Layout

Александр Белёв

Layout (раскладка) –

взаимное расположение крупных блоков страницы.

Grid systems

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonum

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, cons ectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor

pat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, cons ectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor

Grid systems

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, cons ectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet

dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril

delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, cons ectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonum

PRETTY WOMAN
Julia Roberts played the character Vivian Ward in the popular 1990's film.

THE TRUE STORY BEHIND *PRETTY WOMAN*

As the classic rom-com turns 25, screenwriter J.F. Lawton looks back on how his gritty screenplay got turned into a fairy tale—and why he thinks it's the best thing that could have happened.

By KATE ERBLAND

quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus vitae sapien. Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim. Aliquam lorem ante, dapibus in, viverra quis, feugiat a, tellus. Phasellus viverra nulla ut metus varius laoreet. Quisque rutrum. Aenean imperdiet. Etiam ultricies nisi vel augue. Curabitur ullamcorper ultricies nisi. Nam eget dui. Etiam rhoncus. Maecenas tempus, tellus eget condimentum rhoncus, sem quam semper libero, sit amet adipiscing sem neque sed ipsum. Nam quam nunc, blandit vel, luctus pulvinar, hendrerit id, lorem. Maecenas nec odio et ante tincidunt tempus. Donec vitae sapien ut libero venenatis faucibus. Nullam quis ante vitae sapien vitae sapien.

Etiam sit amet orci eget eros faucibus tincidunt. Duis leo. Sed fringilla mauris sit amet nibh. Donec sodales sagittis magna. Sed consequat, leo eget bibendum sodales, augue velit cursus nunc, quis gravida magna mi a libero. Fusce vulputate eleifend sapien. Vestibulum purus quam, scelerisque ut, mollis sed, nonummy id, metus. Nullam accumsan lorem in dui. Cras ultricies mi eu turpis hendrerit fringilla. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; In ac dui quis mi consectetuer lacinia. Nam pretium turpis et arcu. Duis arcu tortor, suscipit eget, imperdiet nec, imperdiet iaculis, ipsum. Sed aliquam ultrices mauris. Integer ante arcu, accumsan a, consectetuer eget, posuere ut, mauris. Praesent adipiscing. Phasellus ullamcorper ipsum rutrum nunc. Duis arcu tortor, suscipit eget, imperdiet nec, imperdiet iaculis, ipsum. Sed aliquam ultrices mauris ante arcu ante arcu.

Integer ante arcu, accumsan a, consectetuer eget, posuere ut, mauris. Praesent adipiscing. Phasellus ullamcorper ipsum rutrum nunc. Duis arcu tortor, suscipit eget, imperdiet nec, imperdiet iaculis, ipsum. Sed aliquam ultrices mauris. Integer ante arcu, accumsan a, consectetuer eget, posuere ut, mauris. Praesent adipiscing. Phasellus ullamcorper ipsum rutrum nunc nec ipsum.

Always a Debate

The lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus vitae sapien. Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim. Aliquam lorem ante, dapibus in, viverra quis, feugiat a, tellus. Phasellus viverra nulla ut metus varius laoreet. Quisque rutrum. Aenean imperdiet. Etiam ultricies nisi vel augue. Curabitur ullamcorper ultricies nisi. Nam eget dui. Etiam rhoncus. Maecenas tempus, tellus eget condimentum rhoncus, sem quam semper libero, sit amet adipiscing sem neque sed ipsum. Nam quam nunc, blandit vel, luctus pulvinar, hendrerit id, lorem. Maecenas nec odio et ante tincidunt tempus. Donec vitae sapien ut libero venenatis faucibus. Nullam quis ante vitae sapien vitae sapien.

Etiam sit amet orci eget eros faucibus tincidunt. Duis leo. Sed fringilla mauris sit amet nibh. Donec sodales sagittis magna. Sed consequat, leo eget bibendum sodales, augue velit cursus nunc, quis gravida magna mi a libero. Fusce vulputate eleifend sapien. Vestibulum purus quam, scelerisque ut, mollis sed, nonummy id, metus. Nullam accumsan lorem in dui. Cras ultricies mi eu turpis hendrerit fringilla. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; In ac dui quis mi consectetuer lacinia. Nam pretium turpis et arcu. Duis arcu tortor, suscipit eget, imperdiet nec, imperdiet iaculis, ipsum. Sed aliquam ultrices mauris. Integer ante arcu, accumsan a, consectetuer eget, posuere ut, mauris. Praesent adipiscing. Phasellus ullamcorper ipsum rutrum nunc.

Duis arcu tortor, suscipit eget, imperdiet nec, imperdiet iaculis, ipsum. Sed aliquam ultrices mauris. Integer ante arcu, accumsan a, consectetuer eget, posuere ut, mauris. Praesent adipiscing. Phasellus ullamcorper ipsum rutrum nunc. Sed aliquam ultrices mauris. Integer ante arcu, accumsan a, consectetuer eget, posuere ut, mauris. Praesent adipiscing. Phasellus ullamcorper ipsum rutrum nunc. Sed aliquam ultrices mauris. Integer ante arcu, accumsan a, consectetuer eget, posuere ut, mauris. Praesent adipiscing. Phasellus ullamcorper ipsum rutrum nunc.

BASICALLY, IT WAS
"DARK" AND "GRITTY"
BEFORE HOLLYWOOD EVEN KNEW THEY WANTED "DARK" AND "GRITTY."

Fairy Tale With A Twist

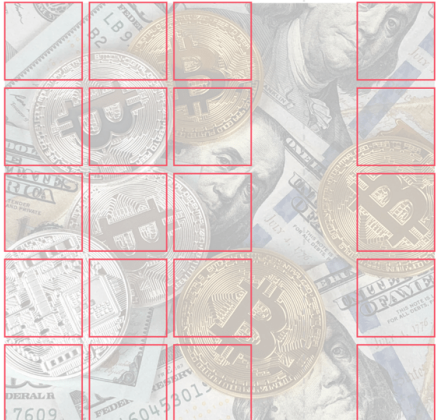
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus vitae sapien.

Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim. Aliquam lorem ante, dapibus in, viverra quis, feugiat a, tellus. Phasellus viverra nulla ut metus varius laoreet. Quisque rutrum. Aenean imperdiet. Etiam ultricies nisi vel augue. Curabitur ullamcorper ultricies nisi. Nam eget dui. Etiam rhoncus. Maecenas tempus, tellus eget condimentum rhoncus, sem quam semper libero, sit amet adipiscing sem neque sed ipsum. Nam quam nunc, blandit vel, luctus pulvinar, hendrerit id, lorem. Maecenas nec odio et ante tincidunt tempus. Donec vitae sapien ut libero venenatis faucibus. Nullam quis ante vitae sapien vitae sapien.

Etiam sit amet orci eget eros faucibus tincidunt. Duis leo. Sed fringilla mauris sit amet nibh. Donec sodales sagittis magna. Sed consequat, leo eget bibendum sodales, augue velit cursus nunc, quis gravida magna mi a libero. Fusce vulputate eleifend sapien. Vestibulum purus quam, scelerisque ut, mollis sed, nonummy id, metus. Nullam accumsan lorem in dui. Cras ultricies mi eu turpis hendrerit fringilla. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; In ac dui quis mi consectetuer lacinia. Nam pretium turpis et arcu. Duis arcu tortor, suscipit eget, imperdiet nec, imperdiet iaculis, ipsum. Sed aliquam ultrices mauris. Integer ante arcu, accumsan a, consectetuer eget, posuere ut, mauris. Praesent adipiscing. Phasellus ullamcorper ipsum rutrum nunc.

Duis arcu tortor, suscipit eget, imperdiet nec, imperdiet iaculis, ipsum. Sed aliquam ultrices mauris. Integer ante arcu, accumsan a, consectetuer eget, posuere ut, mauris. Praesent adipiscing. Phasellus ullamcorper ipsum rutrum nunc. Sed aliquam ultrices mauris. Integer ante arcu, accumsan a, consectetuer eget, posuere ut, mauris. Praesent adipiscing. Phasellus ullamcorper ipsum rutrum nunc. Sed aliquam ultrices mauris. Integer ante arcu, accumsan a, consectetuer eget, posuere ut, mauris. Praesent adipiscing. Phasellus ullamcorper ipsum rutrum nunc.

