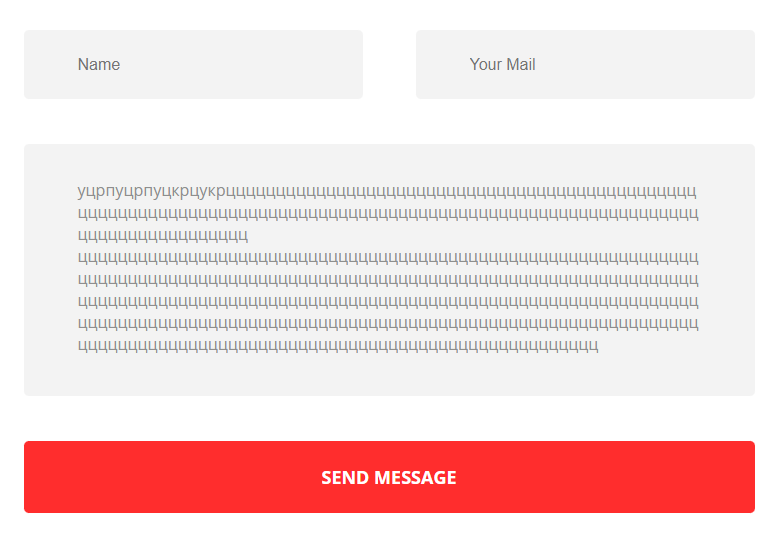
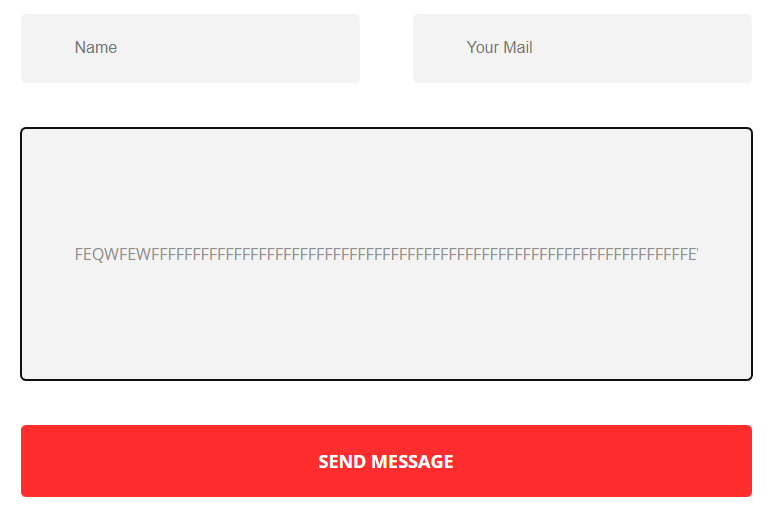
**A5.1 Содержание модуля**

Библиотека **Bootstrap 4.** Но сналача о прошлом. Они в форме вместо **input** использовали **textarea.** А в чем разница? Вот



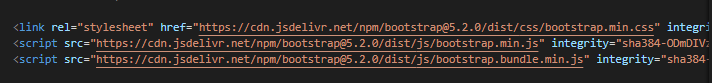
**A5.3 Знакомимся с CDN, подключаем CSS, шаблон-заготовка**

CDN – это распределенная сеть серверов, которые размещают в открытом доступе. Много библиотек можно так найти. Итак **Bootstrap 4.**

Bootstrap - это CSS-фреймфорк, позволяющий создавать сложные и при этом легко конфигурируемые интерфейсы с помощью классов. Сервис популярный и заметно ускоряет скорость верстки сайта.

Скачать его можно здесь. <https://www.bootstrapcdn.com/>

Но буду ли я устанавливать старый 4-ый? Лучше поставлю 5-ый. Думаю там лишь больше фич.

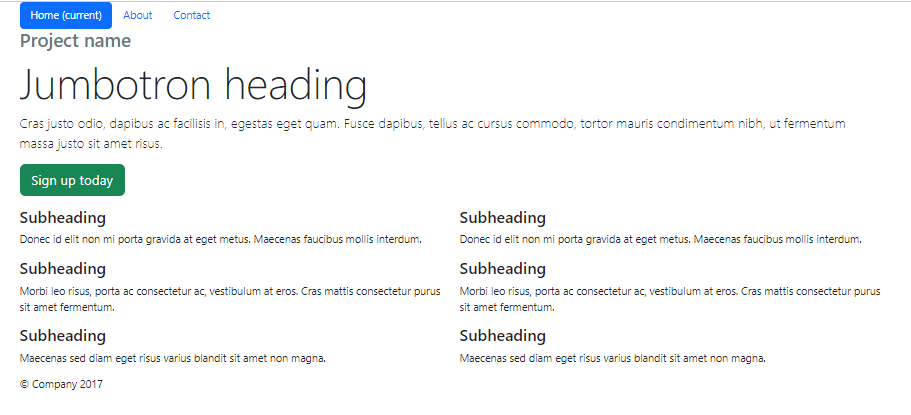


Поставил **js, css** и какой-то **bundle**

****

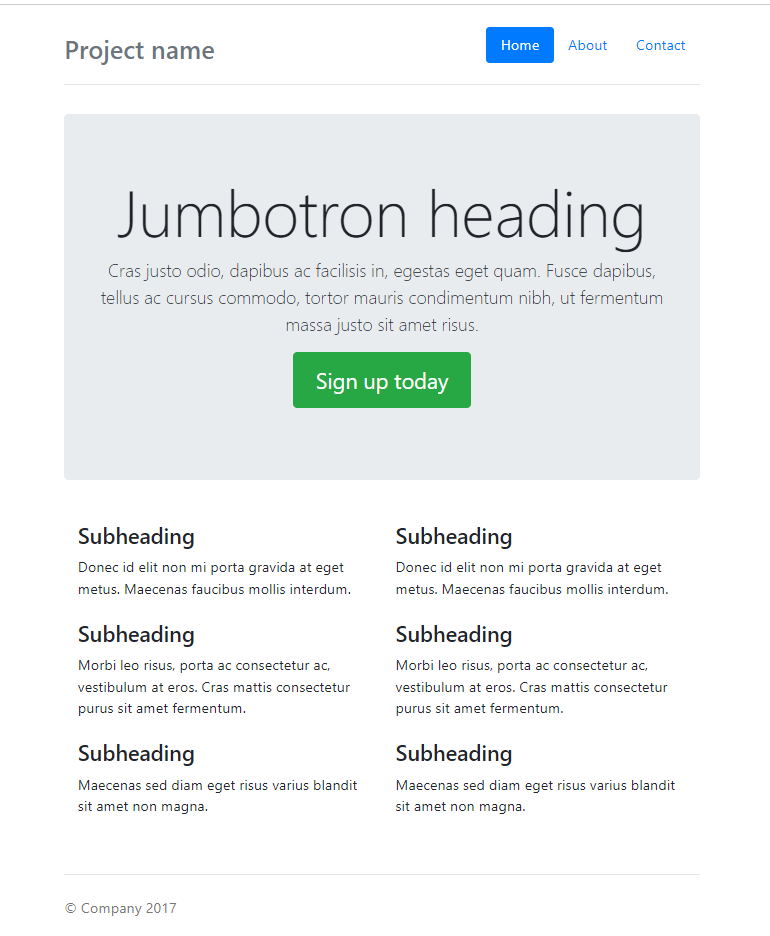
**OK**

Так вот, я скопировал код с другой страницы, где используется **bootstrap и…**

****

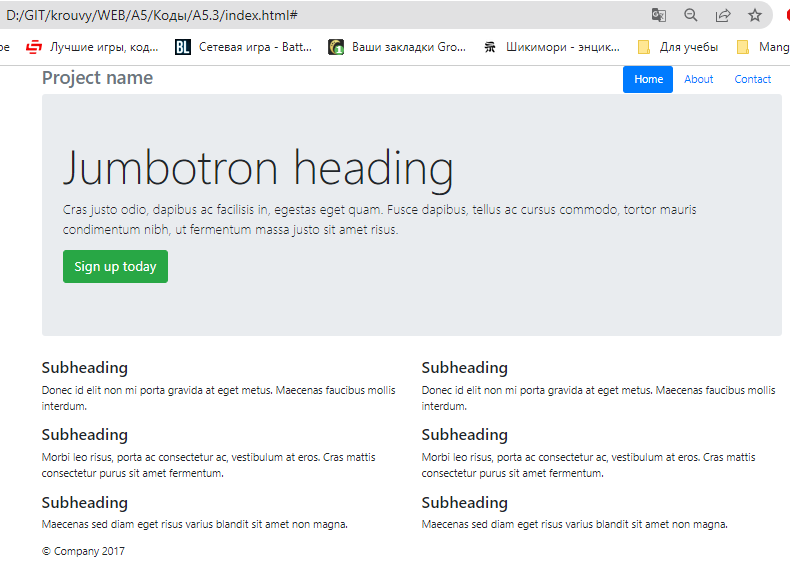
Получил это.

А должен был это:

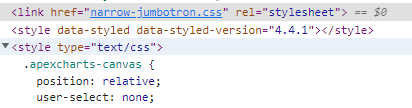
****

Ладно. А если bootsrap 4?

**Ладно с bootstrap 4 пашет также.** Буду следовать инструкциям.

****

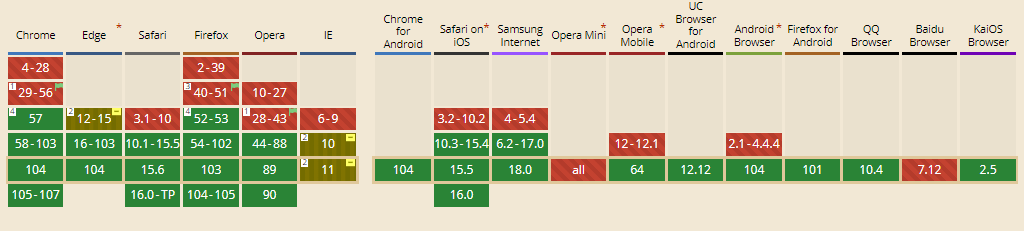
Даже так есть местами отличия, но не страшно. Дело в том, что на той странице есть еще дополнительные ссылки на стили.



