



КАК СТАТЬ АВТОРОМ



Выгорание в IT До покраснения дисплея тестим и бенчим...



0

Рейтинг

Маклауд

Облачные серверы на базе AMD EPYC

[Подписаться](#)

aio350

24 июн 2021 в 19:00

Полное визуальное руководство/шпаргалка по CSS Grid

7 мин

316K

Блог компании Маклауд, Веб-разработка*, CSS*, Программирование*

[Перевод](#)

Автор оригинала: Joy Shaheb

Grid tutorial with

CheatSheet



Сегодня мы с вами рассмотрим свойства CSS Grid (далее также – Грид), позволяющие создавать адаптивные или отзывчивые макеты веб-страниц. Я постараюсь кратко, но полно объяснить, как работает каждое свойство.

Что такое CSS Grid ?

What is Grid?



Грид – это макет для сайта (его схема, проект).

Грид-модель позволяет размещать контент сайта (располагать его определенным образом, позиционировать). Она позволяет создавать структуры, необходимые для обеспечения отзывчивости сайтов на различных устройствах. Это означает, что сайт будет одинаково хорошо смотреться на компьютере, телефоне и планшете.

Вот простой пример макета сайта, созданного с помощью Грида.

Компьютер

The screenshot displays a website layout using CSS Grid. The header features a yellow speech bubble-like element containing the logo 'Toy.io'. To the right of the logo is a navigation bar with 'Home' underlined in orange, 'About', and 'Store'. Below the header is a decorative illustration of a white and black cat playing with a ball of purple yarn. The main content area contains a large purple text block: 'Get Your Pet A Toy Today !'. At the bottom of this section is a red button with the text 'Buy Now !'. The footer contains social media icons for Twitter, Instagram, and Facebook.

Телефон



Toy.io



Get Your Pet
A Toy Today !



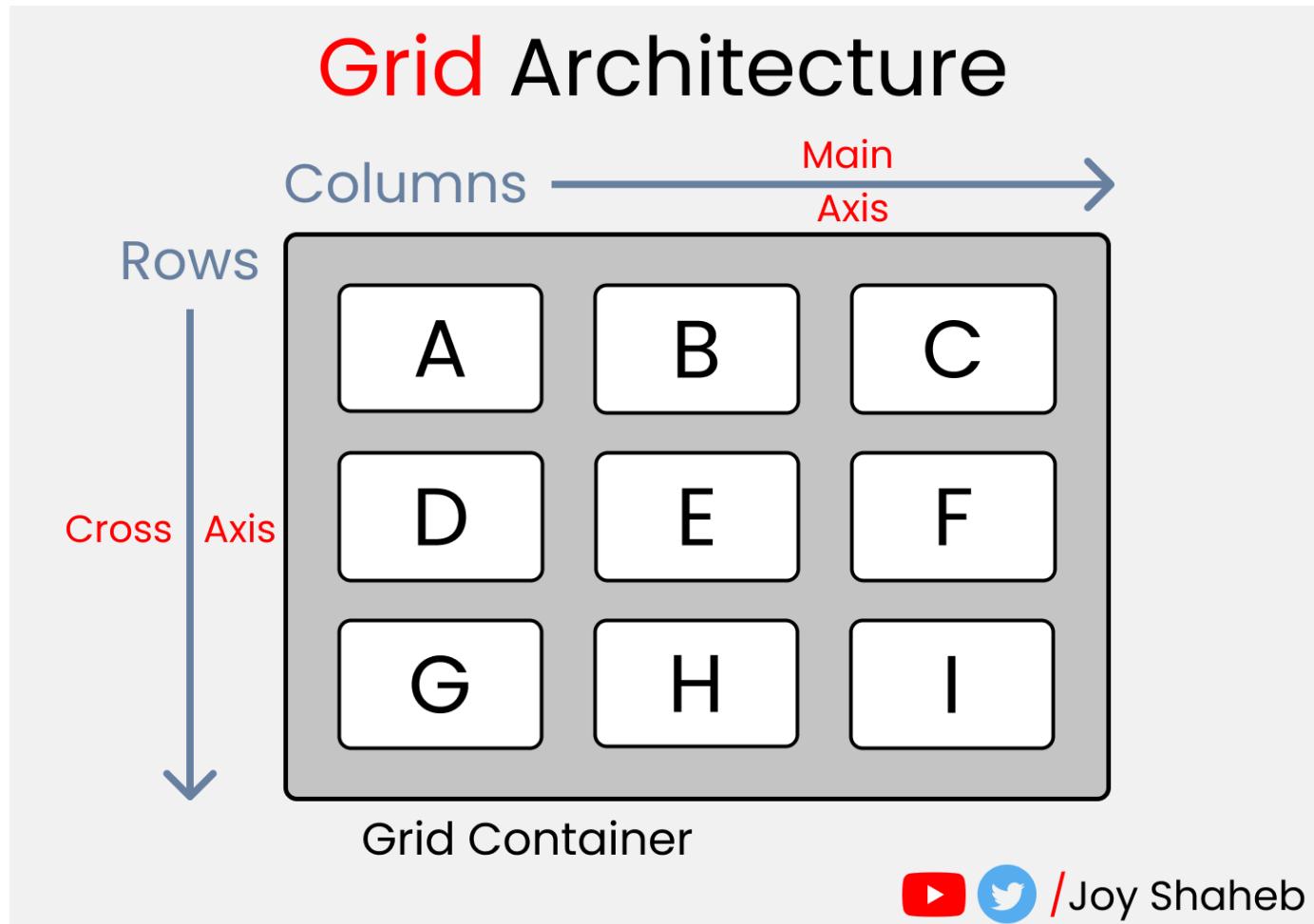
Buy Now !



Архитектура CSS Grid



Как же Грид работает? Элементы Грида (grid items) располагаются вдоль главной или основной (main) и поперечной (cross) оси (axis). При помощи различных свойств мы можем манипулировать элементами для создания макетов.



Помимо прочего, у нас имеется возможность объединять строки и колонки подобно тому, как мы это делаем в Excel, что предоставляет нам большую гибкость, чем Флекс (Flexbox).

К слову, если вас интересует Флекс, вот соответствующая статья.

Схема CSS Grid

The Grid Chart



Схема содержит все возможные свойства, предоставляемые Гридом. Эти свойства делятся на:

- родительские (свойства грид-контейнера) и
- дочерние (свойства грид-элементов)

Обратите внимание: красным цветом отмечены сокращения для свойств:

Grid Chapters

Parent Properties

- grid-template-columns
- grid-template-rows
- grid-template-areas
- **grid-template**
- column & row gaps
- **grid/grid-gap**
- justify-items || align-items
- justify-content || align-content
- **place-items || place-content**

Grid Chapters

Children Properties

- grid-column : start/end
- grid-row : start/end
- grid-area
- justify-self || align-self
- **place-self**

К концу настоящей статьи у вас будет полное понимание того, как работает каждое из них.

Настройка проекта

Grid Chapters

Children Properties

- grid-column : start/end
- grid-row : start/end
- grid-area
- justify-self || align-self
- place-self

Для данного проекта требуются начальные знания HTML , CSS и умение работать с VSCode (или другим редактором по вашему вкусу). Делаем следующее:

1. Создаем директорию для проекта, например, Project1 и открываем ее в редакторе (cd Project1 , code .)
2. Создаем файлы index.html и style.css
3. Устанавливаем в VSCode сервер для разработки (Live Server , расширение) и запускаем его

Или вы можете просто открыть Codepen (или любую другую песочницу) и начать писать код.

Все готово, можно приступать к делу.

Let's Code Together



HTML

Создаем 3 контейнера внутри body :

```
<div class="container">
  <div class="box-1"> A </div>
  <div class="box-2"> B </div>
  <div class="box-3"> C </div>
</div>
```

CSS

Шаг 1

Сбрасываем стили:

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
```

Шаг 2

Немного стилизуем body :

```
body {  
    font-family: sans-serif;  
    font-size: 40px;  
    width: 100%;  
    min-height: 100vh;  
}
```

Шаг 3

Стилизуем все контейнеры:

```
[class^="box-"] {  
    background-color: skyblue;  
    /* Размещаем контейнеры по центру */  
    display: grid;  
    place-items: center;  
}
```

Не волнуйтесь, мы рассмотрим каждое из указанных свойств Грида.

Шаг 4

Добавим небольшой отступ между контейнерами:

```
.container {  
    display: grid;  
    gap: 20px;  
}
```

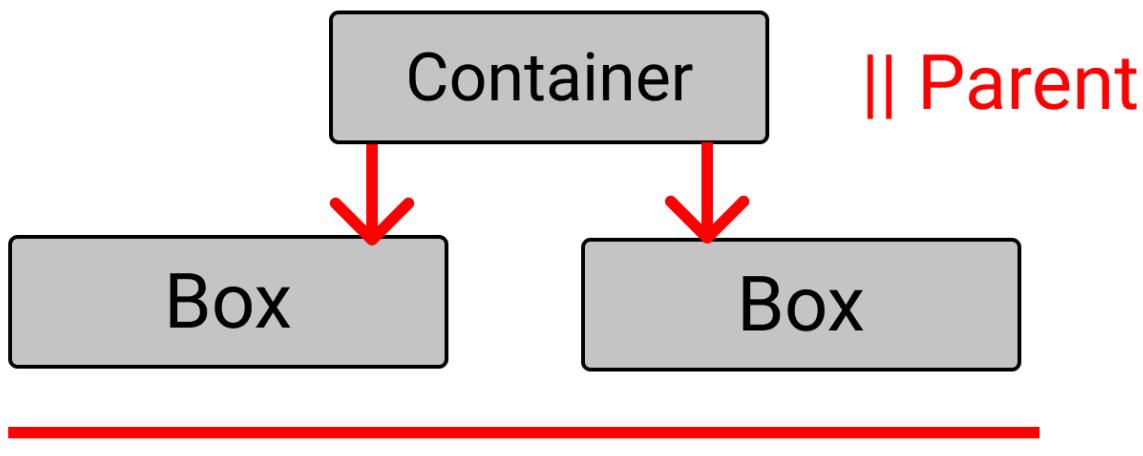
Погодите-ка

Wait a Minute !



Давайте разберемся с отношениями между родительским и дочерними элементами.

Flow



Свойства родительского элемента определяются в `.container`, а свойства дочерних элементов – в `.box-*`.

Свойства грид-контейнера

The Grid Parent Properties



Начнем со свойств родительского элемента.

grid-template-columns

Данное свойство используется для определения **количества и ширины** колонок. При этом, можно определять как свойства для каждой колонки в отдельности, так и устанавливать ширину всех колонок с помощью функции `repeat()`.

`grid-template-columns : 200px auto 100px;`



grid-template-columns : repeat(3, 1fr);



Добавим строку в `style.css`:

```
.container {
  display: grid;
  gap: 20px;

  /* ! */
  grid-template-columns: 200px auto 100px;
}
```

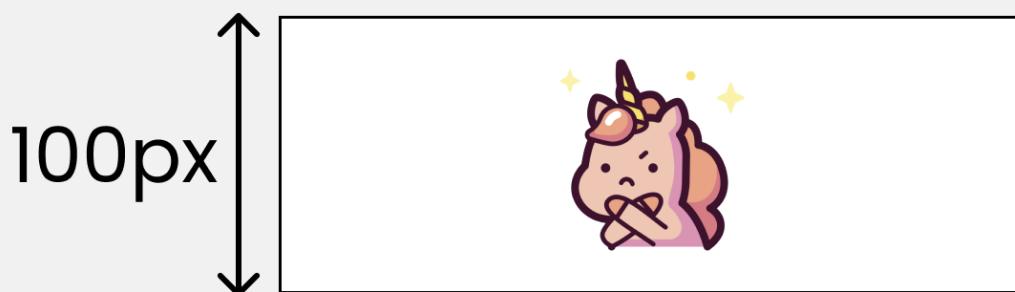
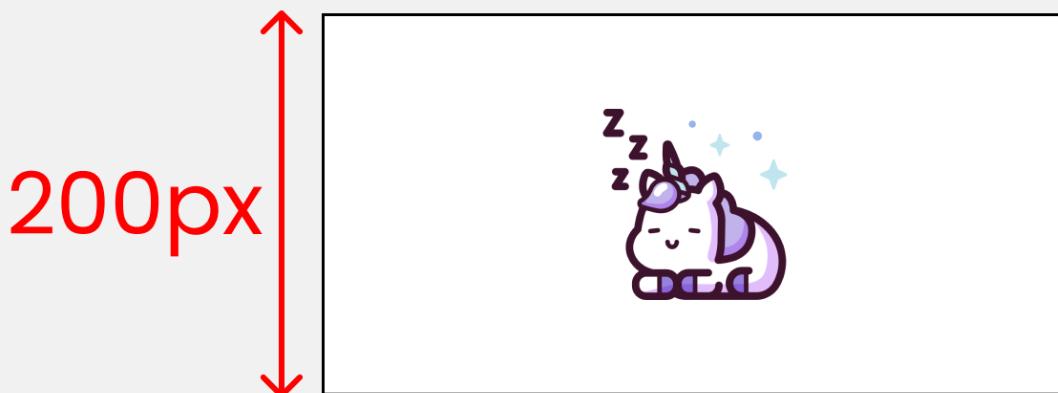
Обратите внимание:

- значения в пикселях будут точными. Ключевое слово `auto` означает заполнение элементом всего доступного пространства
- использование единицы `fr` (фракция) в `repeat()` означает, что все контейнеры будут иметь одинаковую ширину

grid-template-rows

Данное свойство используется для определения **количества и высоты** строк. При этом, можно определять как высоту каждой колонки в отдельности, так и устанавливать высоту всех строк с помощью функции `repeat()`.

grid-template-rows : **200px auto 100px**



grid-template-rows : repeat(3, 1fr);

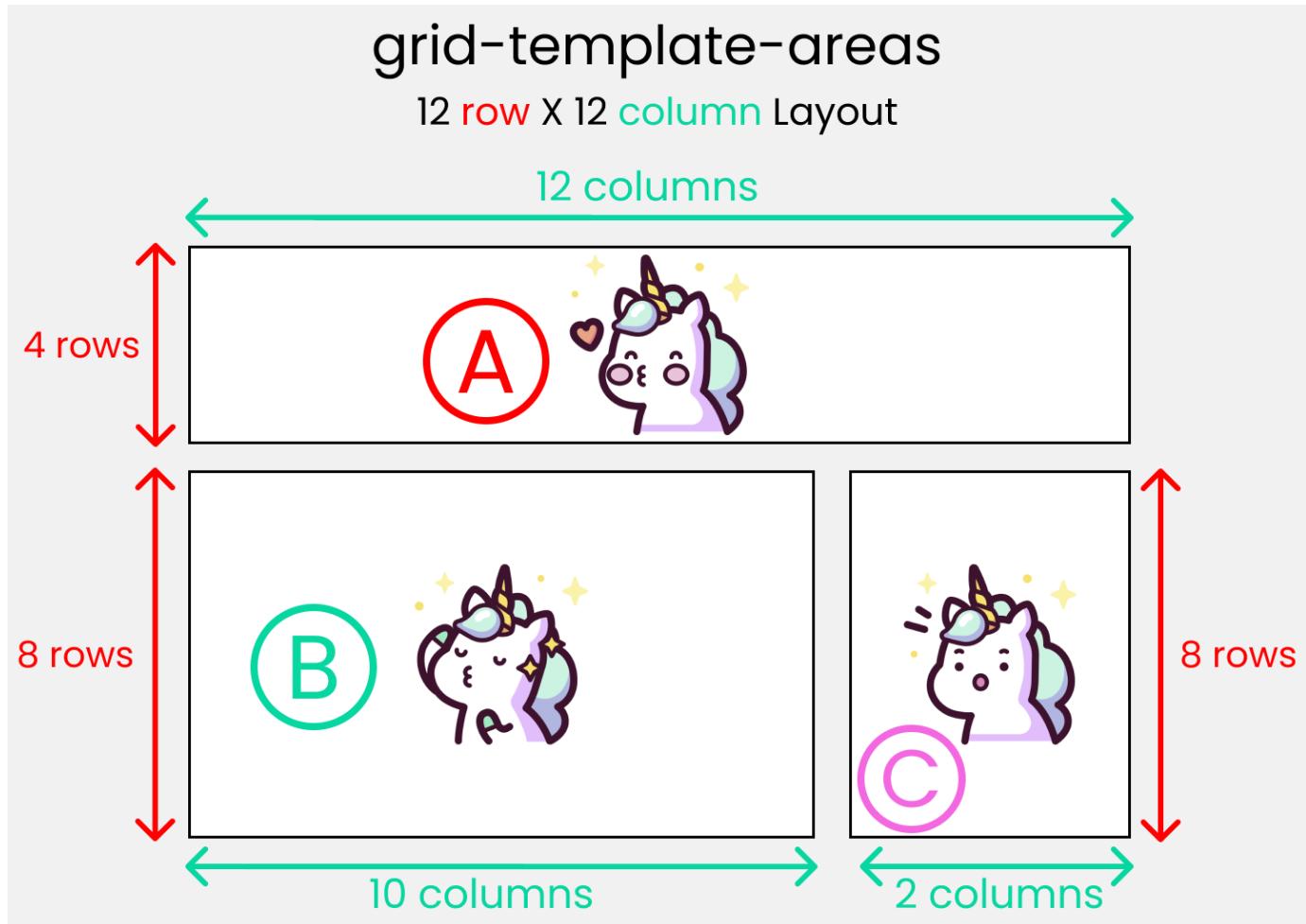


Изменим строку в `style.css`:

```
.container {  
    display: grid;  
    gap: 20px;  
    height: 100vh;  
  
    /* ! */  
    grid-template-rows: 200px auto 100px;  
}
```

grid-template-areas

Данное свойство используется для определения количества пространства, занимаемого ячейкой Грида (`grid cell`), в терминах колонок и строк, в родительском контейнере.



Это можно считать схемой макета:

The Code

`grid-template-areas :`

`"AAAAA AAAA AAAA"`

`"B BBB BBB BBB"`

`"B BBB BBB BBB";`

Для получения результата требуется не только родительское, но и хотя бы одно дочернее свойство:

- `grid-template-areas` : родительское свойство, создающее схему
- `grid-area` : дочернее свойство, которое использует схему

Создаем схему

```
.container {  
  display: grid;  
  gap: 20px;  
  height: 100vh;  
  
  /* ! */  
  grid-template-areas:  
    "A A A A A A A A"  
    "B B B B B B B C C"  
    "B B B B B B B C C";  
}
```

Применяем схему