Bitcoin, the currency of the internet



Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonu nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem el eum iriure dolor in hendrerit in vulputate velit esse molestie consequa , vel illum dolore eu feugiat nulla facilisis at vero eros et Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod

Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet

OPTIMO
CHICAGO

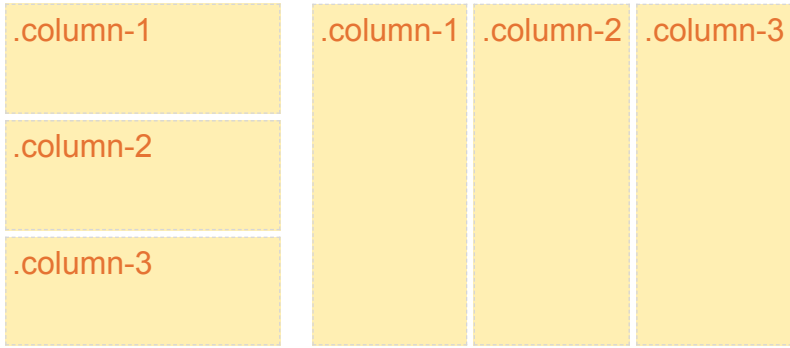
HATS HATMAKING SHOP

COLLECTION

BROWSE OUR POPULAR STYLES. AVAILABLE IN MOST COLORS AND FINISHES. DON'T SEE THE STYLE YOU ARE LOOKING FOR? OPTIMO CUSTOM

Делаем колонки

```
<div class="column-1">
</div>

<div class="column-2">
</div>
<div class="column-3">
</div>
```



Таблицы

```
<table>
    <tbody>
        <tr>
            <td style="width: 33%;">колонка раз</td>
            <td style="width: 33%;">колонка два</td>

            <td style="width: 33%;">колонка три</td>
        </tr>
    </tbody>
</table>
```

колонка раз	колонка два	колонка три

раз

два

три

Раскладка на таблицах:

- колонки выравниваются по высоте
- адекватна при переполнении
- куча лишней разметки
- не семантично

Float

```
<div class="float-left"></div>
<div class="float-left"></div>
<div class="float-left"></div>
```

```
.float-left
{
    width: 33%;
    float: left;
}
```

float: left

float: left

float: left

float: left

float: left

float: left

clear: both

Раскладка на float:

- размеры не зависят от контента
- колонки переносятся
- колонки не выравниваются по высоте
- спецэффекты (схлопывание родителя) и хаки (clearfix)
- перекрытие контента при переполнении

Inline-block

display:
inline-block

display:
inline-

display:
inline-block

Раскладка на inline-block:

- реагируют на выравнивание (text-align и vertical-align)
- можно задавать размеры
- переносятся
- лишние пробелы
- колонки не выравниваются по высоте
- при неаккуратной верстке всё может развалиться

Чем раскладывать-то тогда?

- ~~table~~ — для табличных данных
- ~~float~~ — для обтекания текстом
- ~~inline-block~~ — вы не хотите так делать

Чем раскладывать-то тогда?

- ~~table~~ — для табличных данных
- ~~float~~ — для обтекания текстом
- ~~inline-block~~ — вы не хотите так делать
- flexbox 👍
- grid 👍

Flexbox

Гибкие раскладки:

- управление распределением места
- мощные возможности для выравнивания

display: flex

Термины

флекс-контейнер (flex-container)

флекс-элемент (flex-item)

главная ось (main axis)

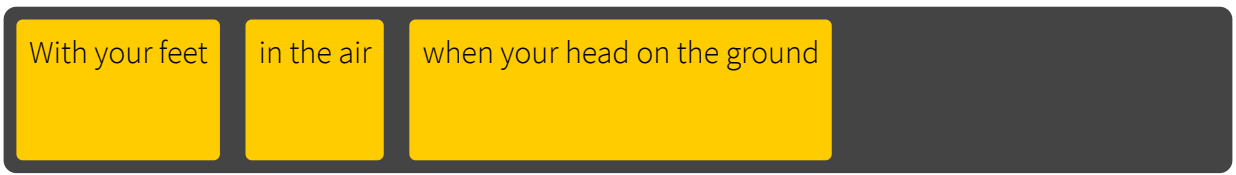
поперечная ось (cross axis)

Флекс-элементы располагаются вдоль главной оси.

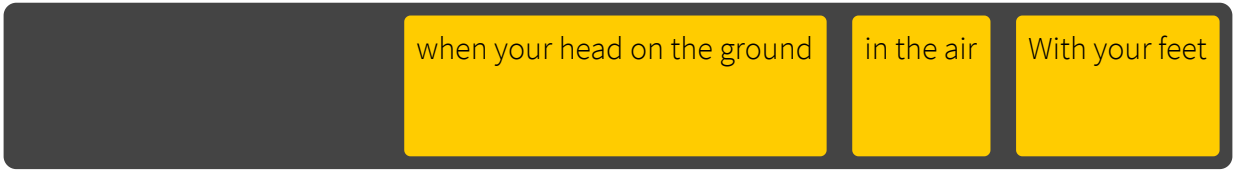
Свойство flex-direction меняет её направление.

flex-direction: row

по умолчанию



flex-direction: row-reverse



flex-direction: column

With your feet

in the air

when your head on the ground

flex-direction: column-reverse

when your head on the ground

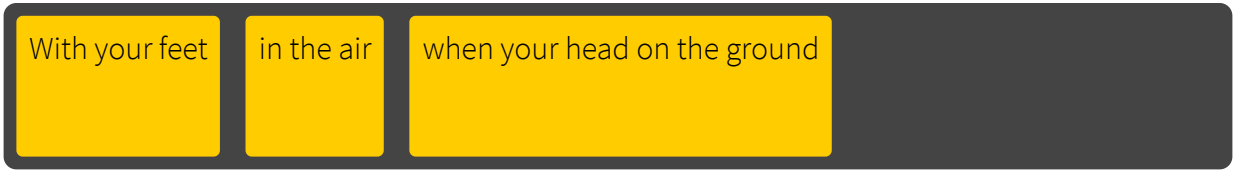
in the air

With your feet

Свойство justify-content управляет выравниванием флекс-элементов вдоль главной оси.

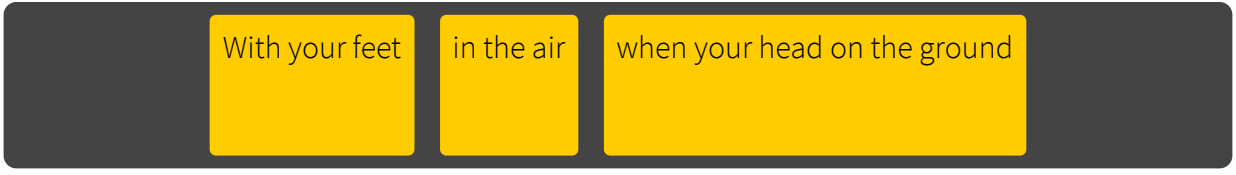
justify-content: flex-start

по умолчанию

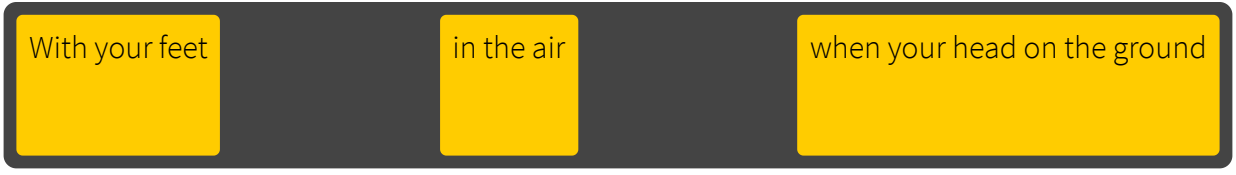


justify-content: flex-end

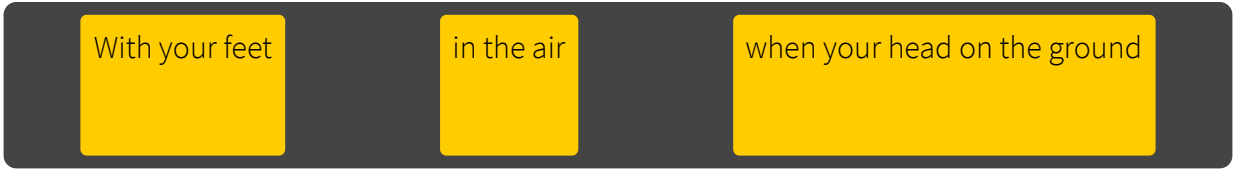
justify-content: center



justify-content: space-between

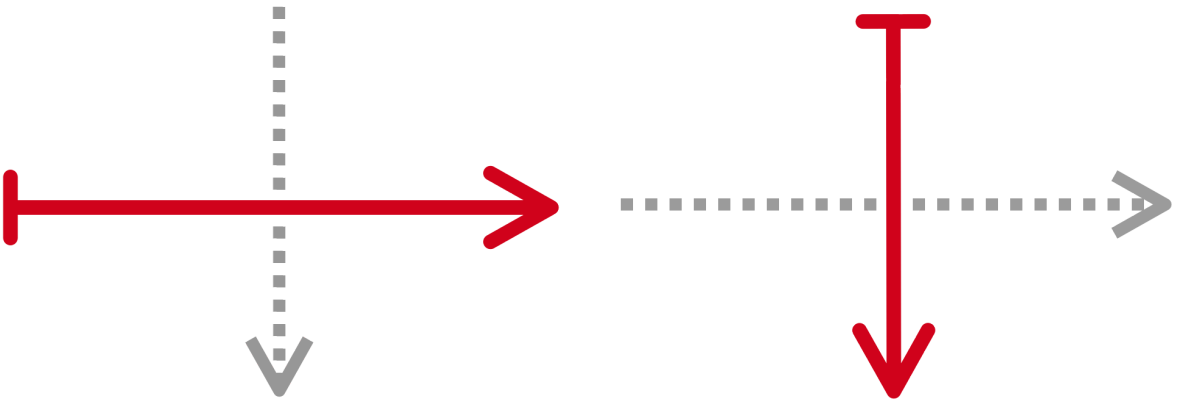


justify-content: space-around



Поперечная ось всегда перпендикулярна главной оси.