Главное, что видно. Bootstrap подключен.

**A5.4 Сетки: построение, использование**

Назначение сеток. Это полезный инструмент при разработке интерфейсов. В новом стандарте **CSS** появилась новая спецификация **Grid Layout –** набор правил для построения интерфейсов в нативной и более удобной форме. Стандарт поддерживается современными браузерами.

****

Научимся же использовать этот грид. А то флексы уже приелись наверно. Основная цель, облегчить создание сложных макетов без сторонних библиотек. Bootstap не единственный фрейморк, поддерживающий сетки. Его аналони это:

**Foundation**

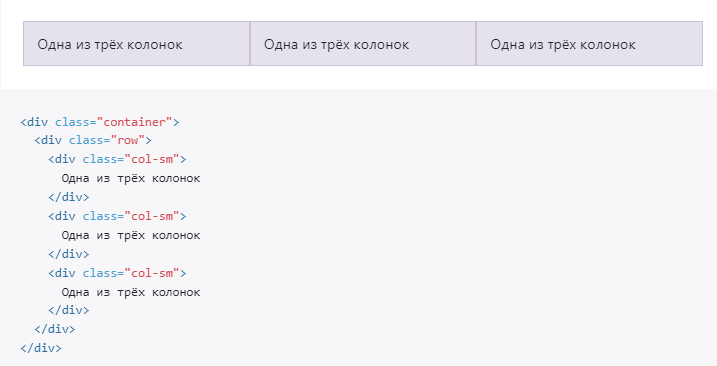
**Bulma**

**Skeleton**

**Bootstrap Grid** нужна для разметки страницы. Если конкретно, то для адаптивных макетов.

Сетка – структура похожая на таблицу. У нее есть общий контейнер обертка(**container**), в нем располагаются ряды**(row),** каждый из которых может содержать от 1 до 12 столбцов(**col**). Колонка тоже может быть контейнером и содержать в себе еще дочерние элементы.

Бутстрап реализован с помощью флексбокса и полностью адаптивен. Оформляется это с помощью классов и дивов.



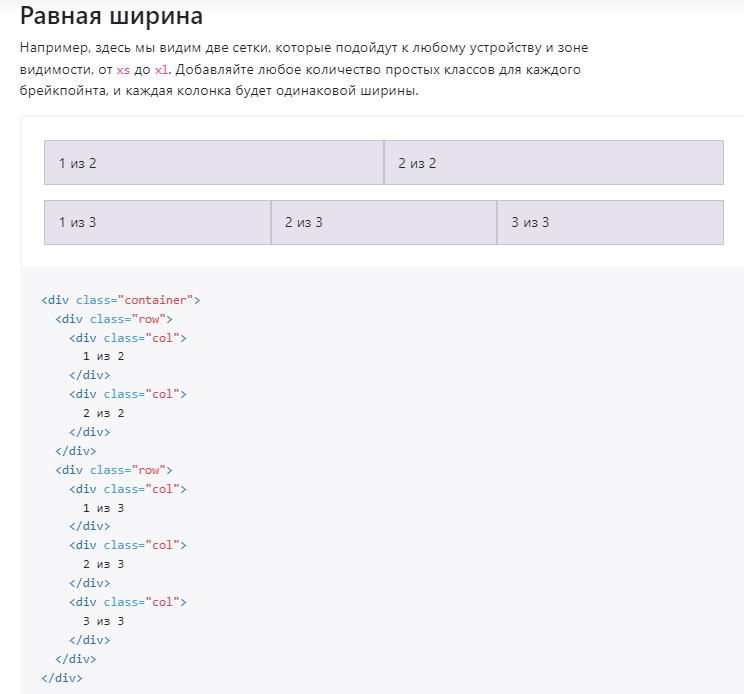
Создана строка с тремя колонками равного размера.

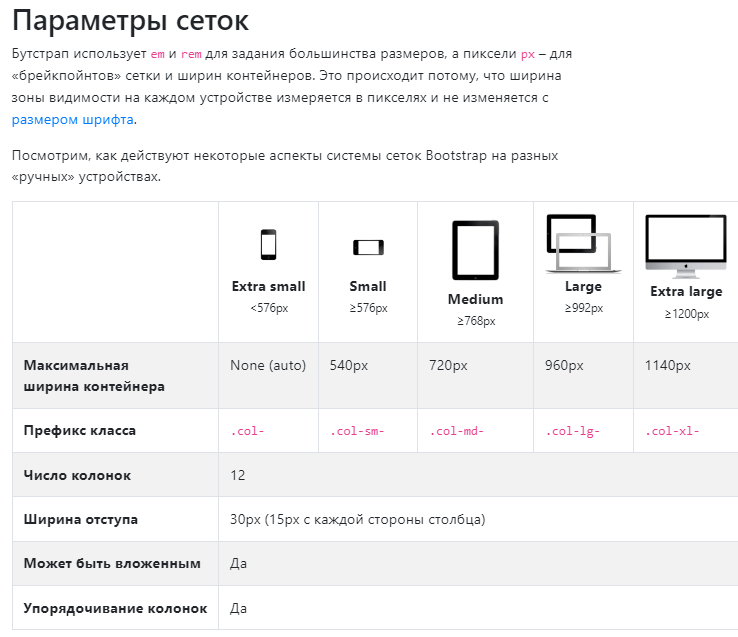
**.container** для фиксированной ширины

**.container-fluid** для 100% ширины

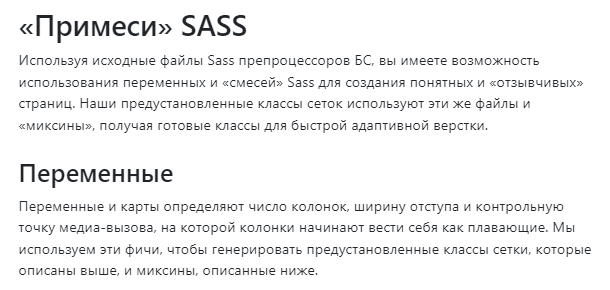
**Содержимое должно быть расположено в колонках, и только колонки могут быть расположены в рядах.**

**Используйте классы колонок со специальными контрольными точками (например, .col-sm-6) для легкого расположения колонок без использования явно обозначенных номеров классов.**

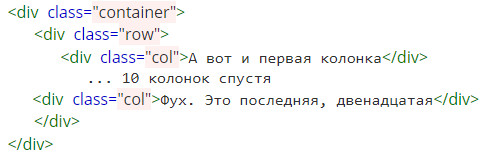
****

****

Короче. Дохера документации. Курс оплачен. Учите. А не ссылки кидайте.

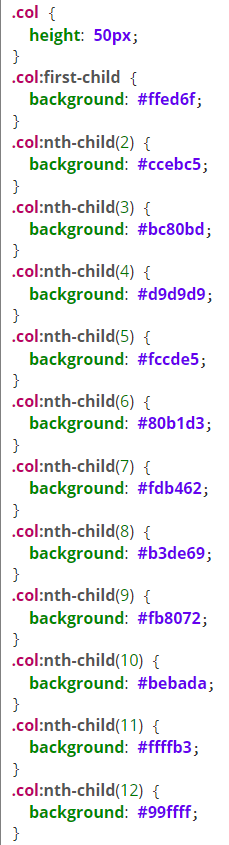


Вот как строится сетка:

****

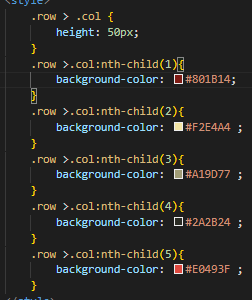
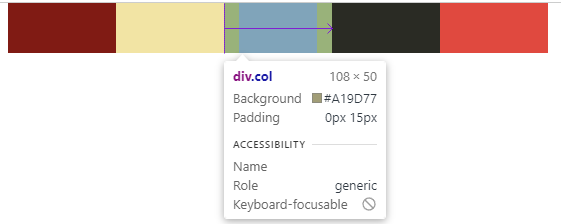
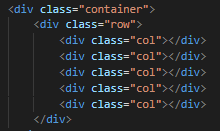
И что, нужно прямо все колонки прописать так? А не.. Ряд может содержать от одной, до 12 колонок.

Допустим дадим колонкам разные цвета. И будет

ЛГБТ со всеми 12 гендерами крч.

Попробую сам.



Нормально. А если вместо **container** использовать **container-fluid,** то сетка будет растянута на всю ширину экрана

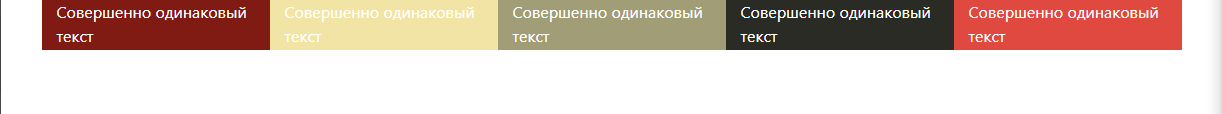


А если вставить текст и взглянуть на адаптивность?