```
.box-1 {  
  grid-area: A;  
}  
.box-2 {  
  grid-area: B;  
}  
.box-3 {  
  grid-area: C;  
}
```

Обратите внимание: мы вернемся к свойству `grid-area`, когда будем говорить о дочерних свойствах.

column-gap

Данное свойство используется для добавления отступа между **КОЛОНКАМИ**.

column-gap: 50px



Red **Dotted** lines are called -> grid lines

style.css :

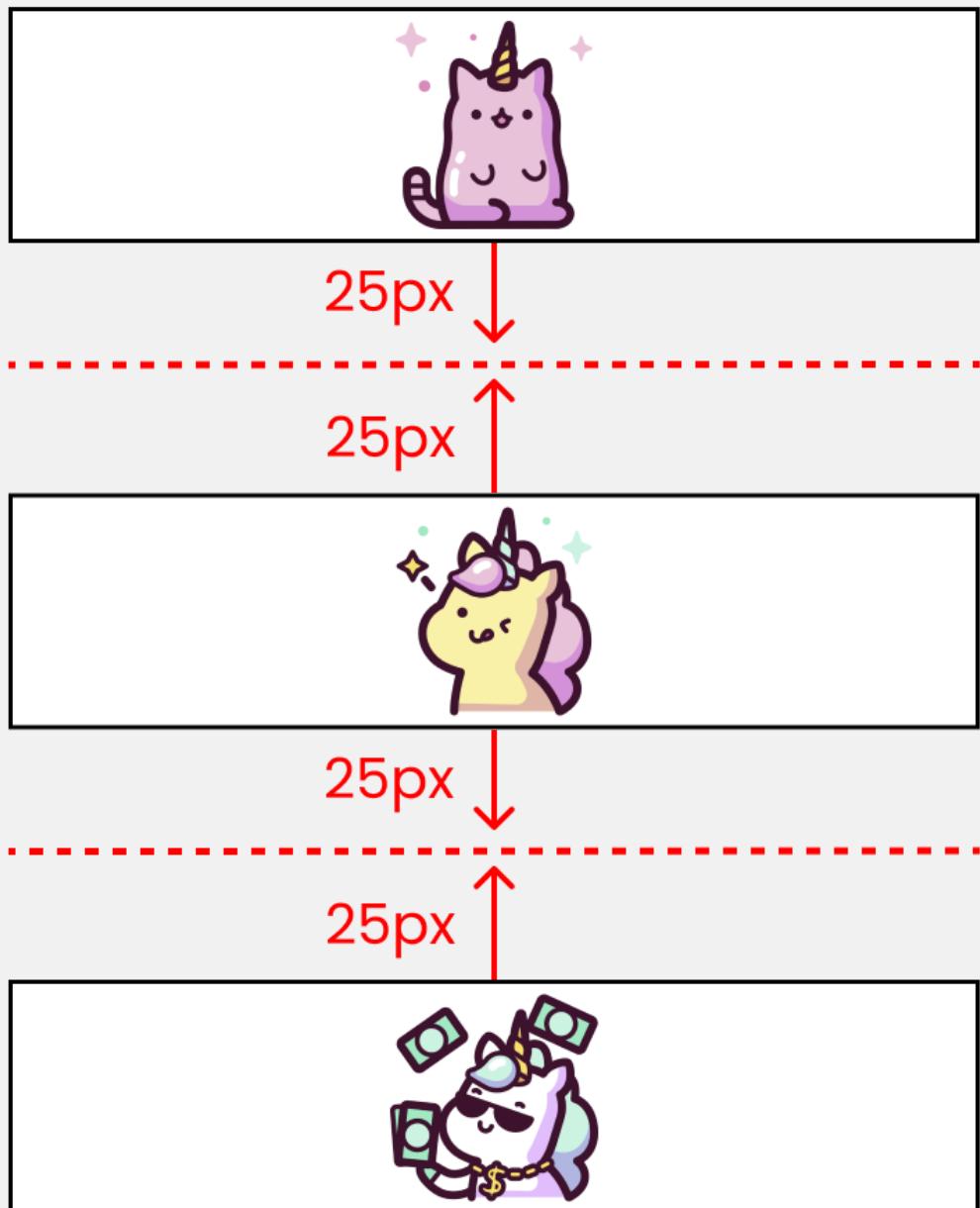
```
.container {  
  display: grid;  
  height: 100vh;  
  grid-template-columns: 100px 100px 100px;  
  
  /* ! */  
  column-gap: 50px;  
}
```

Обратите внимание: свойство column-gap используется совместно со свойством grid-template-columns .

row-gap

Данное свойство используется для добавления отступов между **строками**.

row-gap: 50px



Red Dotted lines are called -> grid lines

style.css :

```
.container {  
  display: grid;  
  height: 100vh;  
  grid-template-rows: 100px 100px 100px;
```

```
/* ! */  
row-gap: 50px;  
}
```

Обратите внимание: свойство `row-gap` используется совместно со свойством `grid-template-rows`.

justify-items

Данное свойство используется для позиционирования грид-элементов внутри грид-контейнера вдоль **главной оси**. Оно принимает **4** возможных значения:

justify-items -> X Axis

Start



end



justify-items -> X Axis

center



stretch



Добавим еще один контейнер в HTML :

```
<div class="container">
    <!-- Здесь находятся контейнеры A, B, C -->
    <div class="box-4"> D </div>
</div>
```

И немного изменим CSS :

```
.container {
    display: grid;
    gap: 50px;
    height: 100vh;

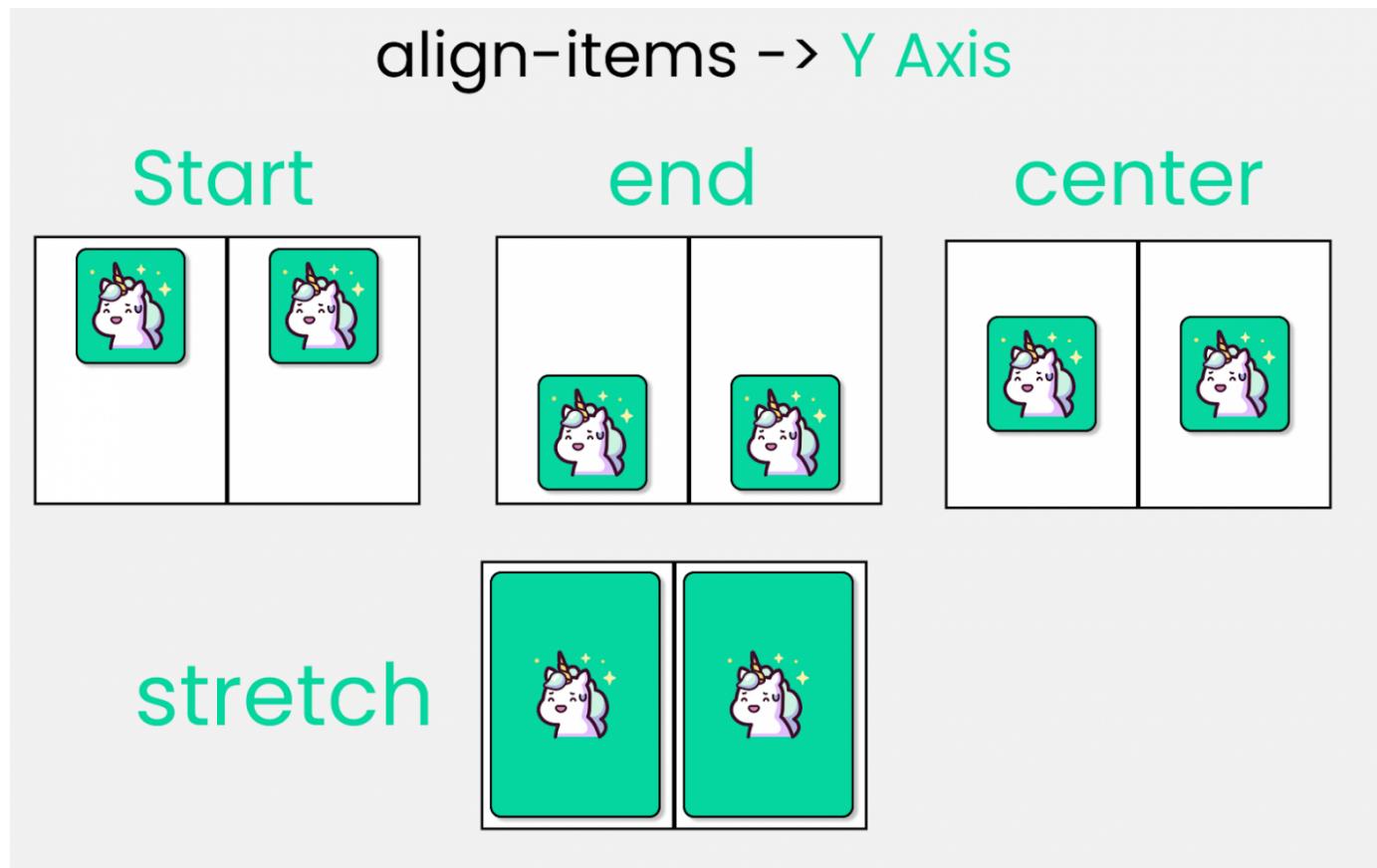
    /* Каждый контейнер имеет размер 200px на 200px */
    grid-template-rows: 200px 200px;
```

```
grid-template-columns: 200px 200px;

/* ! */
justify-items : end;
}
```

align-items

Данное свойство используется для позиционирования грид-элементов внутри грид-контейнера вдоль **поперечной оси**. Оно принимает **4** возможных значения:



style.css :

```
.container {
  display: grid;
  gap: 50px;
  height: 100vh;
  grid-template-rows: 200px 200px;
  grid-template-columns: 200px 200px;