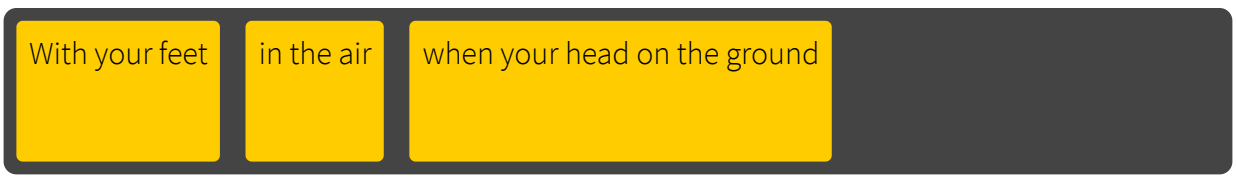
Направление изменить нельзя.



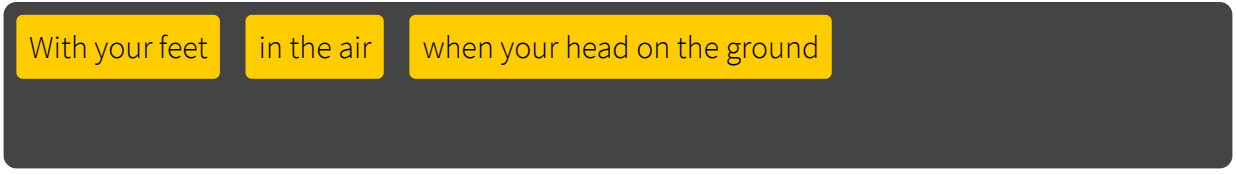
Свойство align-items управляет выравниванием флекс-элементов вдоль поперечной оси.

align-items: stretch

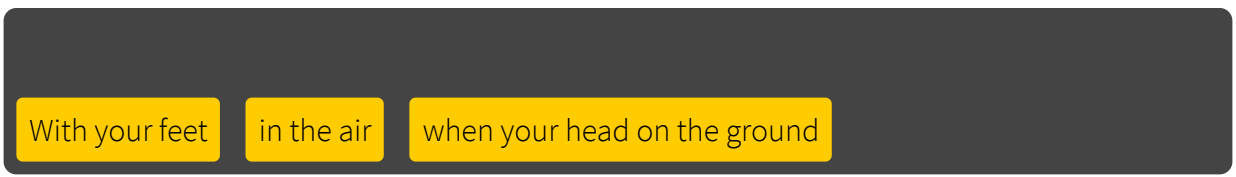
по умолчанию



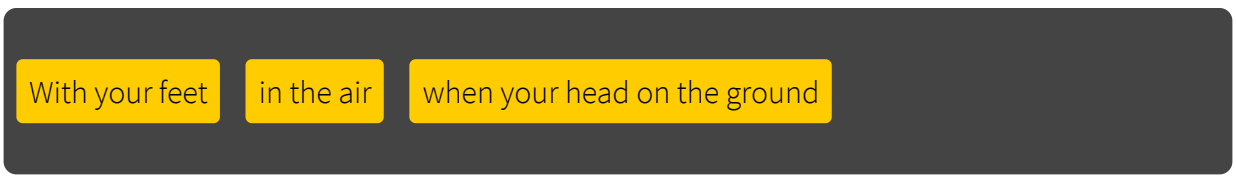
align-items: flex-start



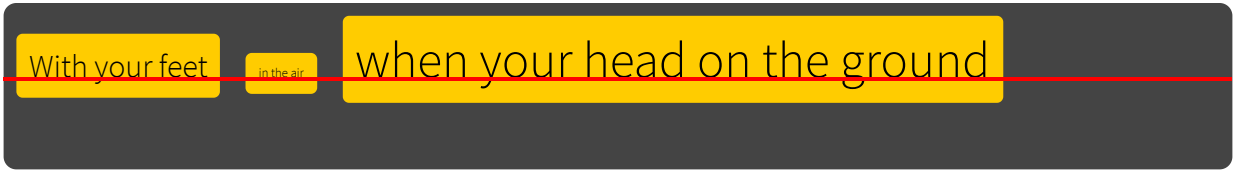
align-items: flex-end



align-items: center



align-items: baseline



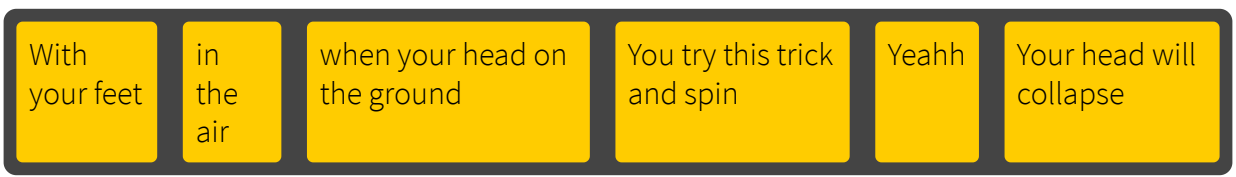
align-self даёт переопределить

выравнивание у флекс-элемента.

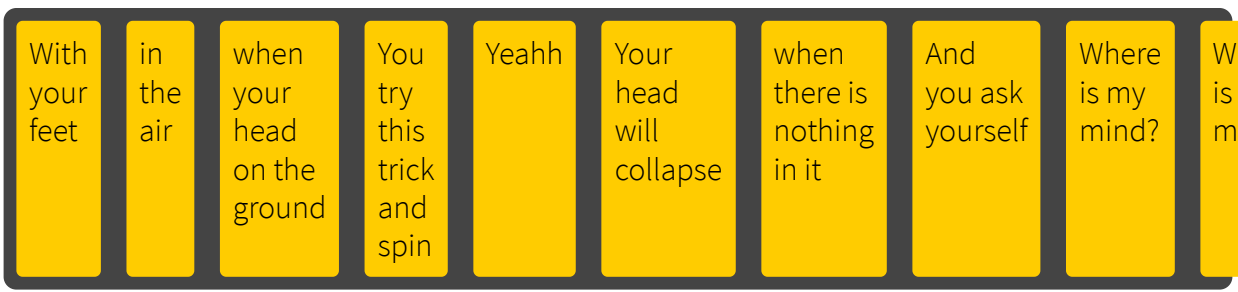
Значения те же, что и у align-items.

Что если флекс-элементов много?

Будут сжиматься до предела.

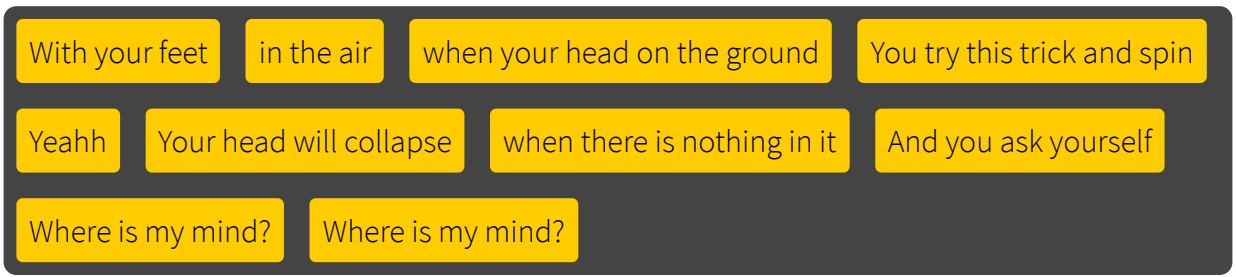


Выйдут за пределы контейнера,
но продолжат располагаться в один ряд.

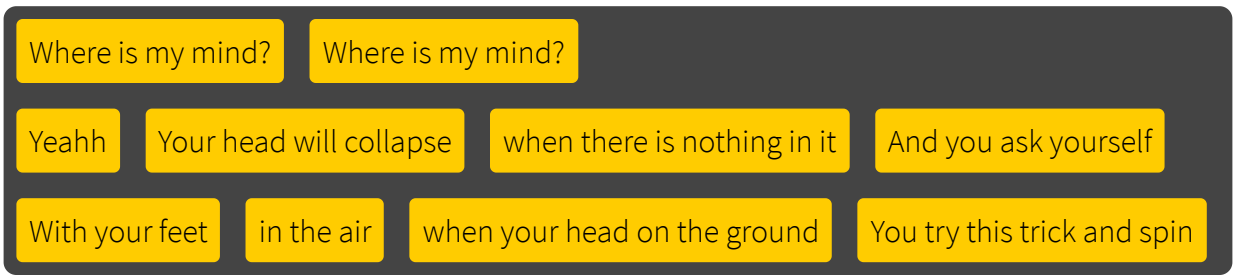


Переносом элементов управляет свойство flex-wrap, и по умолчанию перенос запрещён.

flex-wrap: ~~nowrap~~ wrap



flex-wrap: wrap-reverse



Можно комбинировать направление и перенос в свойстве flex-flow:

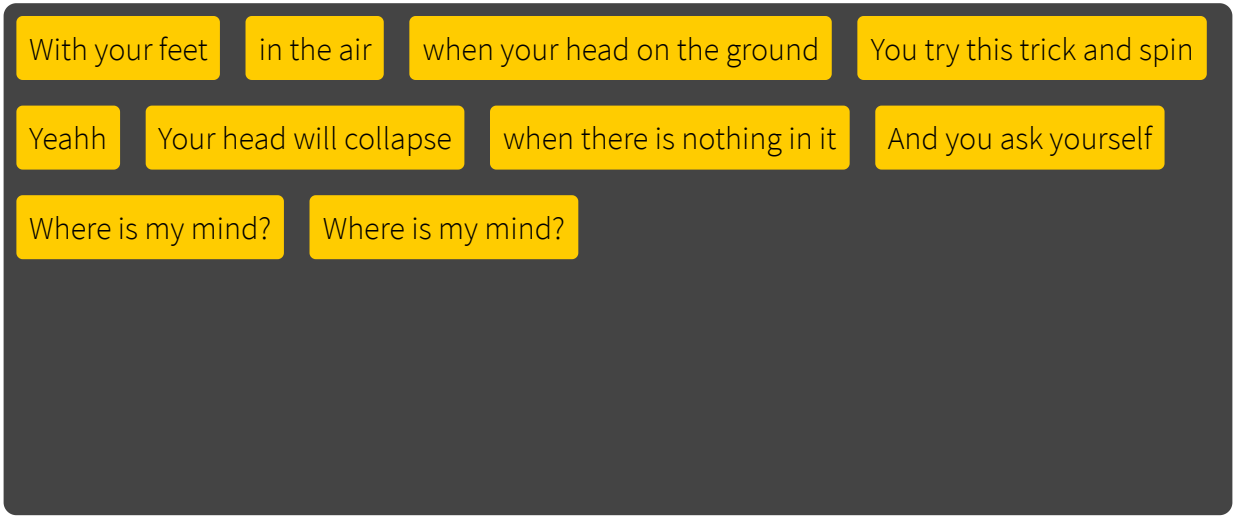
```
.container
{
    display: flex;
    flex-flow: row wrap;
}
```

Свойство align-content управляет распределением рядов флекс-элементов вдоль поперечной оси.