25%



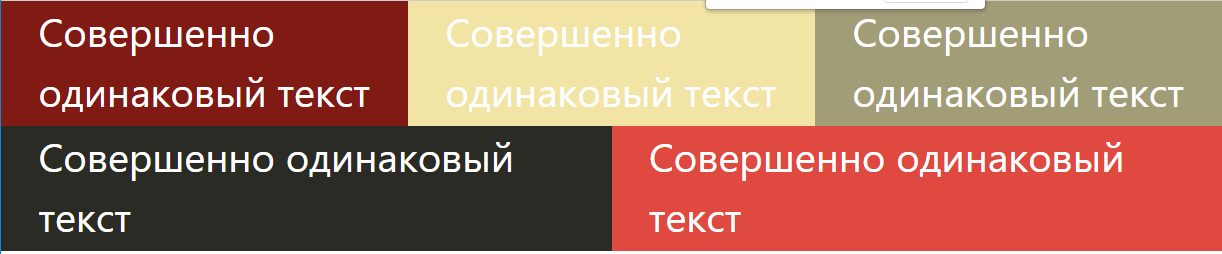
100%



175%



200%

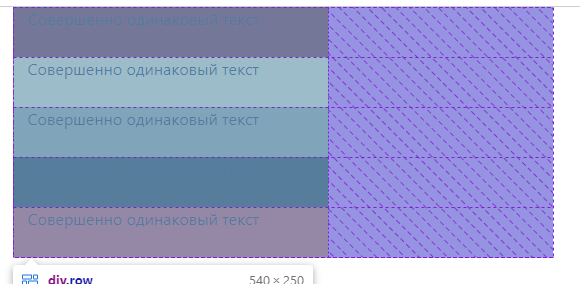
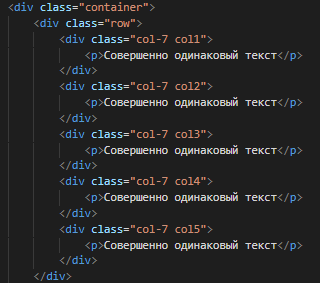


Ну в целом не плохо. Но не идеально. Где-то слово текст съелось.

Как происходит управление размерами колонок? Если не давать размеры, то они будут автоматическими. Нужно больше гибкости. К примеру, чтобы шапка занимала всю ширину, но блоки ниже нет.

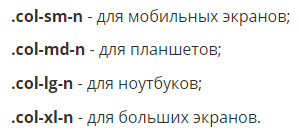
Для задания ширины используются специальные классы вида **.col-x-n,** где вместо **x** ставится буквенное **обозначение размера экрана.** Вместо **n –** число от 1 до 12. Обозначающие сколько место займет **данная колонка.** Если опустить **x.** То размер будет применяться на всех экранах.

Если **n** будет 6. То эта колонка будет шириной в половину от всего ряда. Хм… А если все сделать 7?

Как я и думал.

**Обозначение размера экрана такие:**

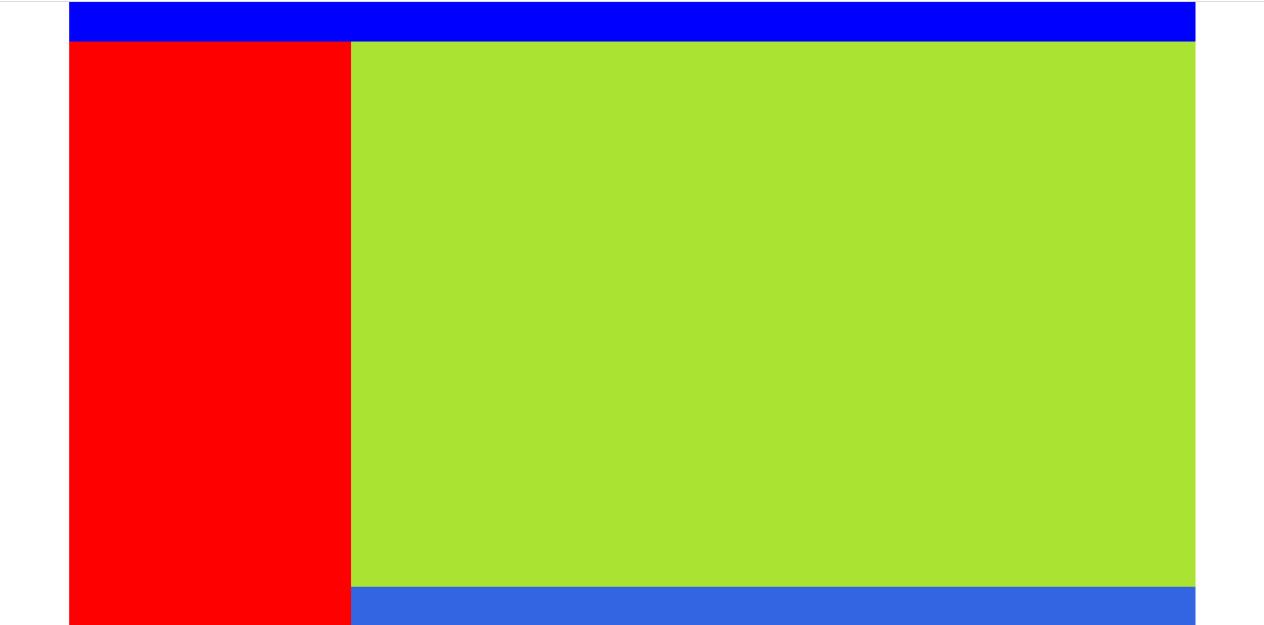


Не обязательно указывать все экраны. У ноутбуков и больших например они могут совпадать.

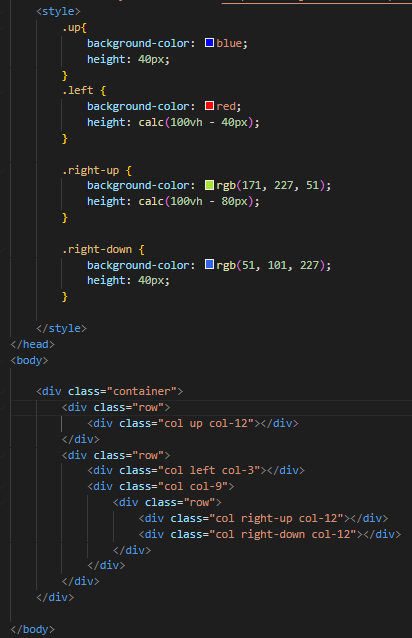
Если у сайта есть 2 раскладки, одна - для всех десктопных устройств, вторая - для мобильных и планшетов, будет достаточно поставить 2 класса: **.col-n** и **.col-lg-n.** Первый класс будет действовать по умолчанию на всех устройствах, где разрешение меньше, чем у ноутбука (по стандартам **Bootstrap** это означает меньше 960px). Как только разрешение будет больше 960px, начнет действовать второй класс.



Пришлось чуть повозится. Но…

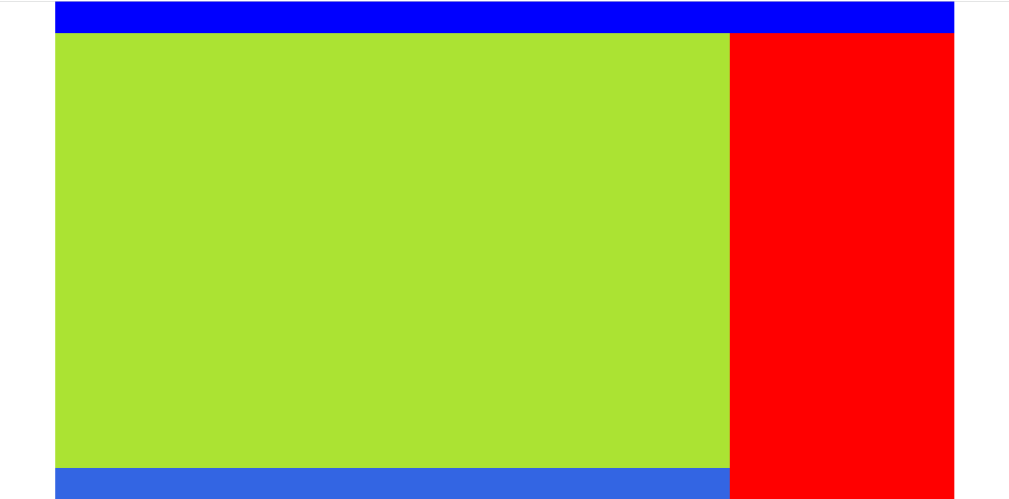


Да… заметно, что шапка и блоки должны быть выше. Но такие параметры были в условии.



Важно отметить. Все **row** имеют в свойства **display:flex;** Так что если нужно как-то изменить позиционирование строки, всегда пожалуйста. Например добавить:

**flex-direction: row-reverse;**

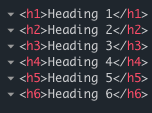
****

**Вот эту статью прочесть:**

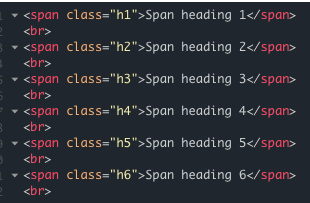
[**https://medium.com/@stasonmars/%D1%81%D0%B5%D1%82%D0%BA%D0%B0-%D0%B2-bootstrap-4-%D0%BF%D0%BE%D0%B4%D1%80%D0%BE%D0%B1%D0%BD%D0%BE%D0%B5-%D1%80%D1%83%D0%BA%D0%BE%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE-bb20ce8a14ad**](https://medium.com/@stasonmars/%D1%81%D0%B5%D1%82%D0%BA%D0%B0-%D0%B2-bootstrap-4-%D0%BF%D0%BE%D0%B4%D1%80%D0%BE%D0%B1%D0%BD%D0%BE%D0%B5-%D1%80%D1%83%D0%BA%D0%BE%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE-bb20ce8a14ad)

**A5.5 Типографика в Bootstrap**

Эм… не по каким-то причинам не удается использовать теги заголовков…



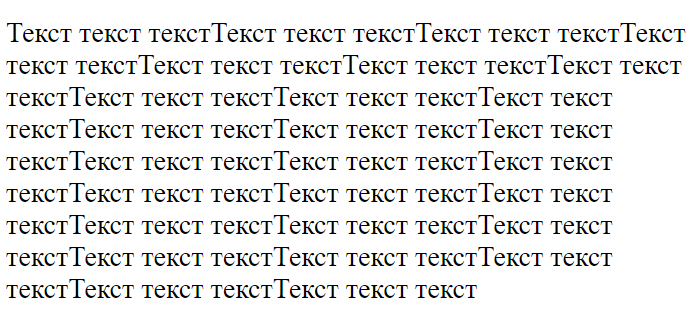
То их можно заменить так:



Эм.. Точно обойдусь без этого.