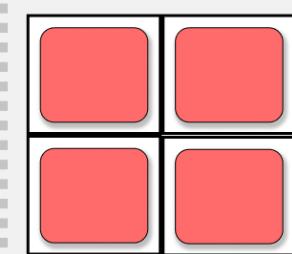
  /* ! */
  align-items: center;
}
```

justify-content

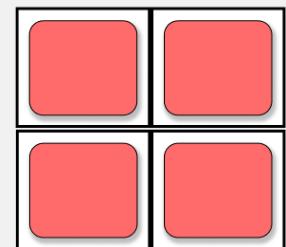
Данное свойство используется для позиционирования самого грида внутри грид-контейнера вдоль **основной оси**. Оно принимает 7 возможных значений:

justify-content -> X Axis

grid container

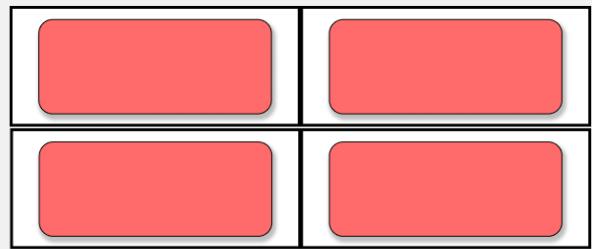
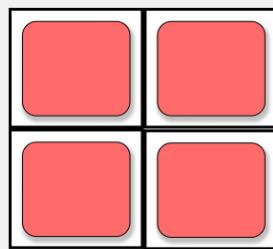


grid container



start

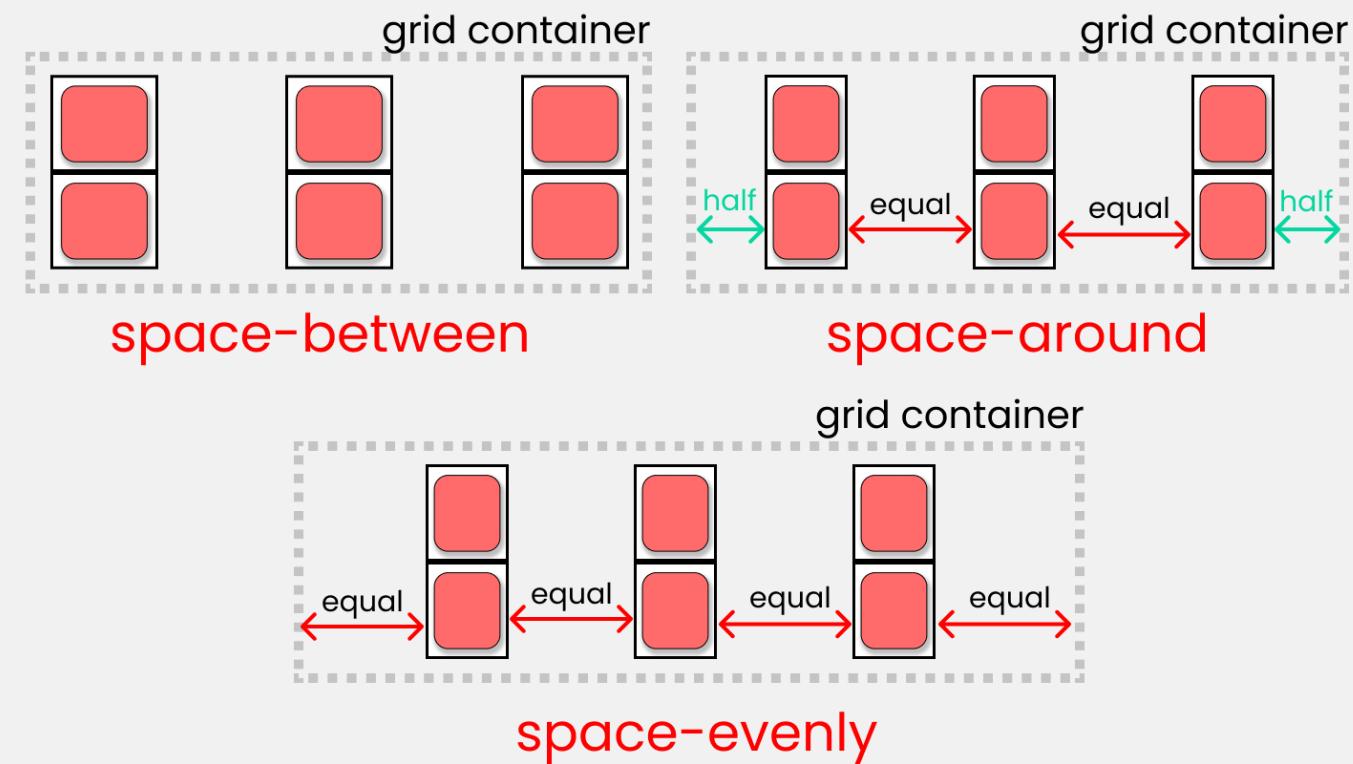
end



center

stretch

justify-content → X Axis



style.css :

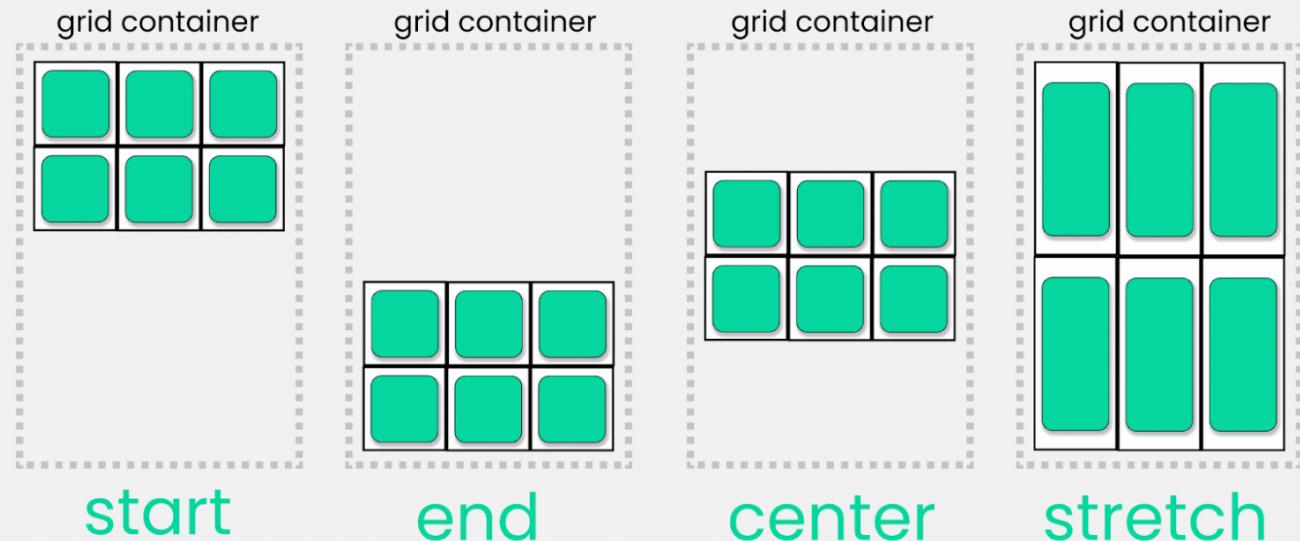
```
.container {
  display: grid;
  gap: 50px;
  height: 100vh;
  grid-template-rows: 200px 200px;
  grid-template-columns: 200px 200px;

  /* ! */
  justify-content: center;
}
```

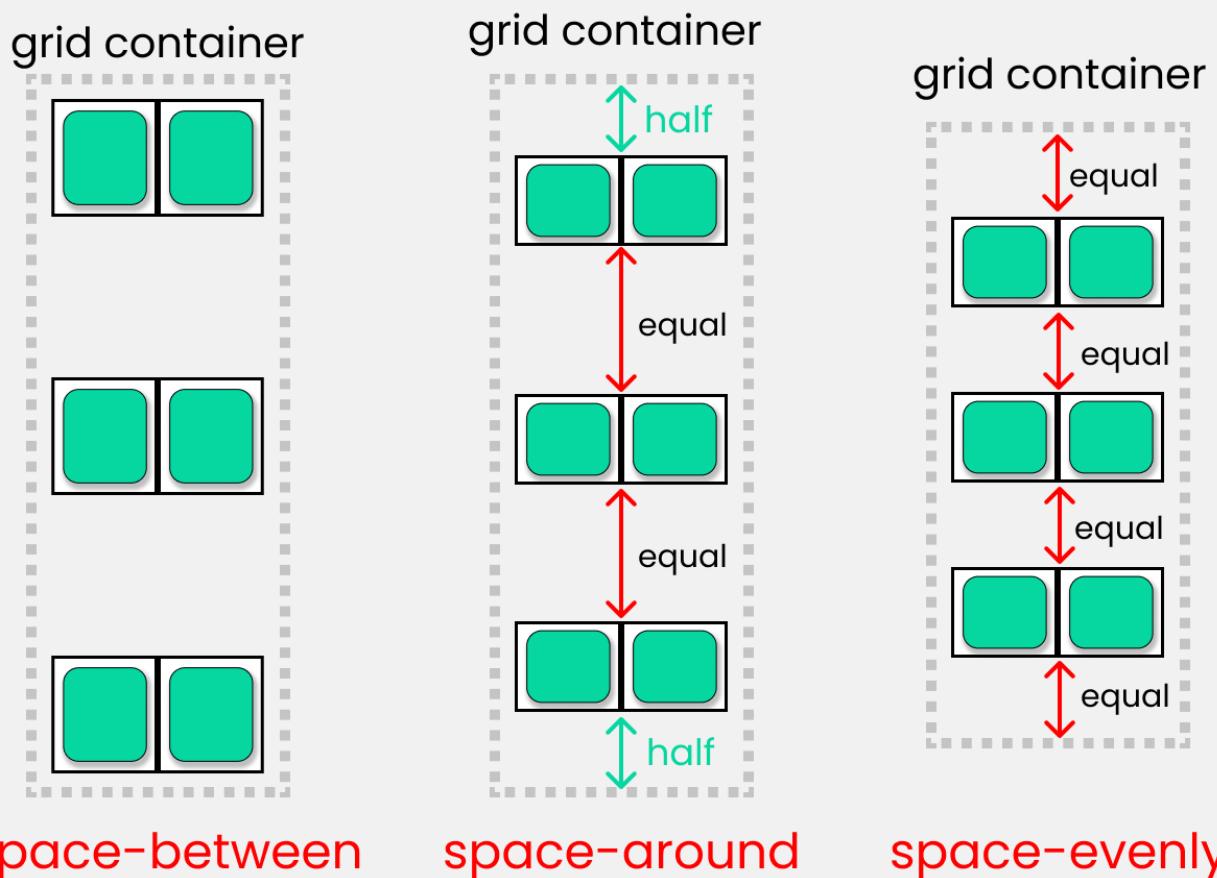
align-content

Данное свойство используется для позиционирования самого грида внутри грид-контейнера вдоль **поперечной оси**. Оно принимает **7** возможных значений:

align-content -> Y Axis



align-content -> Y Axis



style.css :