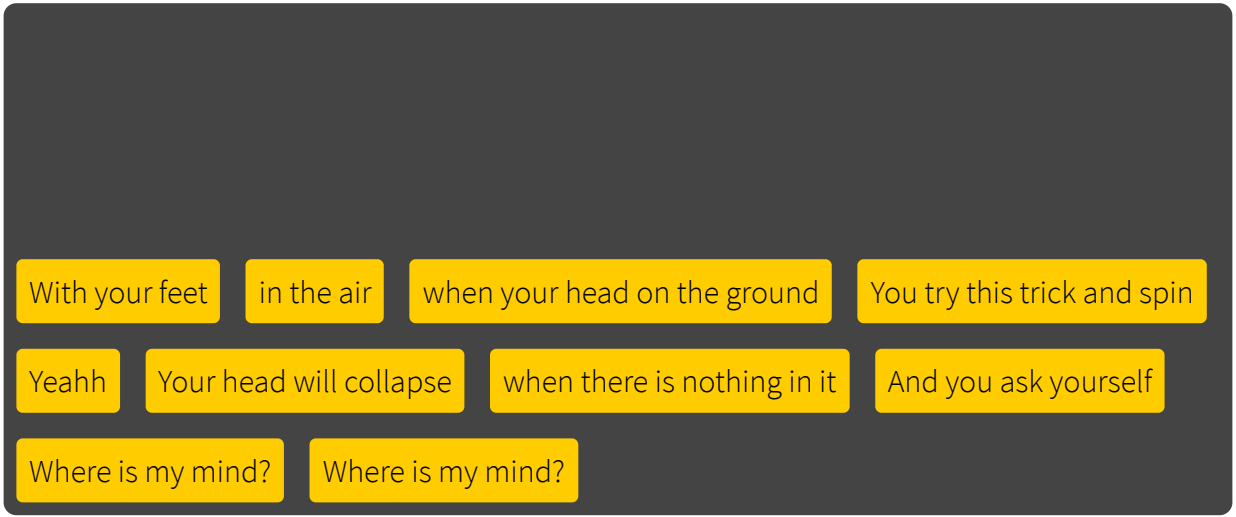
align-content: stretch

по умолчанию

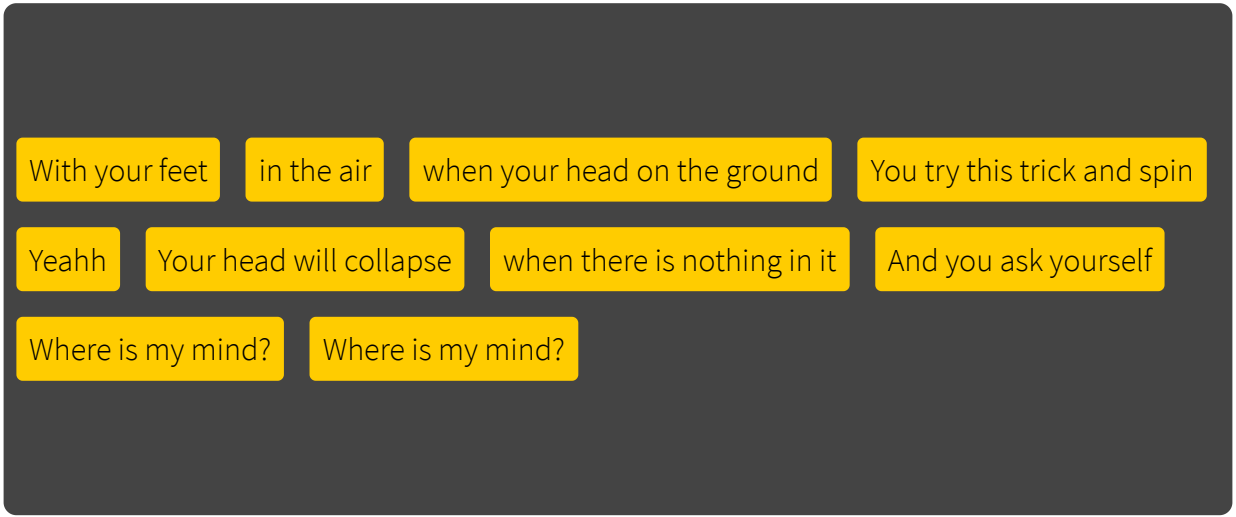
align-content: flex-start



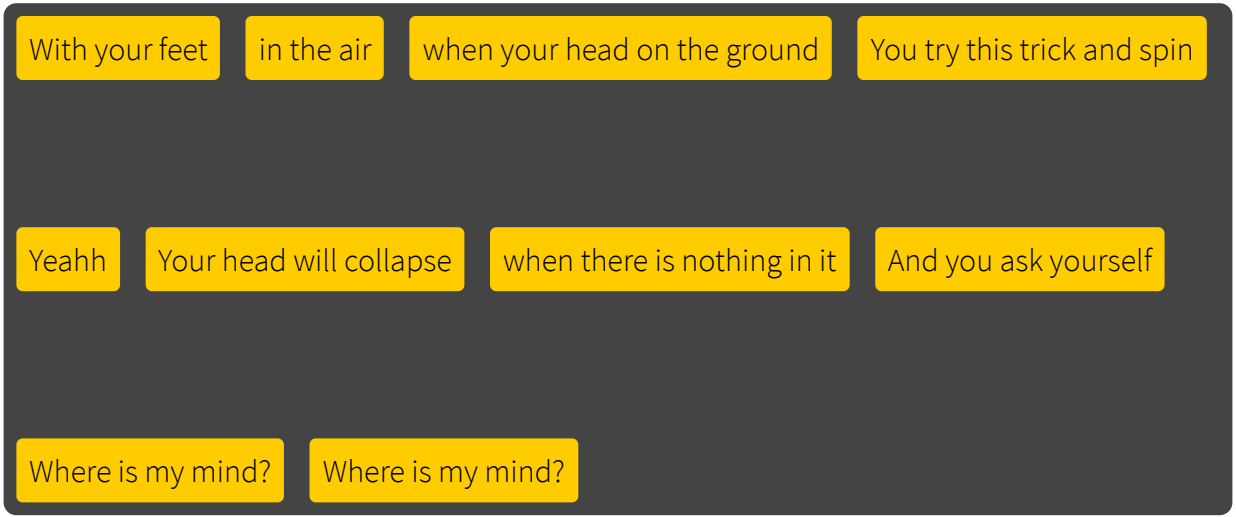
align-content: flex-end



align-content: center



align-content: space-between



align-content: space-around

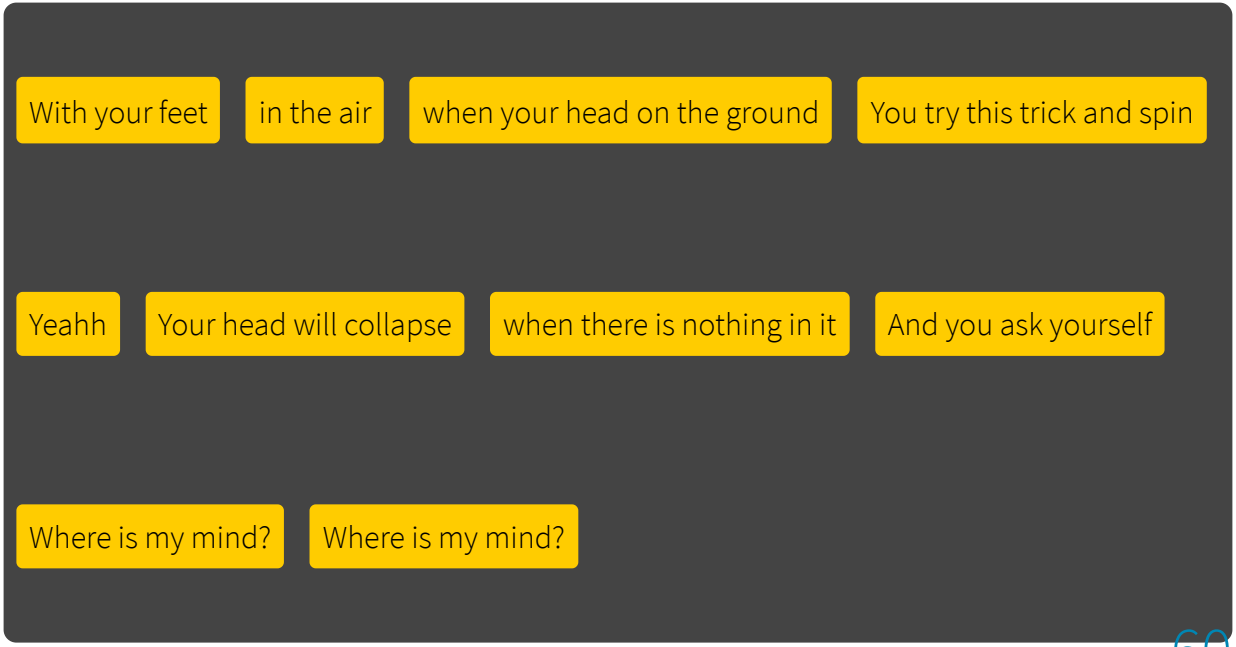
With your feet in the air when your head on the ground You try this trick and spin

Yeahh Your head will collapse when there is nothing in it And you ask yourself

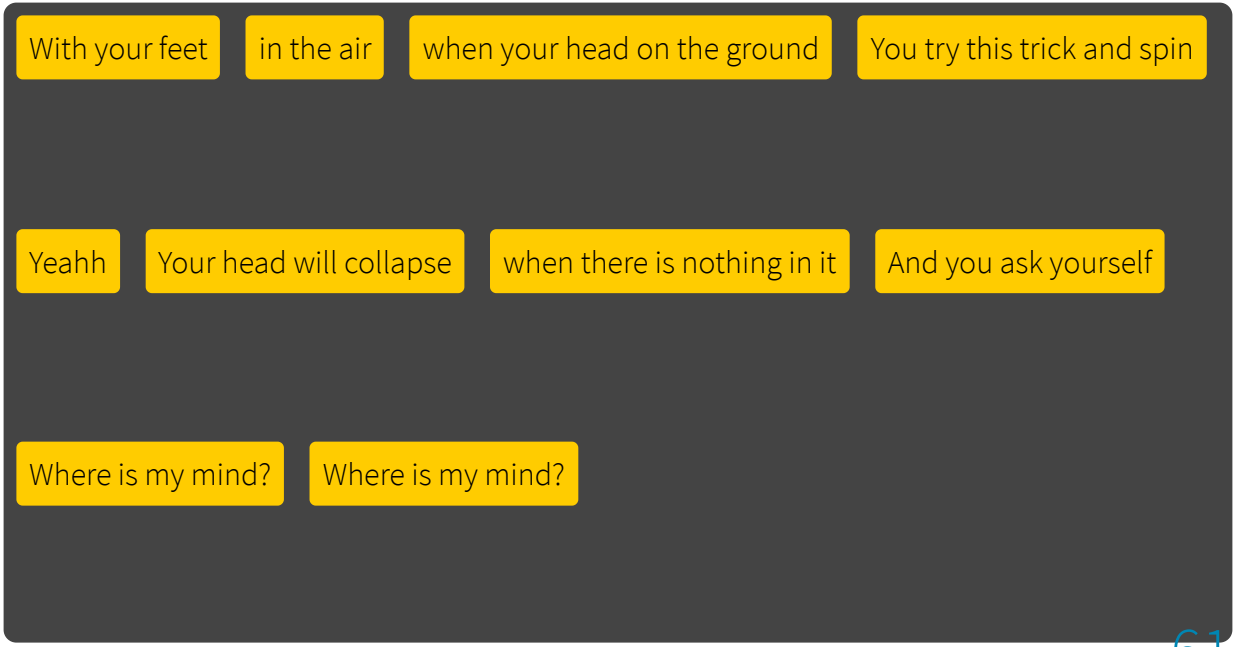
Where is my mind? Where is my mind?

Если активно align-content,
то что с align-items?

align-content: stretch; align-items: center



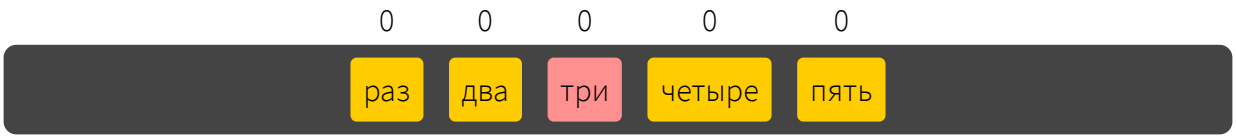
align-content: stretch; align-items: flex-start



С помощью свойства order можно менять порядок следования флекс-элементов.

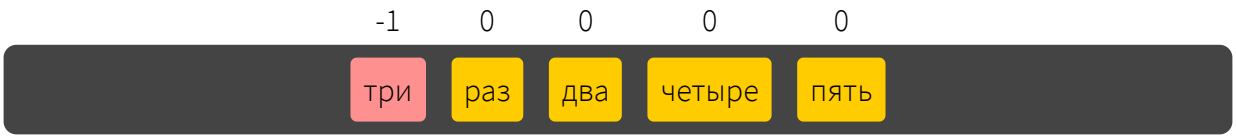
order: 0

по умолчанию

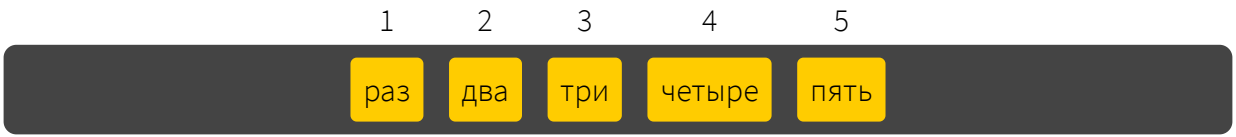


order: 1

order: -1



order



Как управлять размерами флекс-элементов?

Свободное пространство

positive free space

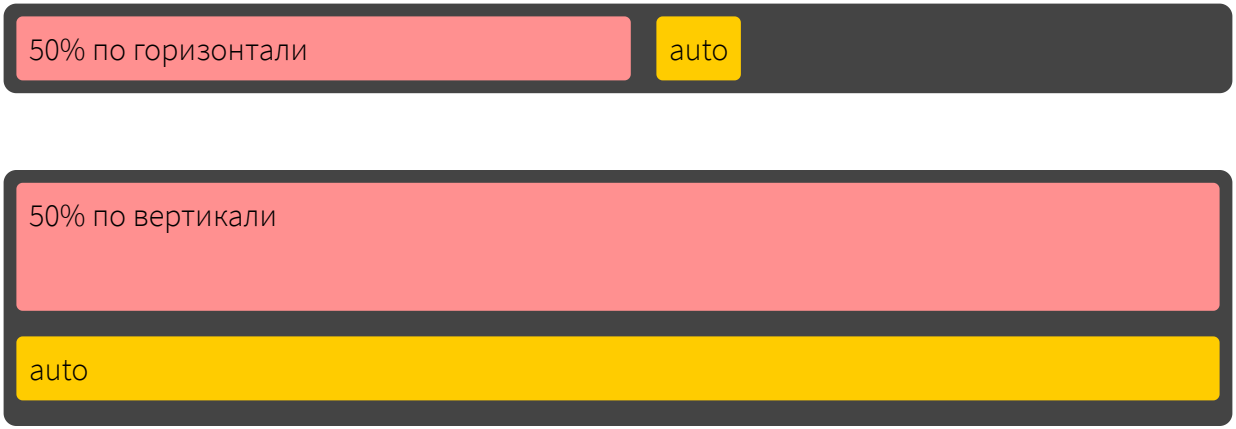
negative free space

Свойство flex-basis задает базовый размер на главной оси.

Возможные значения:

- auto/content — размер контента
- <length> — точное значение
- 0 — не учитывать размер

flex-basis: 50%



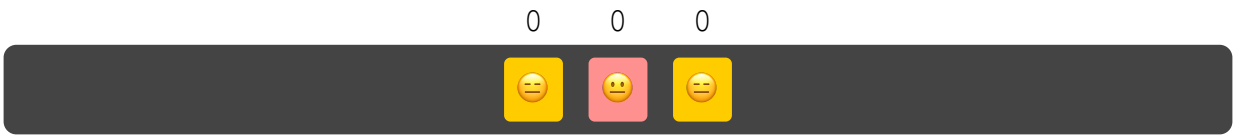
Почему размер базовый?