Абзацы:

Без bootstrap



С bootstrap, точнее тут увеличился отступ снизу с помощью класса **lead.**



**Еще пару удобных тегов.**

**<mark>/** выделяет текст заливкой желтого цвета



**<del>/** и **<s>/** зачеркивают текст

Только **del** принято использовать, чтобы выделить текст, который был удален в старой версии, а

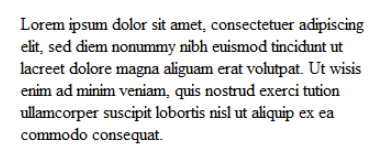
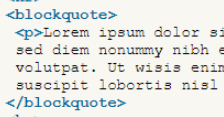
**S** не несёт семантический нагрузки и может использоваться без причин, для дизайна.

****

**<ins>/** подчеркивает текст снизу.



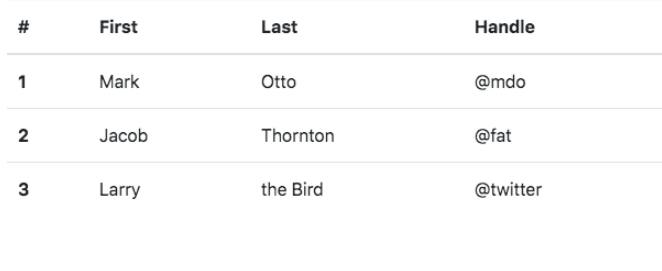
**<blockquote>/** обернуть параграф, выделив его, как цитату



**Классы:**

**list-unstyled –**ненумерованный список

**table –** стилизация таблицы



**table-striped –** таблица с разделенными ячейками.



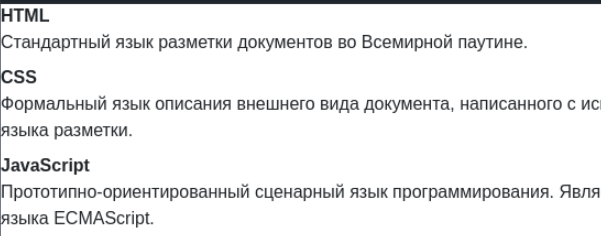
**Сам код оформлен с доп. атрибутами.**

****

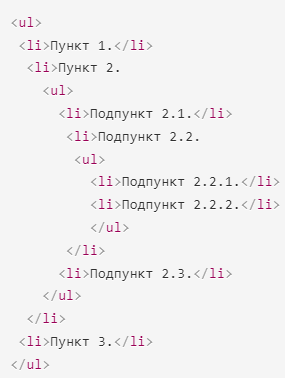
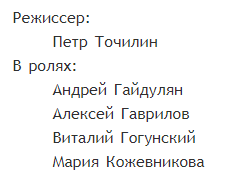
Для списков существуют следующие теги:

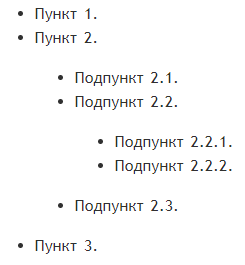
****

**Dl –** список описание. **dt** – для заголовка, **dd** – для описания. К **dl** можно добавить класс: **dl-horizontal**

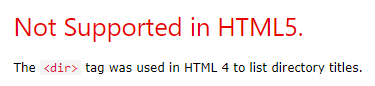
****

Без bootstrap Вложенные списки



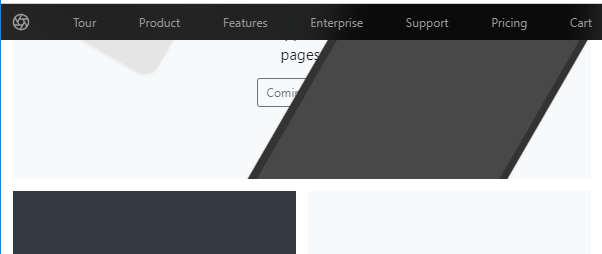


**DIR –** больше не нужен. Его заменяет **ul**

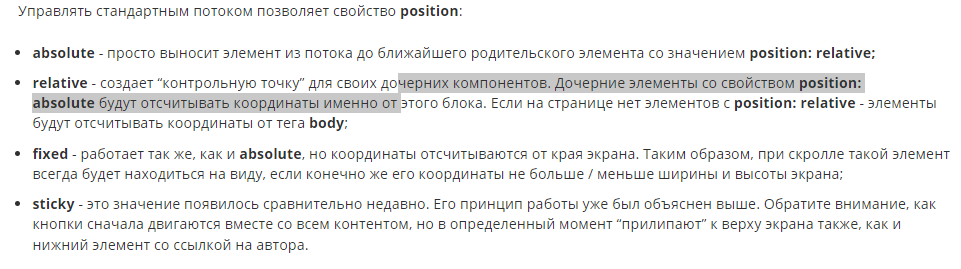


**A5.6 Разбор макета Bootstrap**

В первых шапка имеет класс **sticky-top.** Благодаря нему шапка всегда будет наверху экрана. Но этот класс отрабатывает, когда область просмотра достигает определенной позиции. А если объект с этим классом уже наверху, то сразу.



Напоминание о потоке.



Классы по работе с **flex-box.**

**d-flex –** активирует flexBox

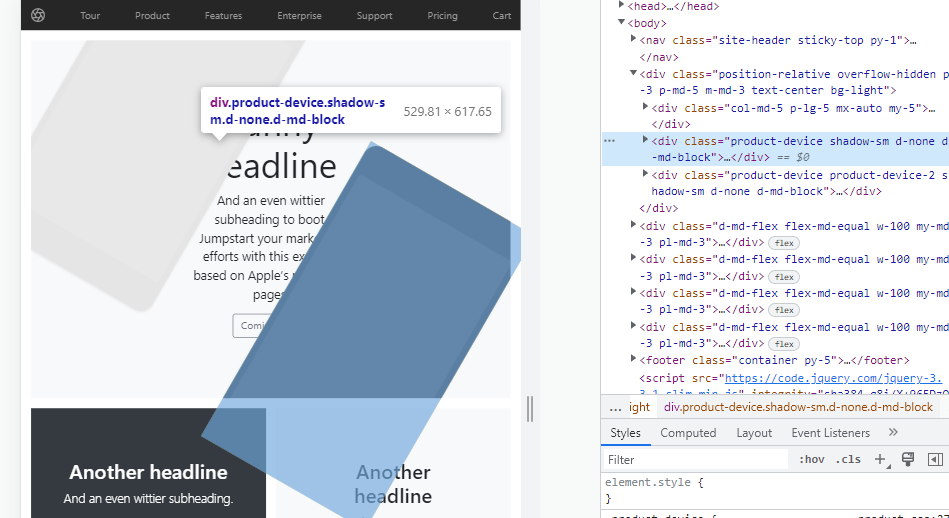
**flex-column –** располагает элементы в колонку, обычно для маленьких экранов

**flex-md-row –** располагает элементы в ряд

**justify-content-between –** равномерное заполнение элементов блока на всю ширину. Крайние элементы будут прилегать к краям контейнера.

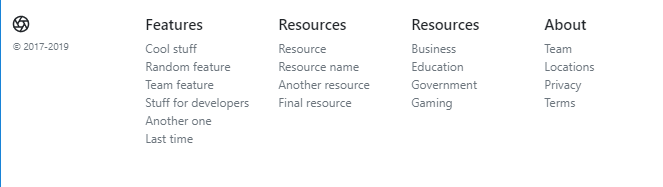
Далее на страницеесть два таких стилизованных блока в виде телефонов. У них есть класс **d-none**

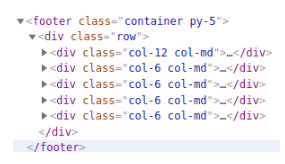
Благодаря нему блоки будут скрываться на мобильных устройствах или маленькой ширине.



**flex-md-equal –** сделает все дочерние элементы равными по ширине.

Самый низ сайта сделан с помощью сетки.





Блок будет занимать всю ширину маленьких экранов. И остальные половину.

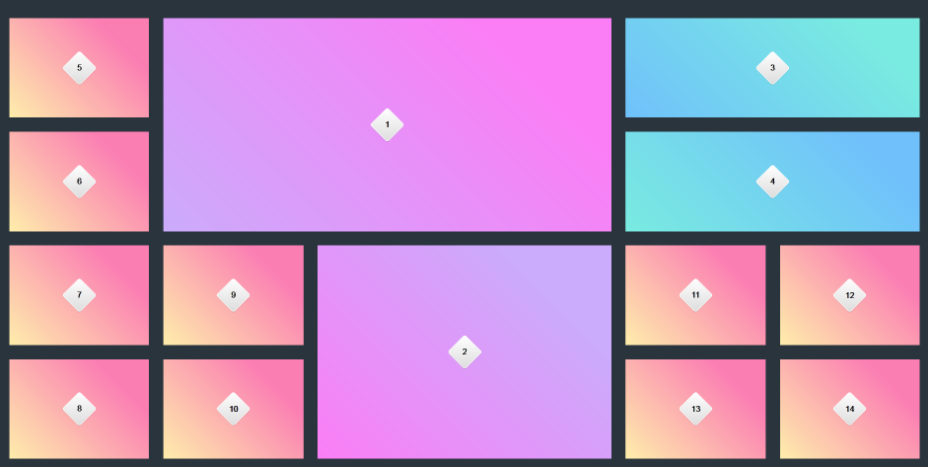
Для остальных размеров экрана каждый блок занимает 1/12 размера контейнера **(.col-md).**

**A5.7 CSS grid: создание сетки без bootstrap**

Основы **grid-**раскладки.

Во первых, зачем он нужен есть если есть **FLEX?** (не знаю, надеюсь он удобнее и лучше)

**FLEX** одномерная раскладка. Элементы выстраиваются вдоль одной оси. Их конечно можно выравнивать по дополнительной оси, но для этого нужно больше флекс оберток. Представь как ты будет раскидывать такой дизайн:



<https://codepen.io/primalivet/pen/ryjKmV>

Номера на блоках обозначают их порядок в html коде.