```
.container {
  display: grid;
  gap: 50px;
  height: 100vh;
```

```
grid-template-rows: 200px 200px;  
grid-template-columns: 200px 200px;  
  
/* ! */  
align-content : center;  
}
```

Свойства грид-элементов

The Grid Children Properties



Шкала CSS Grid

Данная шкала показывает, как вычисляются строки и колонки при их объединении. Для этого используется два вида единиц:

+39

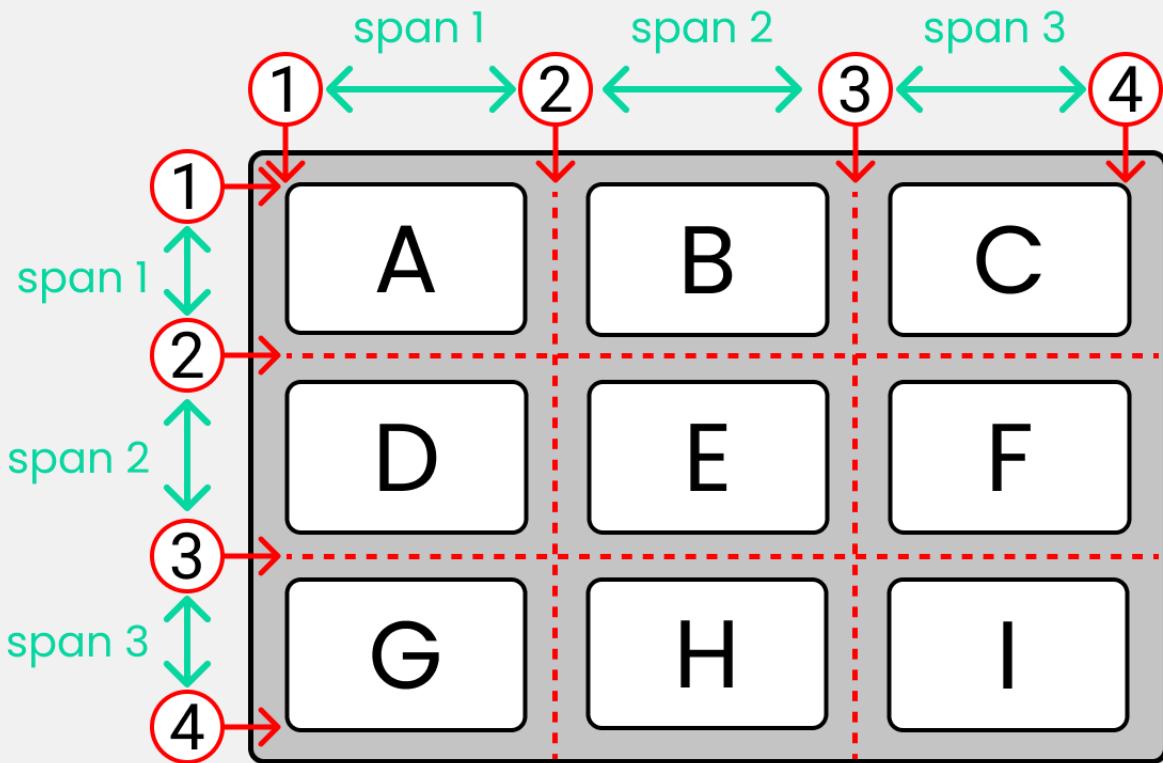
410



4

Комментарии Статья

Grid Scale



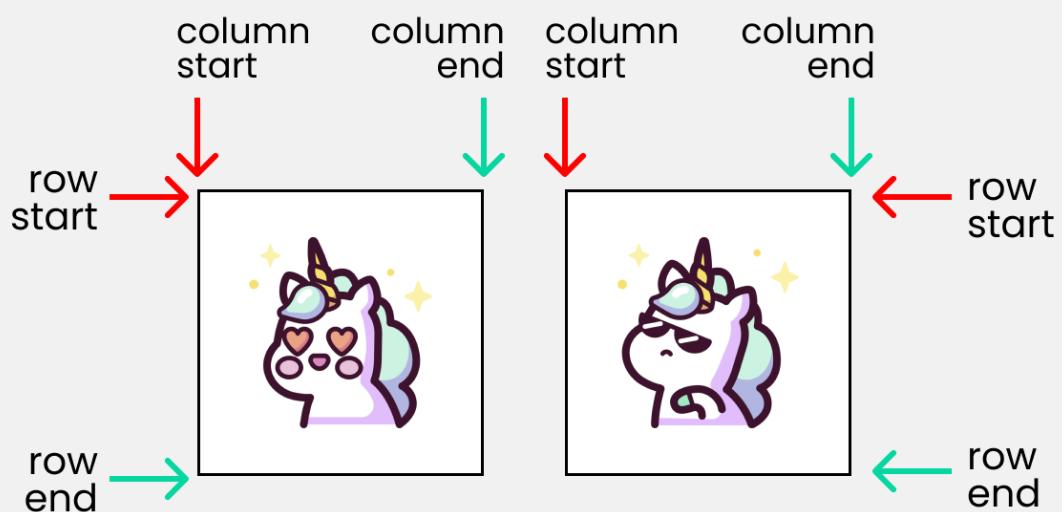
3 x 3 layout



/Joy Shaheb

На представленной ниже иллюстрации показаны начальные и конечные точки строк и колонок в одной ячейке:

grid column & row -> **start & end** points



index.html :

```
<div class="container">
  <div class="box-1"> A </div>
  <div class="box-2"> B </div>
  <div class="box-3"> C </div>
  <div class="box-4"> D </div>
</div>
```

При использовании функции `repeat()` мы можем установить одинаковую ширину/высоту для колонок/строк. Пример с колонками:

```
grid-template-columns : repeat(4, 1fr);
```

Это аналогично следующему:

```
grid-template-columns : 1fr 1fr 1fr 1fr;
```

Небольшая заметка

A Quick Note !



При использовании единицы измерения `fr`, доступное пространство делится на равные части.

```
grid-template-columns : repeat(4, 1fr);
```

В данном случае доступное пространство делится на 4 равные части.

Продолжаем веселиться!

grid-columns: start/end

Данное свойство позволяет объединять **колонки**. Оно является сокращением для:

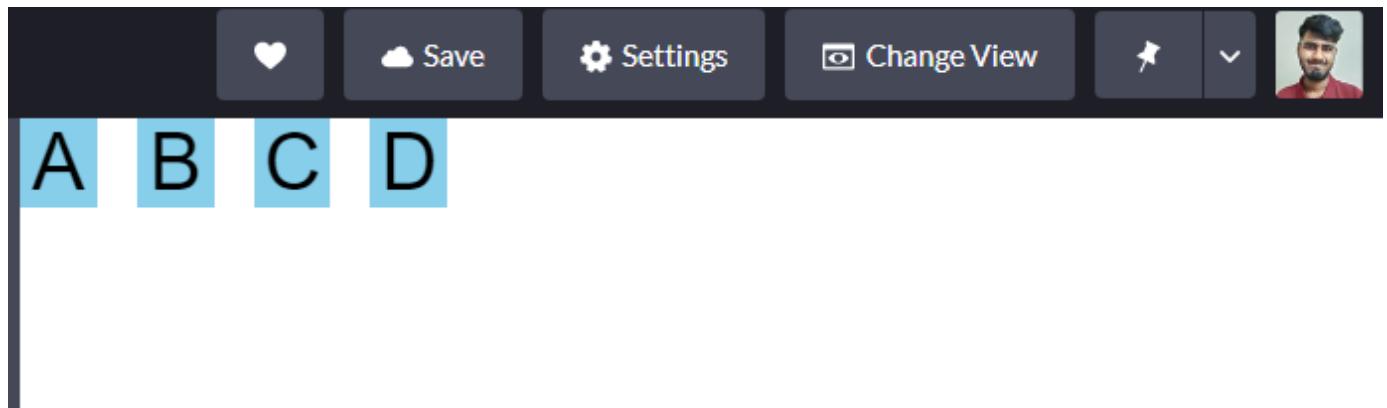
- `grid-column-start`
- `grid-column-end`

`style.css :`