Это исходный размер.

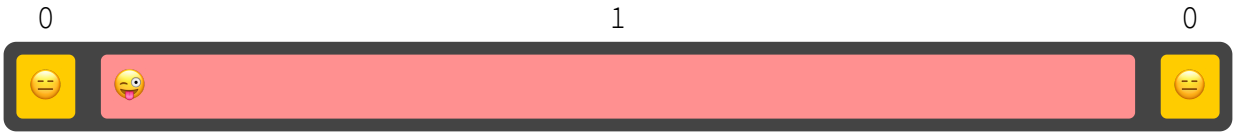
Свободное место можно распределять
в соответствии с коэффициентом жадности
флекс-элемента (flex-grow).

flex-grow: 0

по умолчанию



flex-grow: 1



flex-grow: 2

Как вычисляется итоговый размер?

1. Посчитаем свободное место:
   free space = container width - ∑ flex-basis
2. Посчитаем долю свободного места:
   fraction = free space / ∑ flex-grow
3. Вычислим итоговый размер:
   final size = flex-basis + (fraction × flex-grow)

На размер свободного места могут влиять рамки и отступы

Особенности внешних отступов:

- не схлапываются

- отступ с auto заберет все свободное место по своему направлению
- часть базового размера элемента

```
<div class="box 0"></div>
<div class="box 1"></div>
<div class="box 2"></div>
```

```
.box {
    padding: 10px;
    margin: 10px;
}

.1 {
    flex-grow: 1;
}

.2 {
    flex-grow: 2;
}
```

width = 960px

flex-basis$_i$ = 2×10px + 2×10px + 25px = **65px**

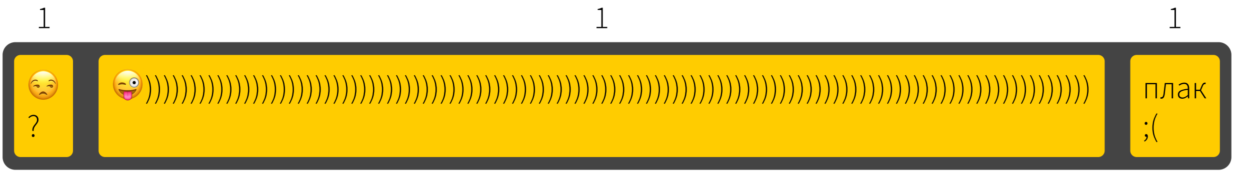
- free space = 960px - 3 × **65px** = 765px
- fraction = 765px / (0 + 1 + 2) = 255px
- final size$_1$ = **65px** + (255px × 0) = 65px;
- final size$_2$ = **65px** + (255px × 1) = 320px;
- final size$_3$ = **65px** + (255px × 2) = 575px;

Что если сумма базовых размеров больше, чем свободного места?

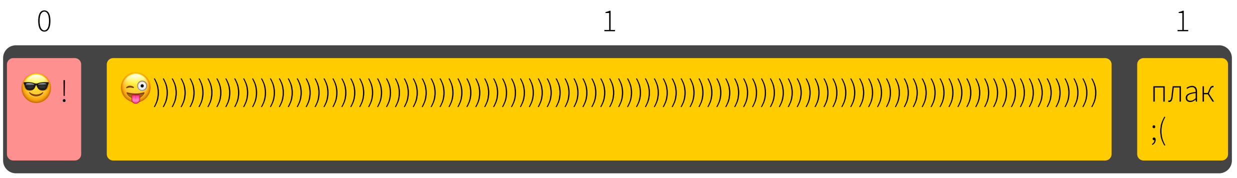
Будем делить отрицательное значение свободного места в соответствии с коэффициентами сжатия (flex-shrink).

flex-shrink: 1

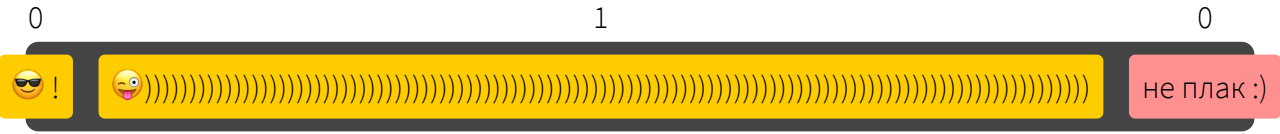
по умолчанию



flex-shrink: 0



flex-shrink: 0



Положительное свободное место — увеличиваем размеры в соответствии с flex- grow

Отрицательное свободное место — уменьшаем в соответствии с flex-shrink

Дока с алгоритмом

С помощью сокращённого свойства flex можно одновременно задать флекс-элементу flex-grow, flex-shrink и flex-basis.

flex: [flex-grow] [flex-shrink] [flex-basis]

```
.elem
{
    flex: initial; /* = 0 1 auto — все по умолчанию */









}
```

flex: [flex-grow] [flex-shrink] [flex-basis]

```
.elem
{
    flex: initial; /* = 0 1 auto — все по умолчанию */
    flex: auto;    /* = 1 1 auto */









}
```

flex: [flex-grow] [flex-shrink] [flex-basis]

```
.elem
{
    flex: initial; /* = 0 1 auto — все по умолчанию */
    flex: auto;    /* = 1 1 auto */
    flex: none;    /* = 0 0 auto */









}
```

flex: [flex-grow] [flex-shrink] [flex-basis]

```
.elem
{
    flex: initial; /* = 0 1 auto — все по умолчанию */
    flex: auto;    /* = 1 1 auto */
    flex: none;    /* = 0 0 auto */
    flex: 2;       /* число -> flex-grow,
                      = 2 1 auto */









}
```

flex: [flex-grow] [flex-shrink] [flex-basis]

```
.elem
{
    flex: initial; /* = 0 1 auto — все по умолчанию */
    flex: auto;    /* = 1 1 auto */
    flex: none;    /* = 0 0 auto */
    flex: 2;       /* число -> flex-grow,
                      = 2 1 auto */
    flex: 50%;     /* единица измерения длины -> flex-basis = 50%,
                      = 0 1 50% */




}
```

flex: [flex-grow] [flex-shrink] [flex-basis]

```
.elem
{
    flex: initial; /* = 0 1 auto – все по умолчанию */
    flex: auto;    /* = 1 1 auto */
    flex: none;    /* = 0 0 auto */
    flex: 2;       /* число -> flex-grow,
                      = 2 1 auto */
    flex: 50%;     /* единица измерения длины -> flex-basis = 50%,
                      = 0 1 50% */
    flex: 0px;     /* не то же самое, что flex: 0! */



}
```

flex: [flex-grow] [flex-shrink] [flex-basis]

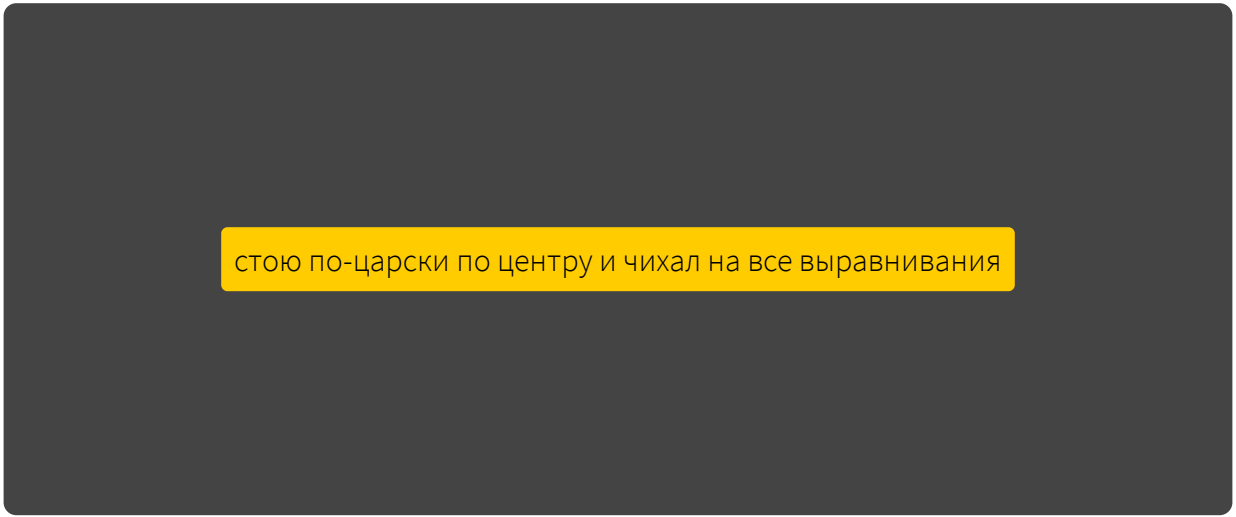
```
.elem
{
    flex: initial; /* = 0 1 auto — все по умолчанию */
    flex: auto;    /* = 1 1 auto */
    flex: none;    /* = 0 0 auto */
    flex: 2;       /* число -> flex-grow,
                      = 2 1 auto */
    flex: 50%;     /* единица измерения длины -> flex-basis = 50%,
                      = 0 1 50% */
    flex: 0px;     /* не то же самое, что flex: 0! */
    flex: 2 50%;   /* flex-grow & flex-basis,
                      = 2 1 50% */

}
```

flex: [flex-grow] [flex-shrink] [flex-basis]

```
.elem
{
    flex: initial; /* = 0 1 auto — все по умолчанию */
    flex: auto;    /* = 1 1 auto */
    flex: none;    /* = 0 0 auto */
    flex: 2;       /* число -> flex-grow,
                      = 2 1 auto */
    flex: 50%;     /* единица измерения длины -> flex-basis = 50%,
                      = 0 1 50% */
    flex: 0px;     /* не то же самое, что flex: 0! */
    flex: 2 50%;   /* flex-grow & flex-basis,
                      = 2 1 50% */
    flex: 2 0 50%;
}
```

А теперь немножко магии...