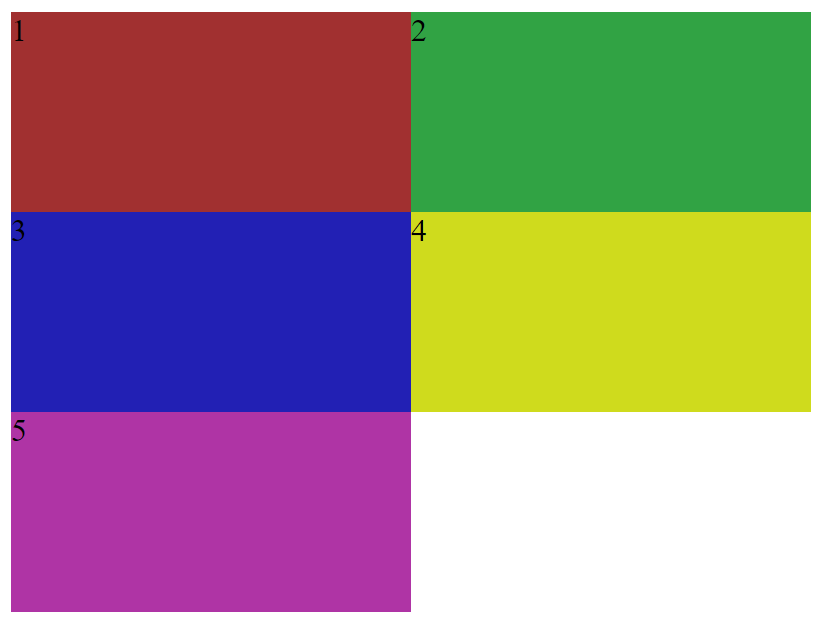
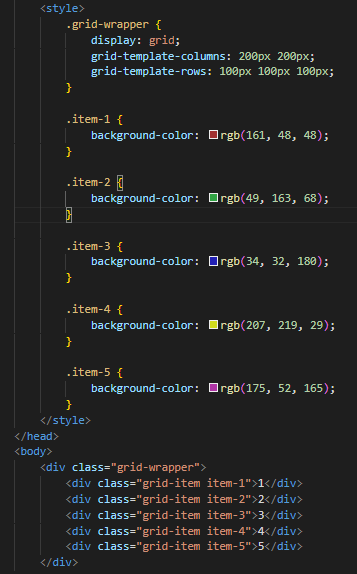
Дизайнеры легко могут нарисовать такой макет в редакторе. Причем у них будет такой инструмент как сетка. В Фигме есть, например. Вертикальные столбцы 4, 8 или 12. Так вот, чтобы перенести подобную дизайнерскую сетку и был придуман грид.

Грид не заменяет **flexBox,** а дополняет(но если не будет проблем, перейду на него полностью)

Грид позволяет заменить ту же верстку сеток из **bootstrap.**

Итак. Как это работает?

Для начала вот несколько блоков



Без каких либо библиотек и чего-то **grid** работает.

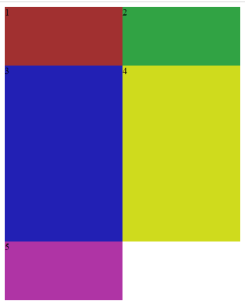
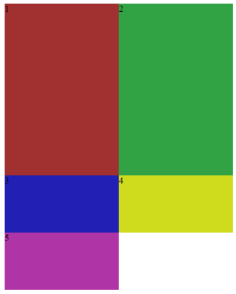
**Display: grid;** понятно, но задание высобы столбцов и колонок еще нет.



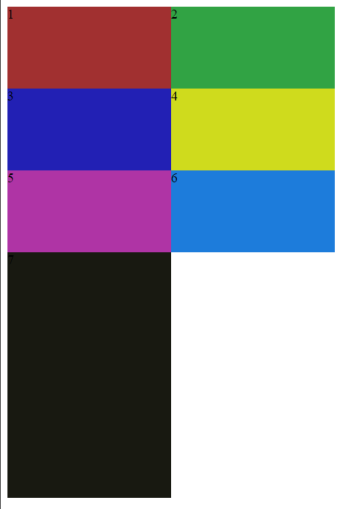
Первый пункт расширяет нечетные столбцы а второй пункт четные.



Хм… Не понятно. А если я добавлю еще пару столбцов, то это уже будет 4 атрибут?

****

****

**Странная штука.** Ладно продолжим. Получается каждый

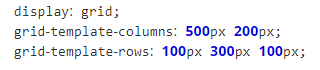


**Хм…** То есть я могу создать условно 4 колонки С определенной шириной?



Тут 4 колонки шириной 200 200 300 и 400 пикселей. И два ряда высотой 400 и 100 пикселей.





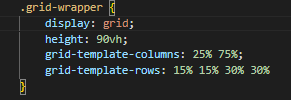


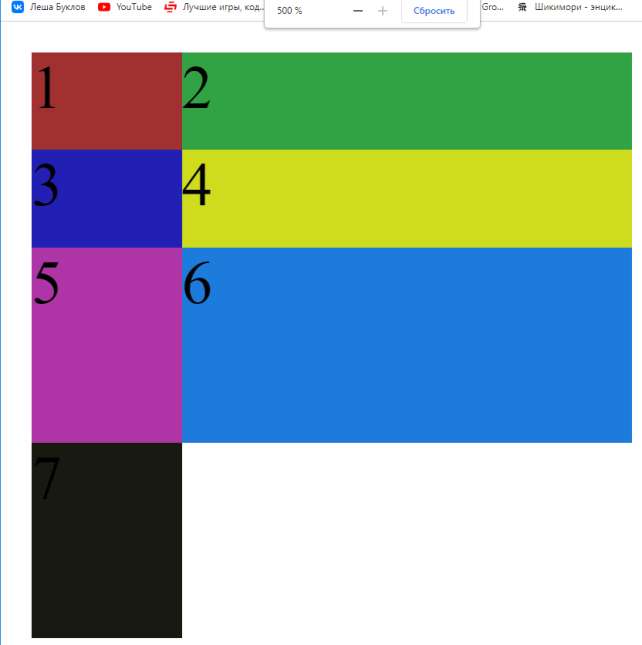
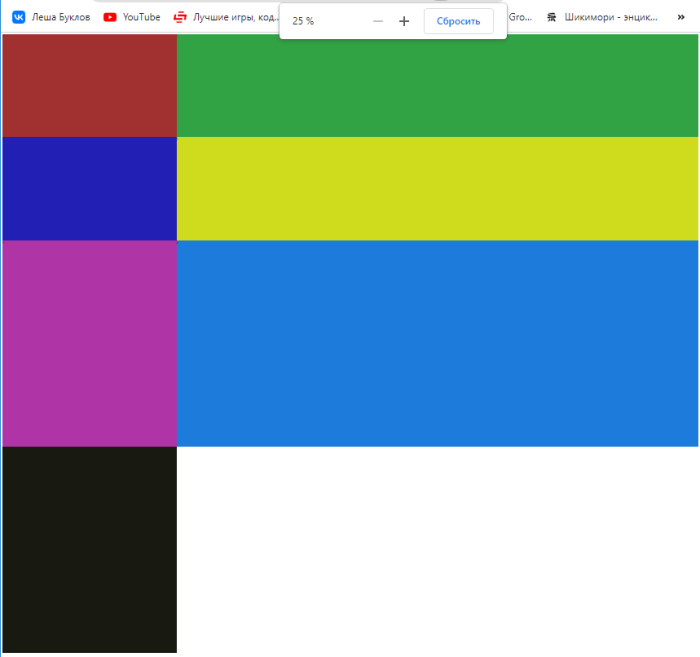
Понятно. Если солонки одинаковые, то можно использовать **css** функцию **repeat()**

Первый аргумент количество повторов, второй аргумент что повторять.

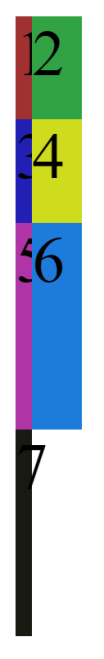


**Помимо пикселей проценты от родителя и от экрана хорошо работают.**

****

****

Но адаптивным я бы это не назвал.

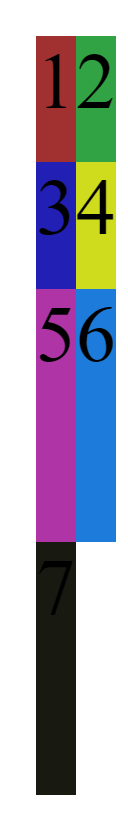
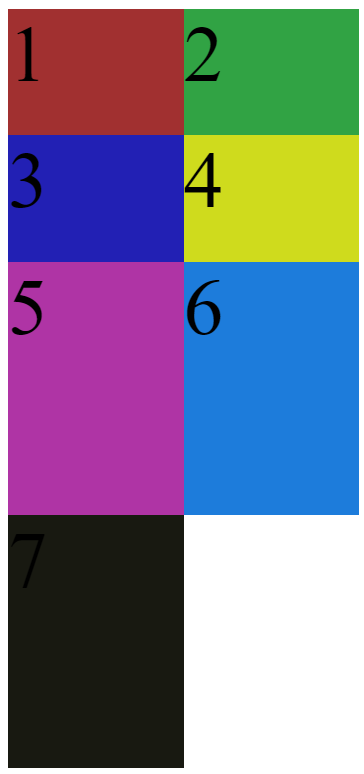
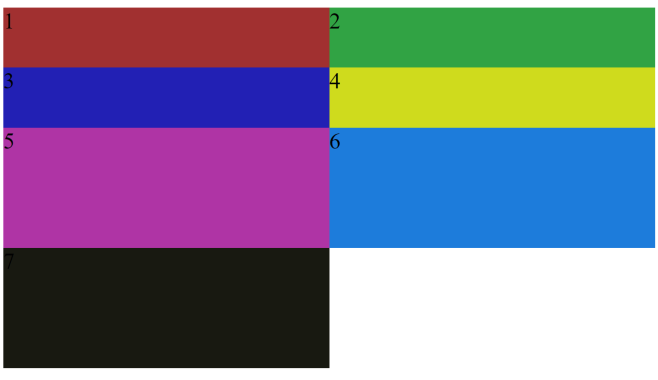
****

**A5.8 Адаптивные гриды**

С адаптивностью у меня всегда было не очень. Посмотрим, что предложит **GRID.**

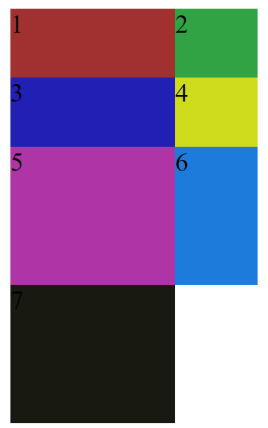
Специально для **grid** создана единица измерения – **fr.** Она указывает размер относительно других колонок или рядов.