```
.container {
  display: grid;
  gap: 20px;
  height: 100vh;

  grid-template-columns: repeat(12, 1fr);
  grid-template-rows: repeat(12, 1fr);
}
```

Результат:



Мы разделили доступное пространство на 12 равных частей как по ширине, так и по высоте. 1 контейнер занимает 1 часть или фракцию. В данном случае 8 фракций остались невостребованными.

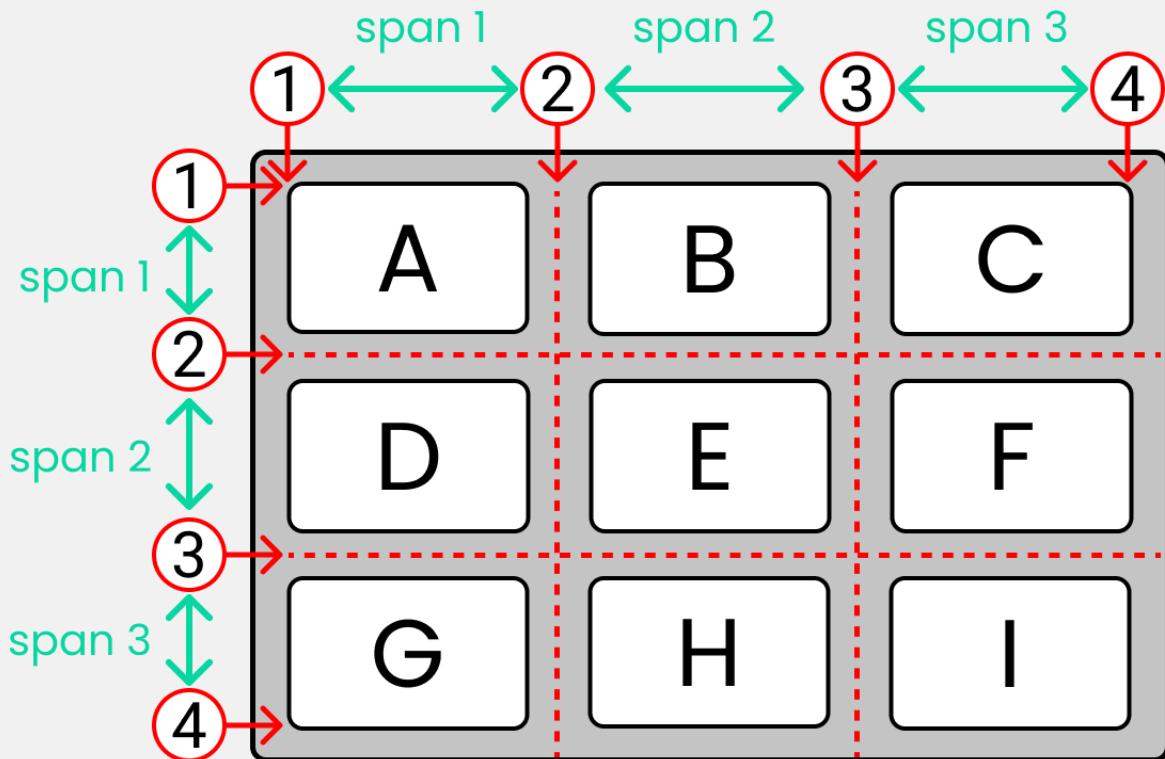
Поскольку мы говорим о свойствах дочерних элементов, имеет смысл разделить их стили:

```
.box-1 {}
.box-2 {}
```

```
.box-3 {}
.box-4 {}
```

Вернемся к шкале. Мы разбираемся с колонками – поэтому пока не обращайте внимания на строки.

Grid Scale



3 X 3 layout

Каждый класс `.box-*` по умолчанию имеет такой масштаб (scale):



```
grid-column-start: 1;
grid-column-end: 2;

/* Сокращение */
grid-column: 1 / 2
```

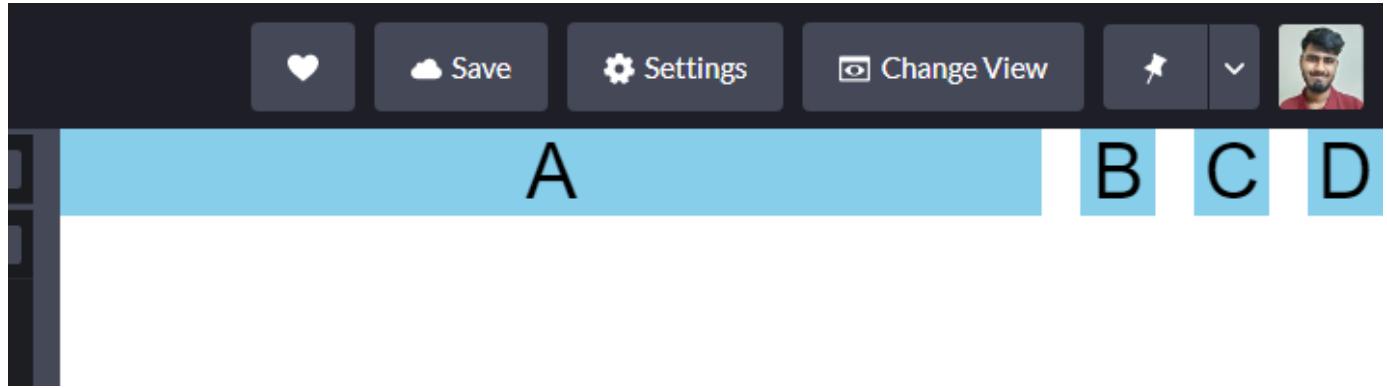
Это можно переписать с помощью ключевого слова `span`:

```
grid-column : span 1;
```

Давайте "присвоим" 8 фракций .box-1 :

```
.box-1 {
  grid-column: 1 / 10
}
```

Результат:



Небольшая заметка

Как мы производим вычисления? box-1 занимает 1 часть. Кроме этого, к ней добавляется еще 8 частей. И еще 1 в конце. Получается: $8 + 1 + 1 = 10$.

Как использовать ключевое слово span

Считается, что использование span делает код более читаемым.

В этом случае нам просто нужно добавить к box-1 8 частей:

```
.box-1 {
  grid-column: span 9;
}
```

Это даст такой же результат.

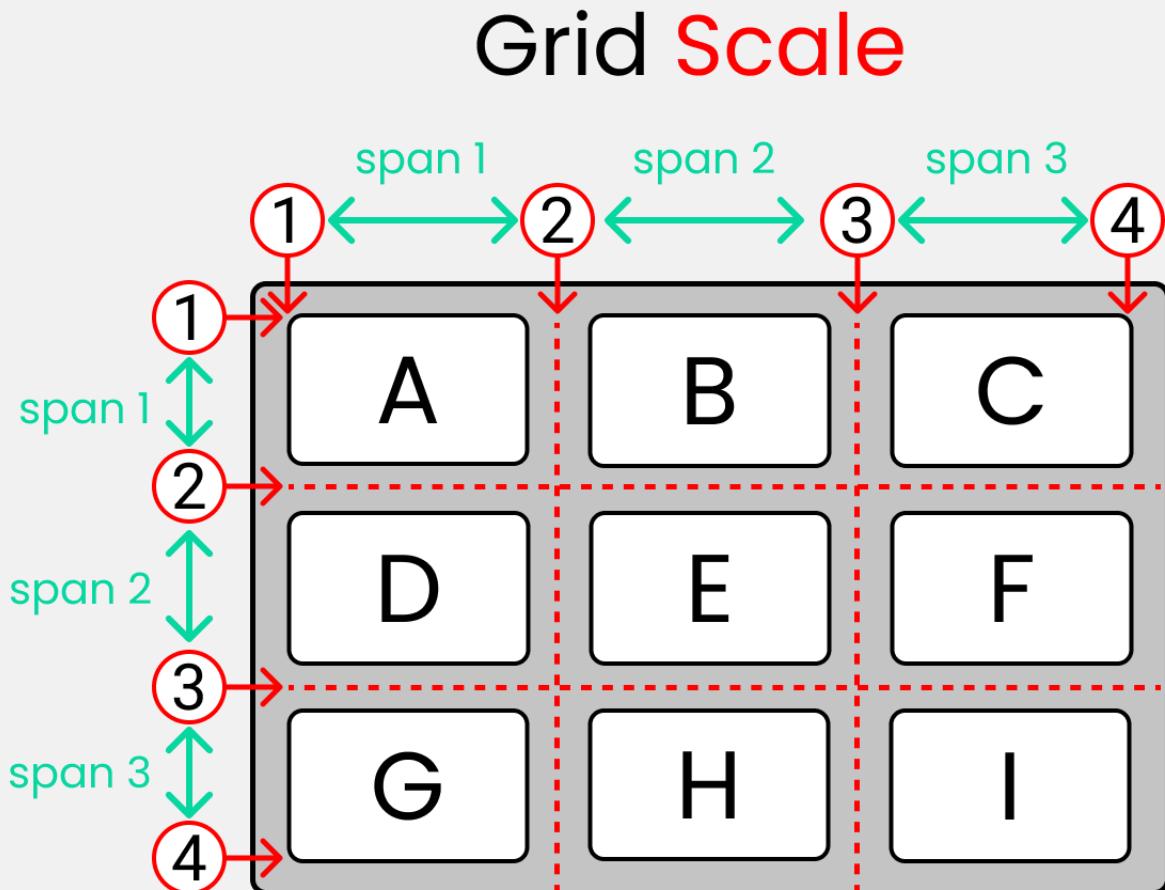
grid-row: start/end

Данное свойство позволяет объединять **строки**. Оно является сокращением для:

- grid-row-start

- grid-row-end

Теперь сосредоточимся на строках:



3 X 3 layout

Давайте добавим к box-1 9 частей:



```
.box-1 {
  grid-row : 1 / 11;
}
```

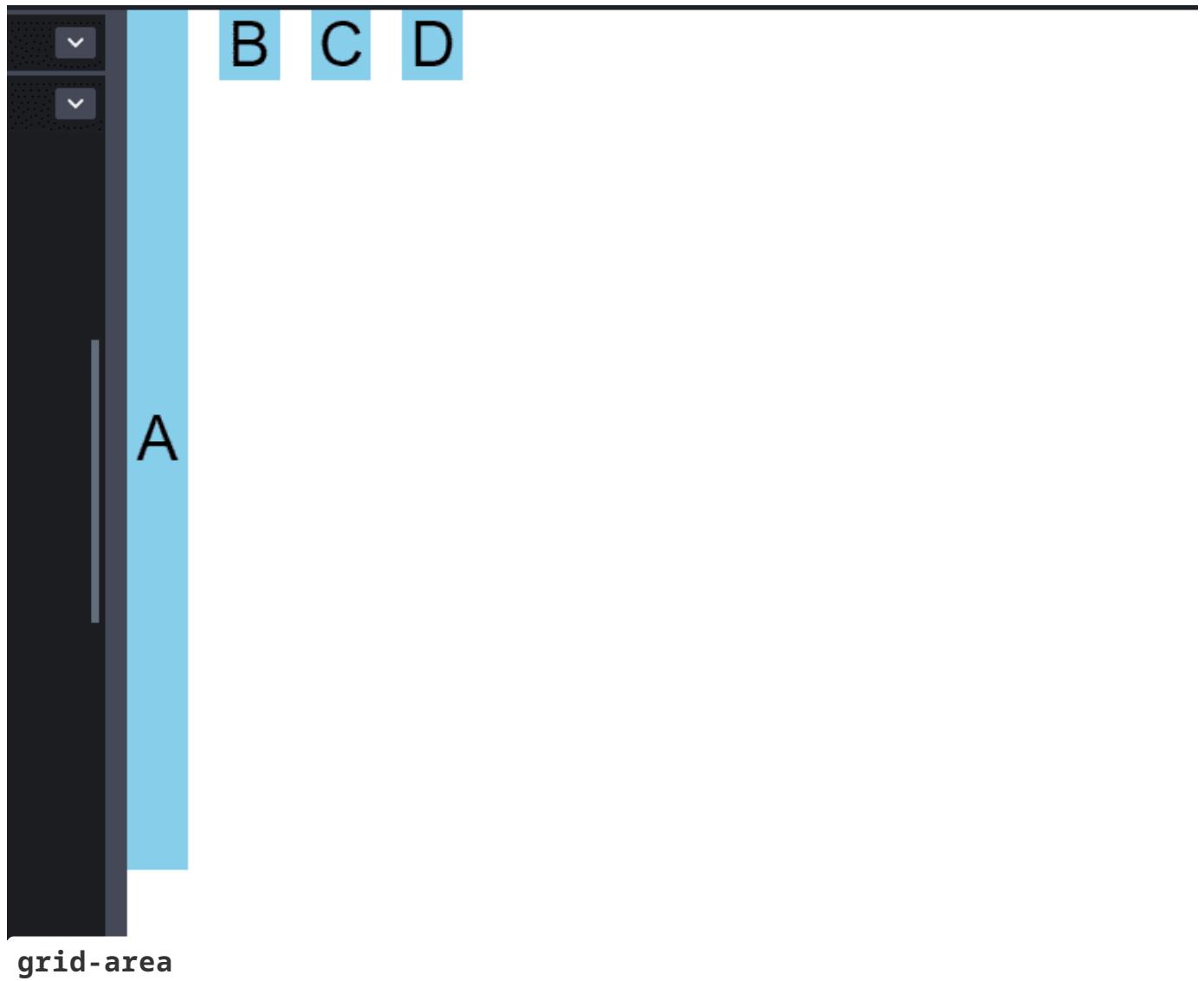
Расчет выглядит так: box-1 занимает 1 часть + 9 частей + 1 часть в конце, получается $9 + 1 + 1 = 11$.

Вот вариант со span :

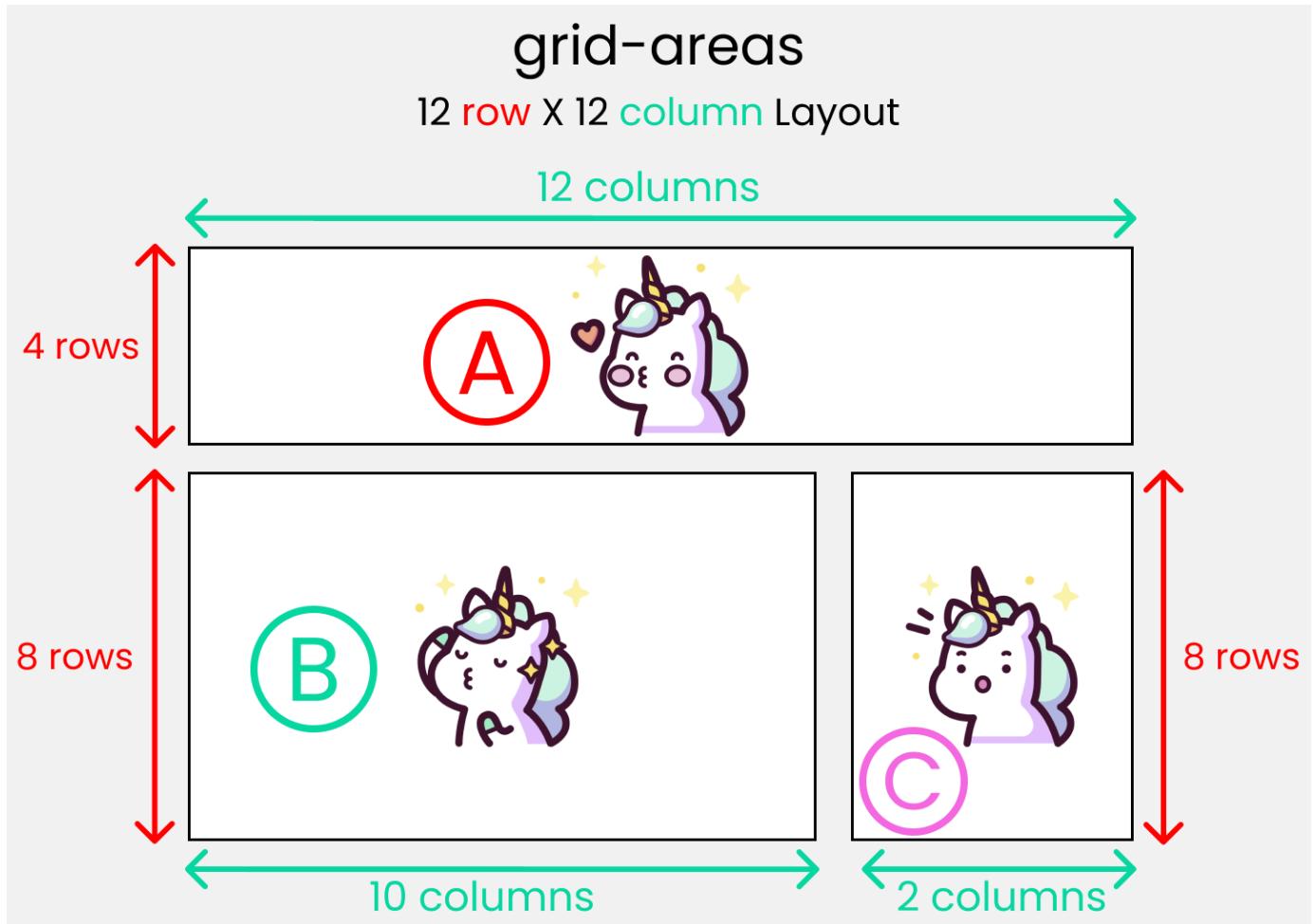
```
.box-1 {
  grid-row: span 10;
```

}

Результат:



Сначала нам нужно настроить `grid-template-areas`, о чём мы говорили выше. После этого в дочерних классах определяются названия областей, которые используются в родительском классе:



Определяем `grid-template-areas` в родительском классе:

The Parent class

```
.container{
```

`grid-template-areas :`

`"AAAAA AAAA AAAA"`

`" BBBB BBBB BBCC"`

`} " BBBB BBBB BBCC";`

`style.css :`

```
.container {
  display: grid;
  gap: 20px;
  height: 100vh;

  grid-template-areas:
    "A A A A   A A A A   A A A A"
    "B B B B   B B B B   B B C C"
    "B B B B   B B B B   B B C C";
}
```

Затем определяем `grid-area` в дочерних классах:

The Children classes

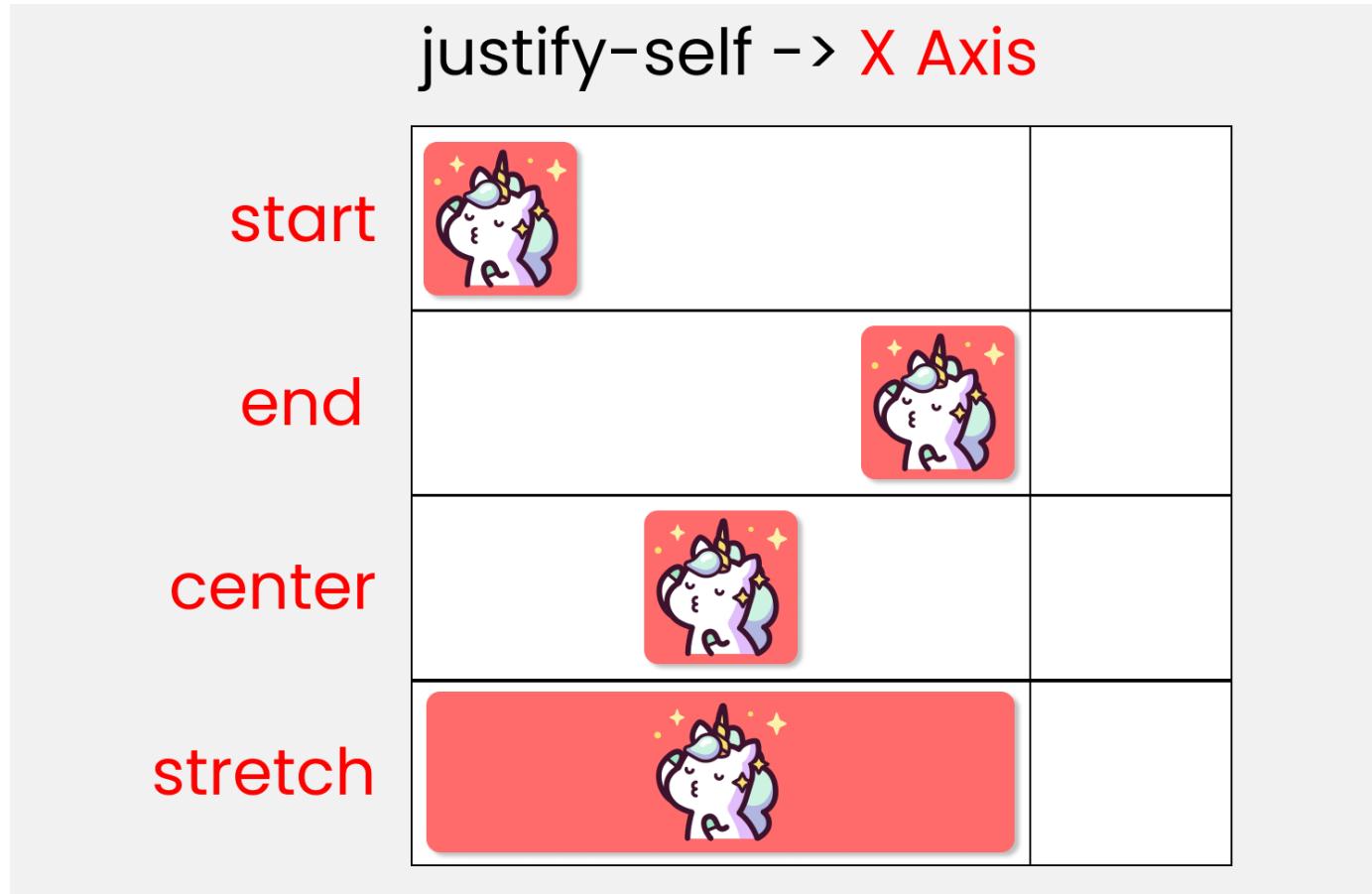
```
.box-1{
  grid-area: A ;
}
.box-2{
  grid-area: B ;
}
.box-3{
  grid-area: C ;
}
```

`style.css :`