Родителю — display: flex, ребёнку — margin: auto



caniuse.com/#feat=flexbox

CSS Flexible Box Layout Module - CR

Method of positioning elements in horizontal or vertical stacks. Support includes all properties prefixed with `flex`, as well as `display: flex`, `display: inline-flex`, `align-content`, `align-items`, `align-self`, `justify-content` and `order`.

Usage	% of all users	?
Global	95.81% + 3.18% =	99%
unprefixed:	95.68% + 1.88% =	97.56%

Current aligned | Usage relative | Date relative | Apply filters | Show all | ?

IE	Edge *	Firefox *	Chrome	Safari *	Opera	iOS Safari *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baic Brows
					10-11.5										
		1 2-21	1 4-20	1 3.1-6	12.1	1 3.2-6.1									
6-9		3 22-27	21-28	6.1-8	15-16	7-8.4		1 2.1-4.3	12						
2 4 10	12-17	28-69	29-77	9-12.1	17-63	9-13.1		4.4-4.4.4	12.1				4-9.2		
4 11	18	70	78	13	64	13.2	all	76	46	78	68	12.12	10.1	1.2	7.1
	76	71-72	79-81	TP		13.3									

Notes | Sub-features (1) | Known issues (9) | Resources (13) | Feedback

Most partial support refers to supporting an older version of the specification or an older syntax.

1 Only supports the old flexbox specification and does not support wrapping.

2 Only supports the 2012 syntax

3 Does not support flex-wrap, flex-flow or align-content properties

4 Partial support is due to large amount of bugs present (see known issues)

Полезные ссылки

1. Гайд по флексбоксу
2. Flexbox Playground
3. Игра для изучения флексбокса

Перерыв

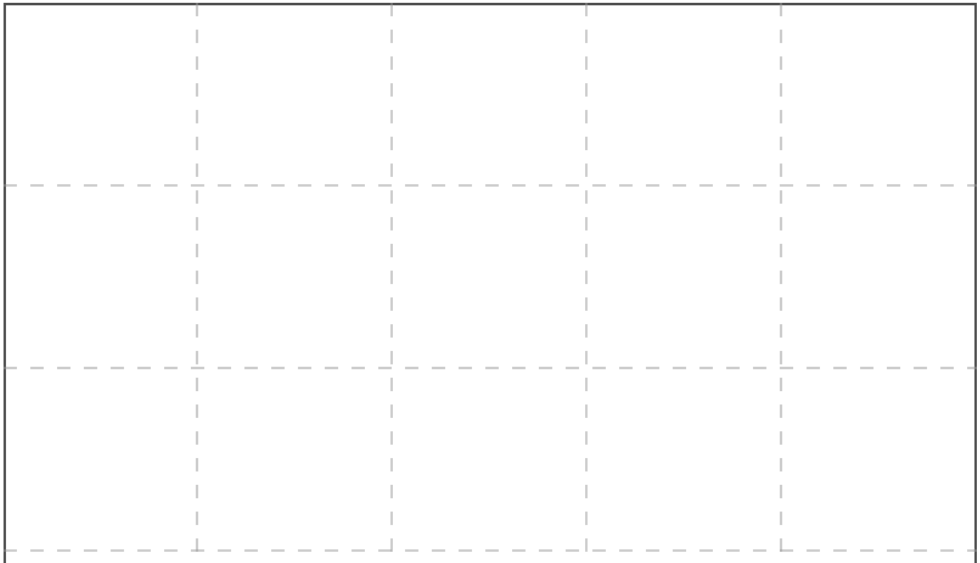
Grid

grid = сетка

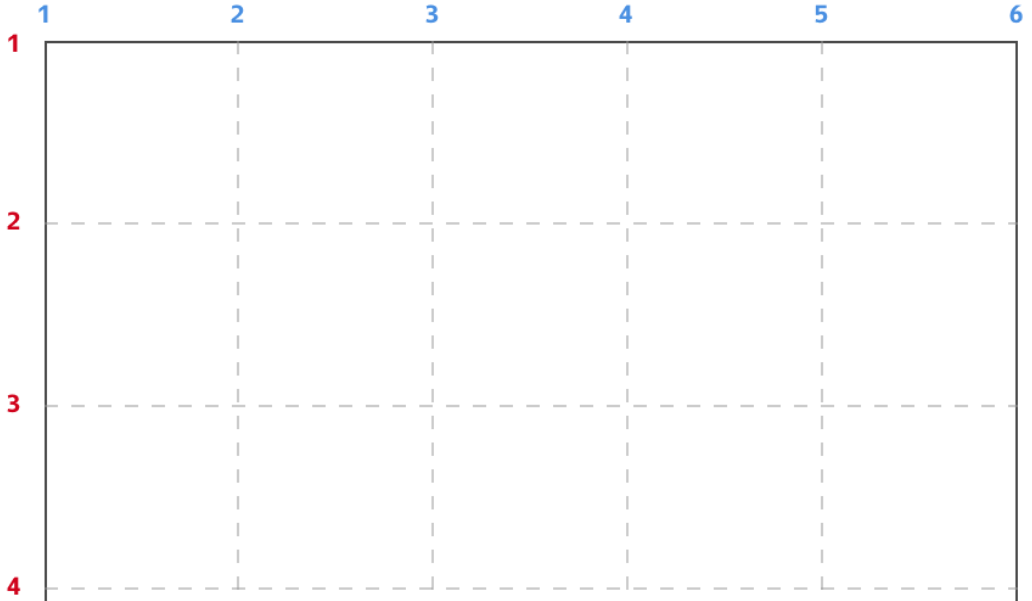
Даёт возможность располагать элементы по сетке!