**grid-template-columns: 1fr 1fr; -** создаст 2 колонки, которые займут все доступное место поровну.

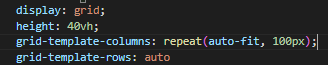
  

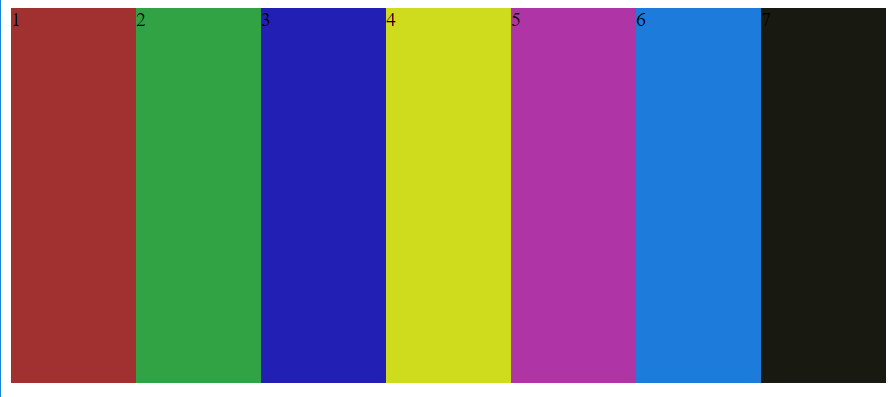
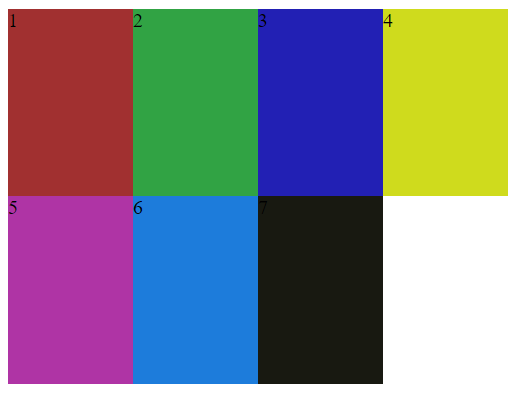
А вот тут хорошо. Блоки не становятся меньше их содержимого.

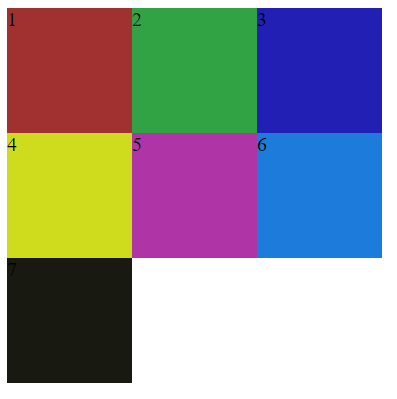
Если же задать одному из столбцов 2**fr,** а другому 1. То один всегда будет в 2 раза больше первого.



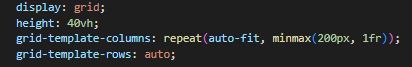
Но резиновая это все еще не адаптивная. Нужно чтобы блоки переносились. Для этого есть ключевое слово **auto-fit** вместо количества ячеек.



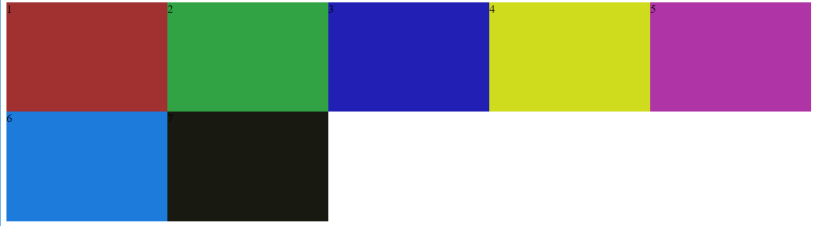
 

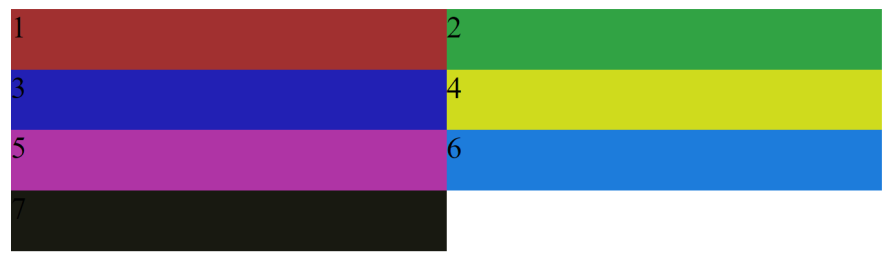


Но чтобы содержимое было адаптивным и при этом занимало всю ширину можно написать такую конструкцию.



Функция **minmax()** принимает в себя минимальную и максимальную ширину.

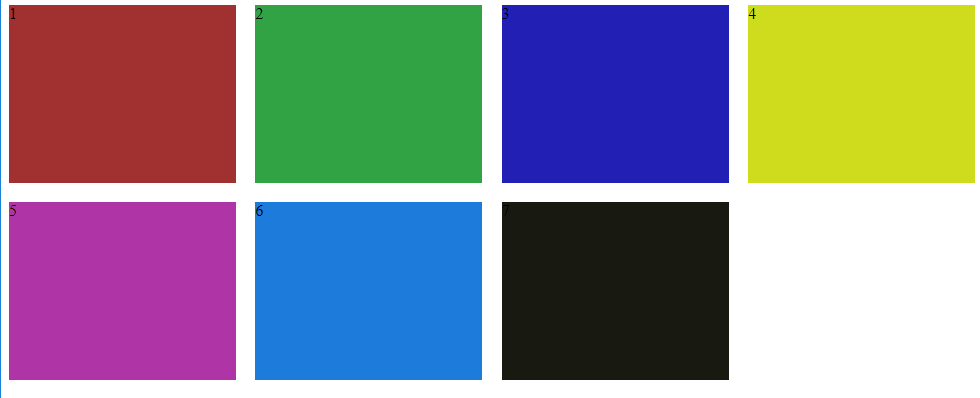






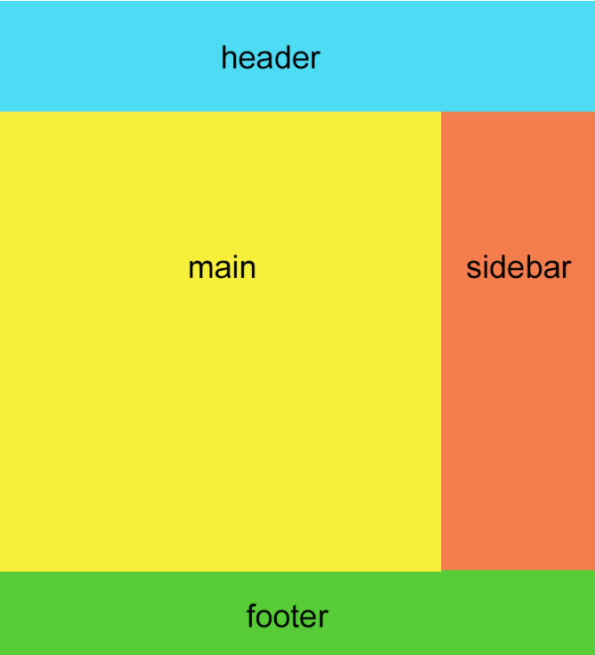
И блокам не обязательно быть в плотную друг к другу. Им можно задать отступы. Для этого есть свойство, оно принимает проценты, пиксели и т.д:

**grid-gap: 2vw;**

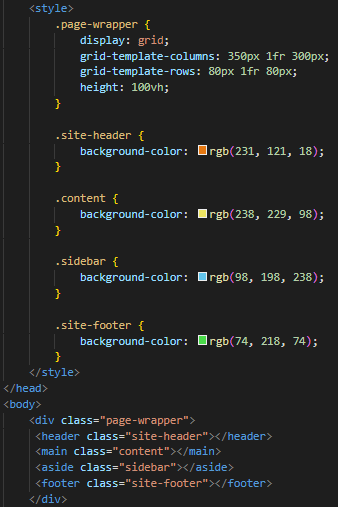


**A5.9 Раскладка страницы**

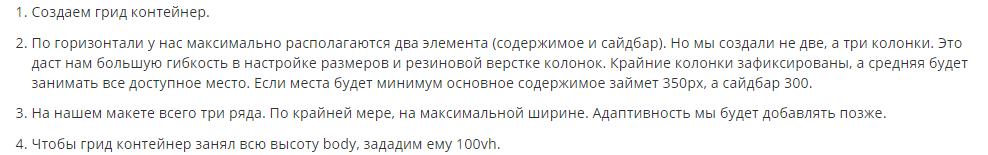
Построение стандартного расположения элементов в дизайне.



Вот с таким кодом пока получается так:

Как его превратить в дизайн?



Второй и третий пункт пока не понятны.