```
.box-1 {
  grid-area: A;
}
.box-2 {
  grid-area: B;
}
.box-3 {
  grid-area: C;
}
```

justify-self

Данное свойство используется для позиционирования **отдельного** грид-элемента вдоль **основной оси**. Оно принимает **4** возможных значения:



`style.css :`

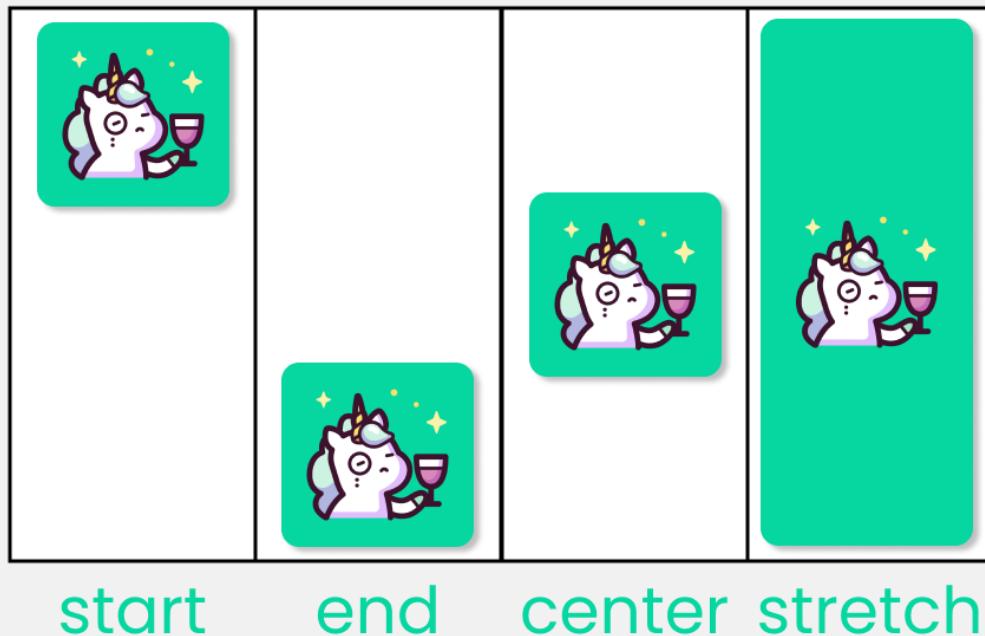
```
.container {
  display: grid;
  gap: 25px;
  height: 100vh;
  grid-template-rows: 1fr 1fr;
  grid-template-columns: 1fr 1fr;
}

.box-1 {
  /* ! */
  justify-self: start;
}
```

align-self

Данное свойство используется для позиционирования **отдельного** грид-элемента вдоль поперечной оси. Оно принимает 4 возможных значения:

align-self -> Y Axis



`style.css :`

```
.container {
  display: grid;
  gap: 25px;
  height: 100vh;
  grid-template-rows: 1fr 1fr;
  grid-template-columns: 1fr 1fr;
}

.box-1 {
  /* ! */
  align-self: start;
}
```

Сокращения для свойств CSS Grid

- place-content
- place-items
- place-self
- grid-template

- gap / grid-gap

place-content

place-content : align-content / justify-content



Y-axis



X-axis

Данное свойство является сокращением для:

- align-content
- justify-content

Пример:

```
align-content: center;  
justify-content: end;  
  
/* ! */  
place-content: center / end;
```

place-items

place-items : align-items / justify-items



Y-axis



X-axis

Данное свойство является сокращением для:

- align-items
- justify-items

Пример:

```
align-items: end;  
justify-items: center;  
  
/* ! */  
place-items: end / center;
```

place-self

place-self : align-self / justify-self



Y-axis

X-axis

Данное свойство является сокращением для:

- align-self
- justify-self

Пример:

```
align-self: start;  
justify-self: end;  
  
/* ! */  
place-self: start / end;
```

grid-template

grid-template : grid-template-rows / grid-template-columns

Данное свойство является сокращением для:

- `grid-template-rows`
- `grid-template-columns`

Пример:

```
grid-template-rows: 100px 100px;  
grid-template-columns: 200px 200px;  
  
/* ! */  
grid-template: 100px 100px / 200px 200px;
```

gap/grid-gap

gap : row-gap column-gap;

Данное свойство является сокращением для:

- `row-gap`
- `column-gap`

Пример:

```
row-gap: 20px ;  
column-gap: 30px ;  
  
/* ! */  
gap: 20px 30px ;
```

Заключение

Теперь в ваших руках имеется мощное средство для создания адаптивных макетов веб-страниц.

VPS-хостинг с быстрыми NVMe-дисками и посutoчной оплатой. Загрузка своего ISO.

Облачные серверы нового поколения

ЗАКАЗАТЬ

Теги: grid, css, guide, cheatsheet, макет, схема, дизайн, руководство, шпаргалка, blueprint, layout

Хабы: Блог компании Маклауд, Веб-разработка, CSS, Программирование

 **Маклауд**
Облачные серверы на базе AMD EPYC

[Подписаться](#)

Сайт

 **226** **51.2**
Карма Рейтинг

Igor Agarov @aio350

JavaScript Developer

[Подписаться](#)



 Комментарии 4

Публикации

ЛУЧШИЕ ЗА СУТКИ **ПОХОЖИЕ**



petr97

18 часов назад

Еще один год из жизни ReactOS

 6 мин  6.2K

Ретроспектива

 +71 17 26 shiru8bit

17 часов назад

Новый год и Atari 2600

 Средний  16 мин  1.8K

Кейс

 +52 14 3 Erwinmal

21 час назад

Главные мемы 2023 года: атомный советпанк, русы с ящерами, барбенгеймер и гусь-матерщинник

 Простой  7 мин  5.3K

Обзор

 +47 23 2 Rouse

20 часов назад

Process Memory Map

 3 мин  3.3K

Роадмэп

 +44 37 11 Rundik

10 часов назад

Традиционный новогодний Хабрачат-2024

 1 мин  984 +20 2 0 CyberexTech

23 часа назад

Электроника для самых маленьких: или еще один UV излучатель для активации фотополимера

Средний 4 мин 2.6K

Кейс

+20

25

13



SkyZion

2 часа назад

Белгород. Telegram-bot для поиска укрытий

13 мин 1.9K

Кейс

+18

3

2



PatientZero

23 часа назад

Быстрый парсинг 8-битных целых чисел

Простой 6 мин 4.8K

Обзор

Перевод

+18

41

14



Exosphere

10 часов назад

Поздравление-загадка от Хабра

1 мин 1.3K

+14

1

0



SkywardFire

6 часов назад

Статистика по Linux за 2023

Простой 3 мин 3.5K

Дайджест

+13

4

5

[Показать еще](#)

Ваш аккаунт

Профиль

Трекер

Диалоги

Настройки

ППА

Разделы

Статьи

Новости

Хабы

Компании

Авторы

Песочница

Информация

Устройство сайта

Для авторов

Для компаний

Документы

Соглашение

Конфиденциальность

Услуги

Корпоративный блог

Медийная реклама

Нативные проекты

Образовательные

программы

Стартапам



Настройка языка

Техническая поддержка

© 2006–2023, Habr

ИНФОРМАЦИЯ

Сайт	macloud.ru
Дата регистрации	17 марта 2021
Дата основания	4 марта 2021
Численность	11-30 человек
Местоположение	Россия
Представитель	Mikhail

БЛОГ НА ХАБРЕ

9 июл 2021 в 14:00

Беспроводной тачпад из смартфона

 14K  35

8 июл 2021 в 19:00

Безопасный ввод и сохранение зашифрованных паролей в конфигах Linux: пишем скрипт на Python

 16K  14

8 июл 2021 в 16:00

История российского IPO

 11K  3

7 июл 2021 в 19:00

Интервью с создателем SQLite (часть 2): Android 2005, хвала Кнуту, 100% тестовое покрытие, собственная CVS

 15K  12

7 июл 2021 в 16:00

Чему поучиться у братьев Райт – как резать фичи и запускать MVP

 17K  48