дальше.



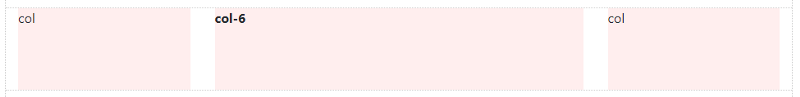
Учитывая такую гибкость, возможности шаблона кажутся бесконечными.



Но сетка не всегда может быть из 12 элементов. Спасибо flexbox, у Bootstrap 4 есть новые “auto-layout” колонки. Такие безразмерные колонки дают вам больше гибкости, когда нужно разрабатывать шаблоны.

Теперь вы знаете, как использовать колонки для создания горизонтального шаблона. Но подождите, есть ещё кое что! Давайте поговорим о некоторых модных штучках, которые могут делать колонки.

Как вы видели выше, колонки могут быть разной ширины:



А вы знали, что ширина колонки может изменяться в зависимости от ширины экрана?



Это называется адаптивный дизайн и я расскажу вам как это на самом деле работает.

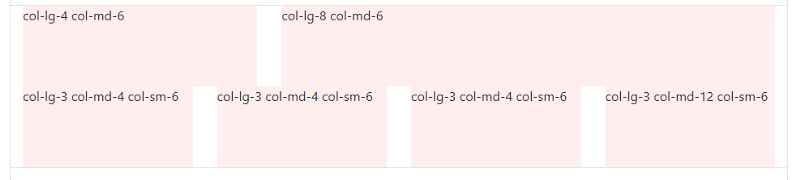
Но все по порядку, мне нужно закончить рассказывать вам о колонках. Помните я говорил вам до этого, что можно использовать больше 12-ти колонок в строке?

Колонки в одной строке располагаются вдоль горизонтально, а затем встают вертикально вниз. Это вертикальное сталкивание или “обертывание”, происходит когда элементы колонок в одном ряду превышают количество двенадцати. Этот [процесс известен как “Обертывание колонок”](http://getbootstrap.com/docs/4.1/layout/grid/#column-wrapping)

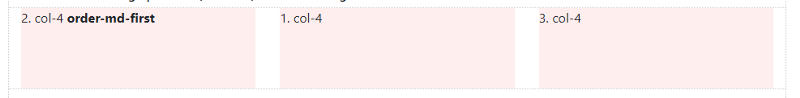
Колонки в одной строке переходят на следующую строку, каждые 12 элементов:



Ширина колонки и “обертывание” может контролироваться, используя разные ряды адаптивной сетки. (a.k.a “Grid Breakpoints”)



Колонки могут менять позицию относительно потомков в том же ряду:



Колонки могут содержать других потомков Rows & Columns. Это называется вложением:



Колонки могут “расти” и “урезаться” по ширине. Это auto-layout колонки:



**Как делать адаптивный дизайн с Bootstrap.**

Ряды сетки, медиа запросы и прерывания, ю-хууу!

Я еще не прекратил говорить о рядах и колонках, так как они звезды шоу, когда речь заходит об адаптивном дизайне.

**Всё из-за ширины.**

В Bootstrap 4 есть 5 адаптивных рядов (ну или моментов прерываний), которые вы возможно заметили в предидущем примере. (ie; col-lg-4, col-md).

Адаптивные моменты прерываний, основаны на ширине экрана:

(xs) — ширина экрана < 576px. Это стандарт.

sm — ширина экрана ≥ 576px

md — ширина экрана ≥ 768px

lg — ширина экрана ≥ 992px

xl — ширина экрана ≥ 1200px

Почему я заключил (xs) в скобки, а не сделал как с другими прерываниями? Так как xs это дефолтное прерывание, -xs инфикс, который использовался в Bootstrap 3, больше не используется в Bootstrap 4. Так что вместо col-xs-6, просто col-6.

Bootstrap использует медиа запросы CSS, что установить адаптивные точки прерываний. Они дают вам возможность контролировать поведение колонок при разных размерах экрана.

Для примера: вот 2 колонки, каждая шириной 50%:

<div class="container">

<div class="row">

<div class="col-sm-6">Column 1</div>

<div class="col-sm-6">Column 2</div>

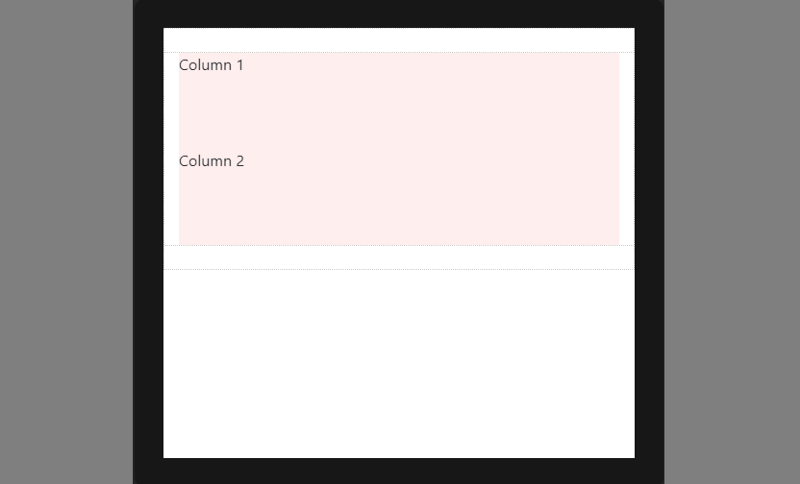
</div>

</div>

Col-sm-6 означает использование 6 колонок из 12, то есть 50% ширины на типичных маленьких размерах экранов. Больше или равные 768px:



На экранах меньше, чем 768px, 2 колонки станут шириной 100% и встанут вертикально:



Это происходит, потому что (xs) по дефолту или подразумевает точку прерывания. Так как я не указывал дефектную ширинку колонки, 50% ширина была применена только на 768px и шире для sm прерывания.

**Mobile-first!**

Так как (xs) это дефолтная точка прерывания, то подразумевает col-12. Следовательно:

<div class="col-12 col-sm-6">Column</div>

Работает так же, как и:

<div class="col-sm-6">Column</div>

xs(default) > переписывает sm > переписывает md > переписывает lg > переписывает xl

Или в обратном порядке… xl > переписывает lg > переписывает md> переписывает sm > переписывает (xs)

Следовательно, col-sm-6 реально означает 50% ширины на мелких и выше дисплеях. Для одинаковой ширины колонок на всех уровнях, просто выставите ширину для самого маленького ряда как хотите:

<div class="col-lg-3 col-md-3 col-sm-3">..</div> тоже самое, что и:

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

Для разной ширины колонки на разных уровнях, используйте подходящие прерывания для перезаписи маленьких прерываний. Для примера, 3 колоночная ширина на sm и 4 колоночная ширина на md и выше:

<div class="col-sm-3 col-md-4">..</div>

Auto-layout колонки в Bootstrap 4 также работают адаптивно. Из-за их простоты, теперь я предпочитаю использовать их, а не классические элементы 12 колонок. Auto-layout колонки идеальны для любых сценариев шаблонов, где равная ширина колонок обязательна. Но не забудьте, что 12-ти колоночные юниты могут быть смешаны, если надо.

**Давайте посмотрим на auto-layout сетку.**

И так, 3 равные колонки. ‘col’ остаются горизонтальным на всей широте и не встают вертикально, так как xs прерывание дефолтно:

<div class="container">

<div class="row">

<div class="col">1</div>

<div class="col">2</div>

<div class="col">3</div>

</div>

</div>

3 равные колонки, адаптивные. В этом примере, ‘col’ остаются горизонтальным до прерывания sm на 576px, а затем они становятся вертикальными. Помните, что вы можете менять sm на какой угодно (md,lg,xl), если нужно:

<div class="container">

<div class="row">

<div class="col-sm">1</div>

<div class="col-sm">2</div>

<div class="col-sm">3</div>

</div>

</div>

2 колонки, левый сайдбар. А вот пример комбинирования классически определнной ширины колонок с колонками auto-layout. Правая колонка будет автоматически расти, чтобы занять ширину, так как мы используем auto-layout .col. Сайдбар будет 16.6% ширины на больших экранах и затем встанет выше на sm прерывании с 576px:

<div class="container">

<div class="row">

<div class="col-sm-2">sidebar</div>

<div class="col">main content</div>

</div>

</div>

3 колонки, правый сайдбар (сокращение, чтобы уместиться): В этом примере есть левый сайдбар, центральная область контента и правый сайдбар, который сокращается по ширине, чтобы подстроиться под свой контент.

<div class="container">

<div class="row">

<div class="col-sm-2">left</div>

<div class="col">main content</div>

<div class="col-auto">right</div>

</div>

</div>

Ключевые моменты адаптивного дизайна используемые в сетке Bootstrap 4:

Колонки встанут вертикально и станут шириной во весь экран на устройствах с маленьким разрешением, если вы не используете col-\* класс в HTML разметке. Используйте col-\* для пререкания такого вертикального выстраивания.

Классы сеток поменьше, также применяются на больших экранах, пока не перепишутся конкретно под ширину большего экрана. Следовательно, <div class="col-md-6"></div> в сущности тоже самое, что и <div class="col-md-6 col-lg-6"></div>. Следовательно, вам только надо использовать класс для самых маленьких разрешений, которые вам нужно поддерживать.

Строки (row) — display: flex и следовательно колонки имеют равную высоту в одном и том же ряду. Используйте auto-margin или Flexbox align-item и justify-content для горизонтального или вертикального выравнивания.