**Одна из особенностей гридов — вы можете располагать содержимое в тех частях сетки, где захотите. Есть множество способов это сделать.**

Свойства:

**column-start:**

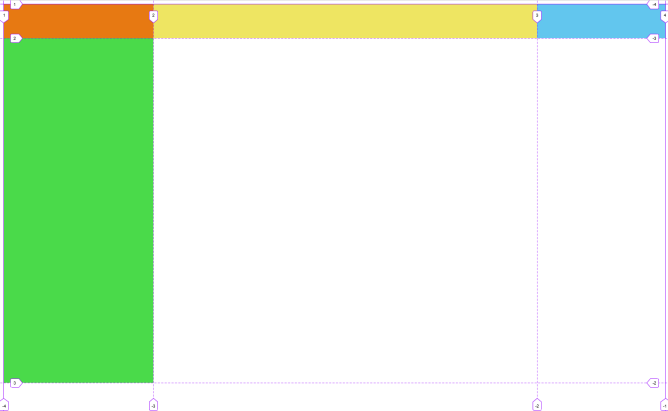
**column-end:**

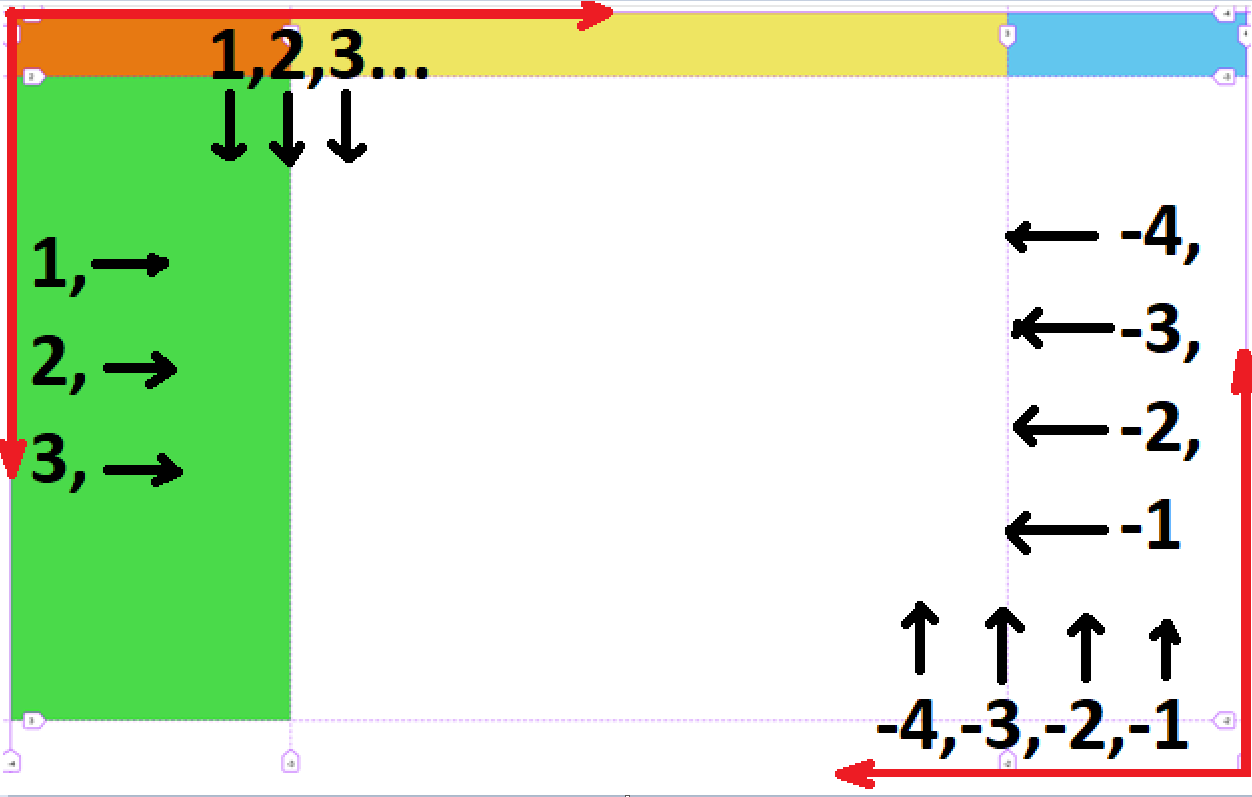
**row-start:**

**row-end:**

Они позволяют указать начало, конец содержимого. Границы содержимого отмечаются грид линиями. Элемент и его содержимое расположатся между этими линиями. Пространство между линиями называют грид треком (**grid track**).

В **Mozilla** их можно даже включить, нажав на значок сетки.

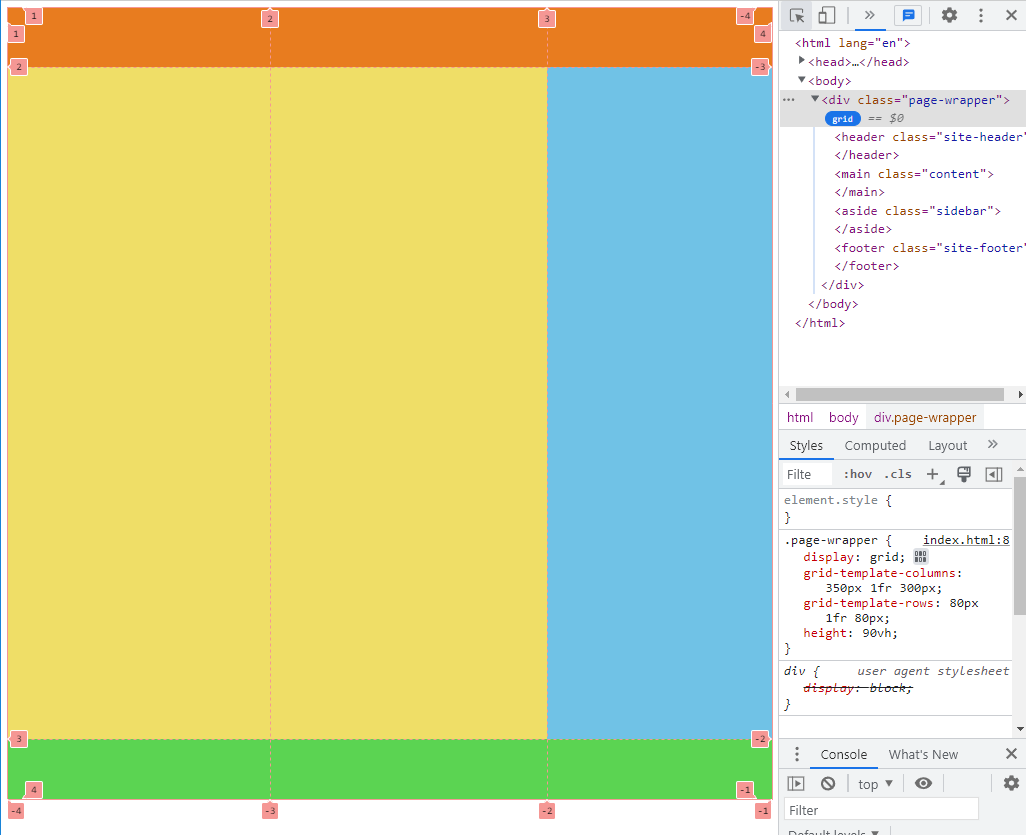


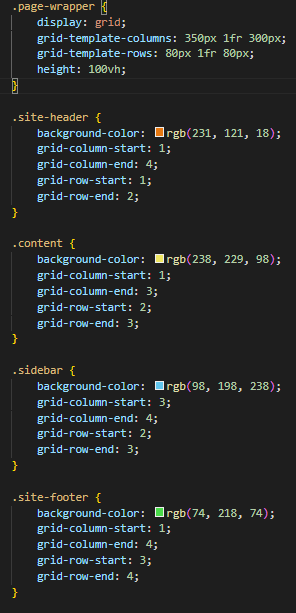
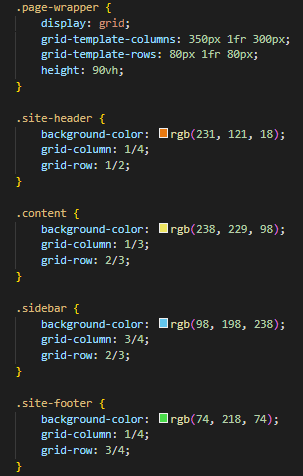


**Вот как я вижу это построение.**

Линий всегда на один больше чем строк и столбцов, так как в одном столбце 2 линии.

Попробую сам. О в Хроме тоже есть своя кнопка.



Но как я вижу, можно упростить задачу без использования **слов start/end.**

**Отлично получилось.**

**A5.10 Грид зоны**

Второй способ расположения.

Классам мы даем зону:

**grid-area: header;**

И эти зоны располагаем по колонкам и столбцам.

**grid-template-areas: “header header header” “main main sidebar”**

Я попробовал. Да, работает достаточно понятно.



Имена могут быть любыми, главное не допускать ошибки когда саму зону формируешь. Если хотя бы одно имя не будет найдено, не прорисуется ничего.

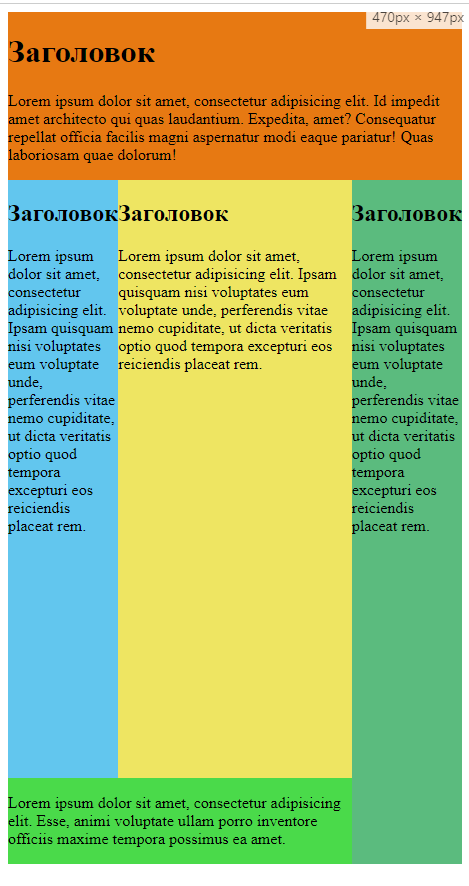
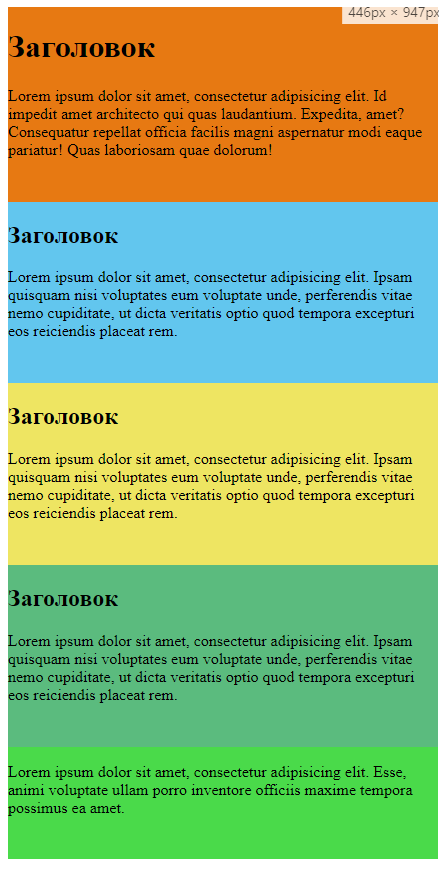


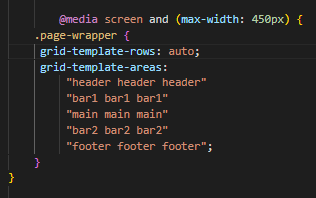
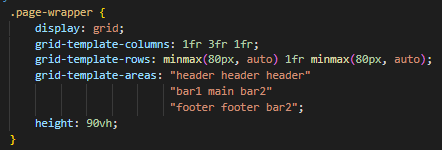
Расположить можно много как.



Главное, чтобы были заполнены все зоны и не было диагональных зон. Иначе ничего не прорисуется.

А как только что-то не будет помещать, можно будет простым медиазапросом изменить раскладку в этом классе.

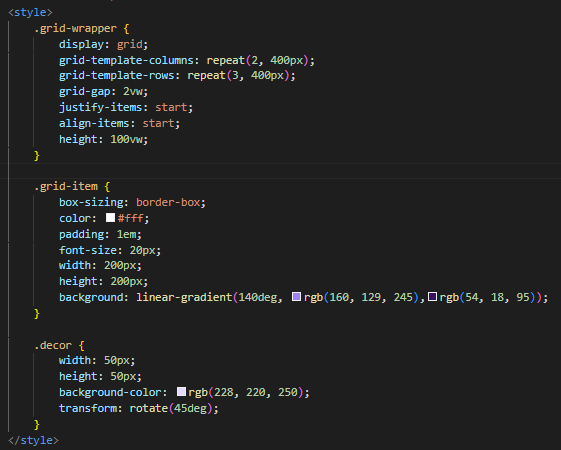
 



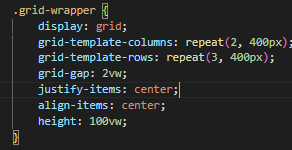
**A5.11 Выравнивание в grid**

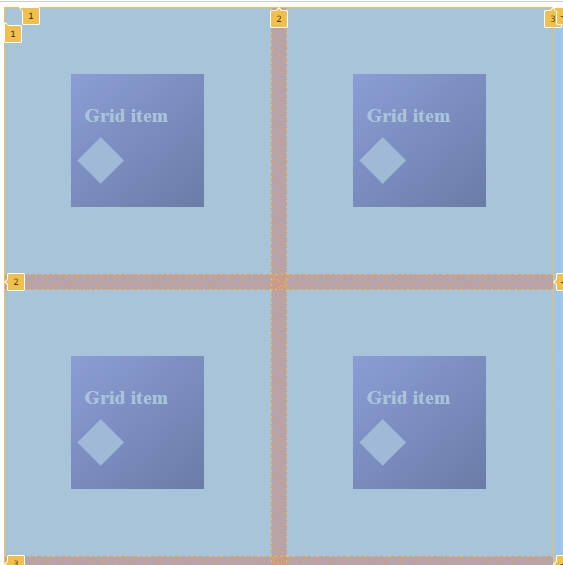
Теперь речь о выравнивании самой грид и ее содержимого. Вот взял готовый код.

****

** **

Симпатичная страничка. О знакомые слова. Сейчас их все в центр кину.





Да понятно. Осталось только записать.

**Justify-items –** выравнивание по горизонтали

**Align-items –** выравнивание по вертикали.

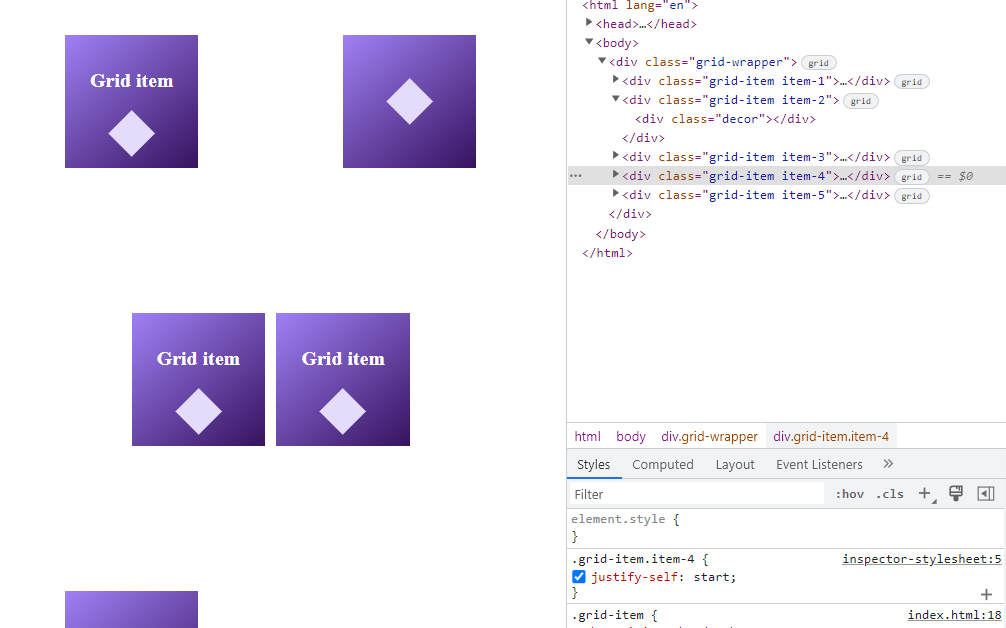
Это же можно проворачивать и с содержимым.



С помощью этих команд можно выровнять отдельные блоки, независимо от других. Эти свойства прописываются непосредственно внутри блоков.

**align-self:**

**justify-self:**



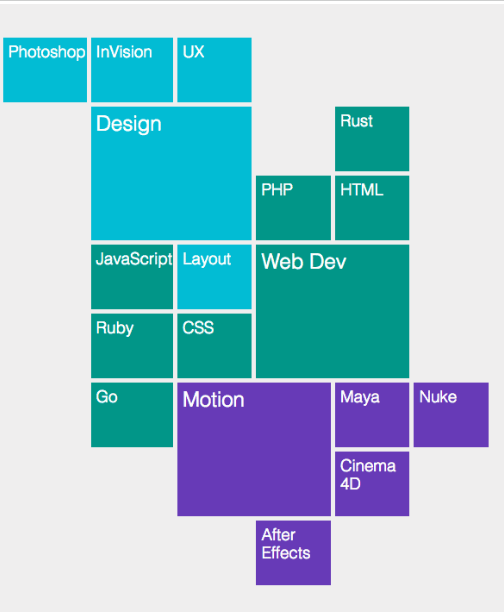
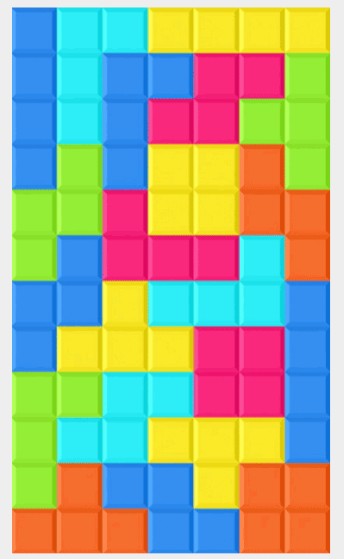
**align-content:**

**justify-content:**

Выравнивают саму сетку, если величина сетки меньше контейнера.

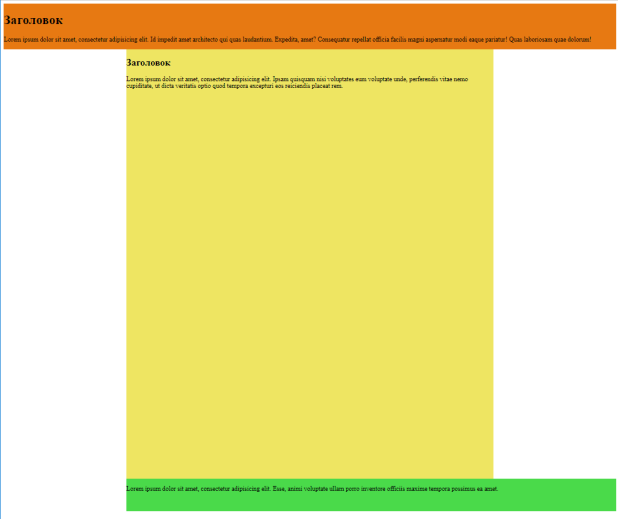
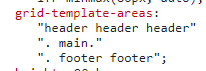
Подытожим. **Что нельзя делать в гридах?**

**Нельзя Льзя**



Правда для второй картинки наверно придется использовать какие-то пустые контейнеры или как-то так. Или на каждый прописывать **start end.**

**Все ок. Можно использовать точки для пустых мест**



А так много чего еще на практике надо попробовать. Без этого никак.