display: grid

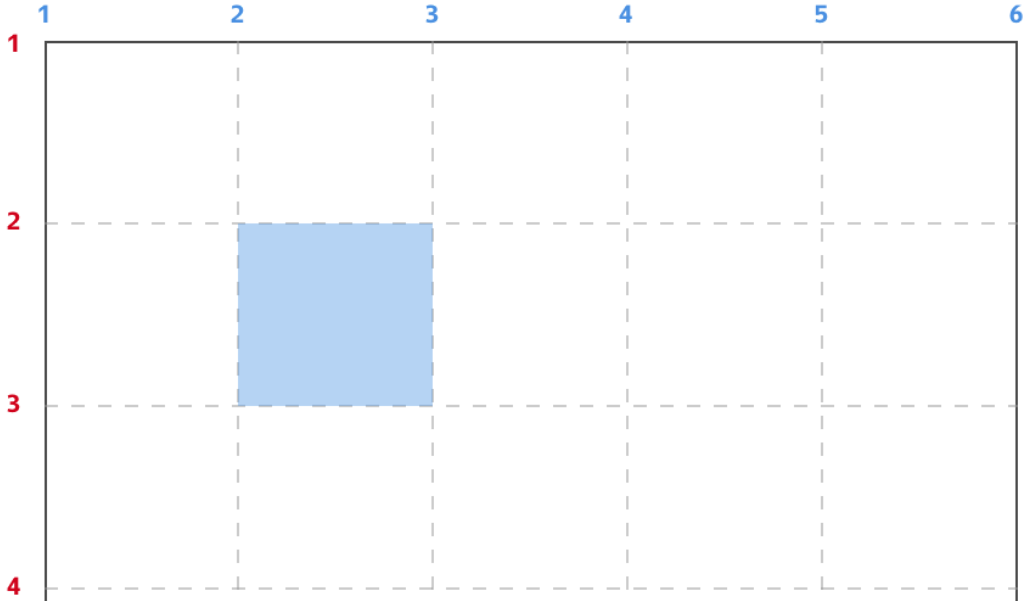
Контейнер



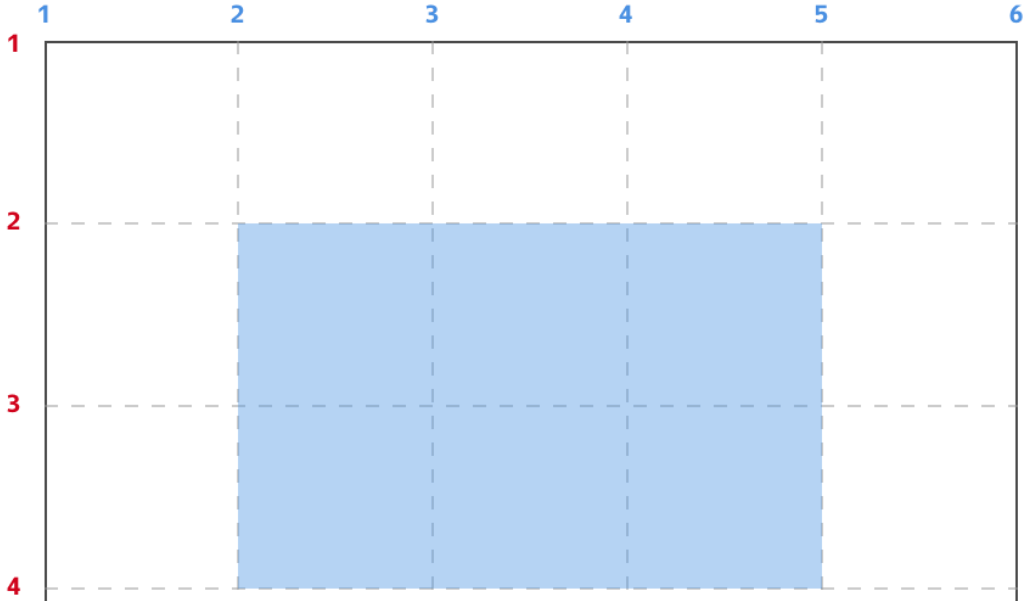
Линия



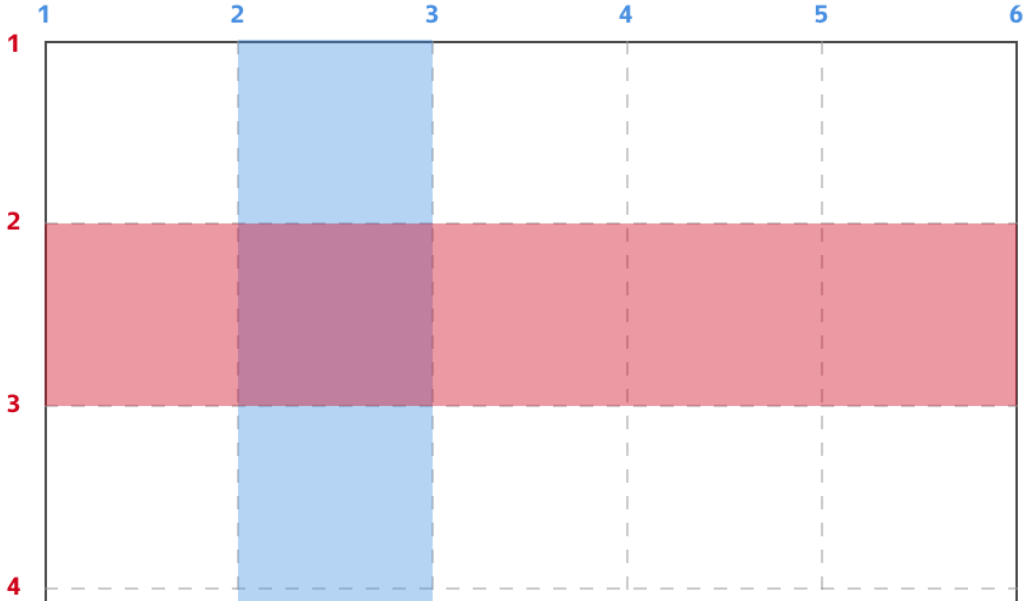
Ячейка



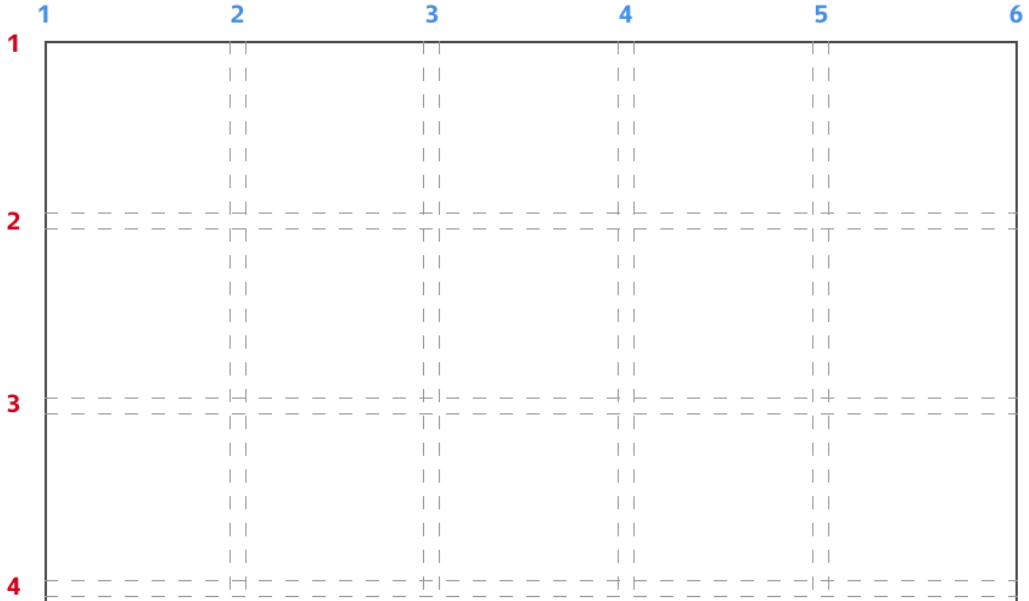
Область



Дорожка

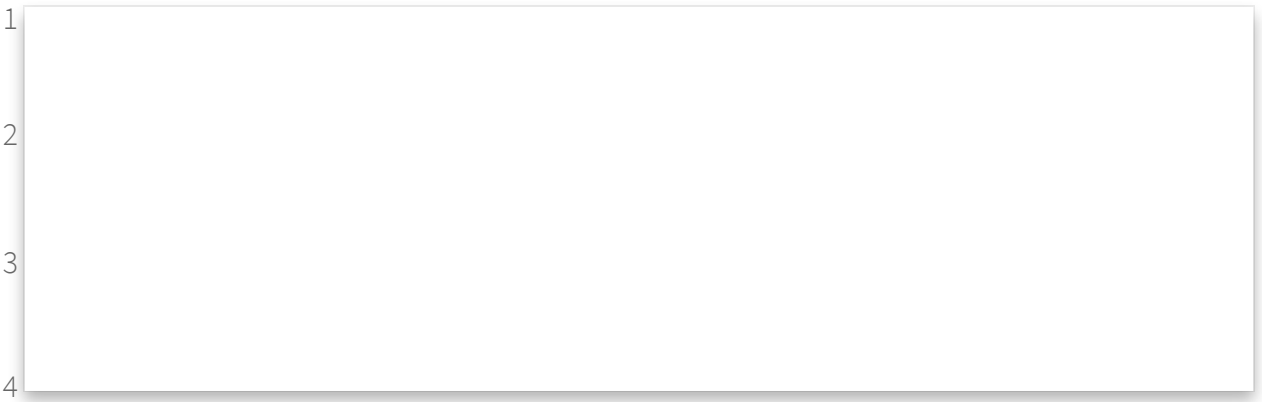


Интервал

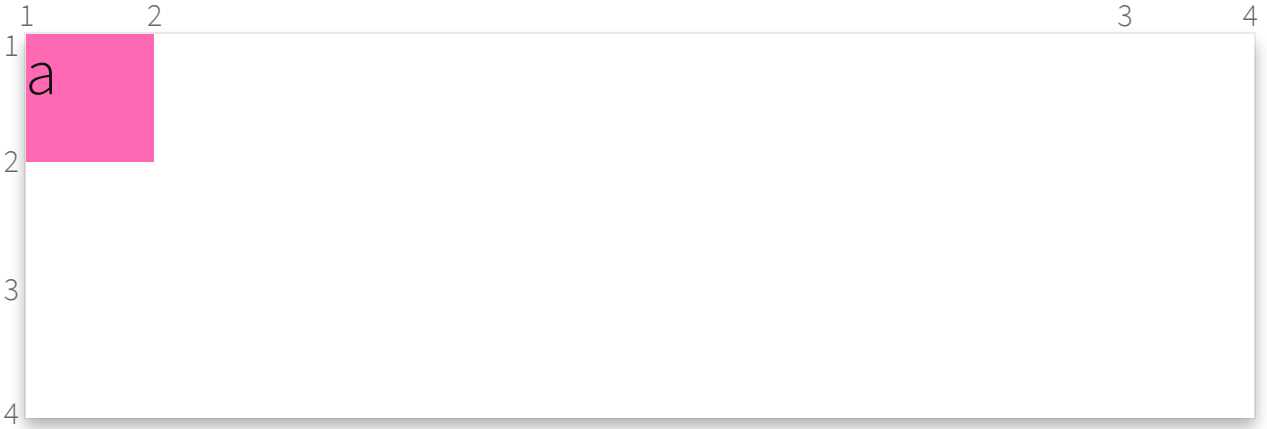


Чтобы наполнить сетку колонками и рядами существуют свойства grid-template-columns и grid-template-rows.

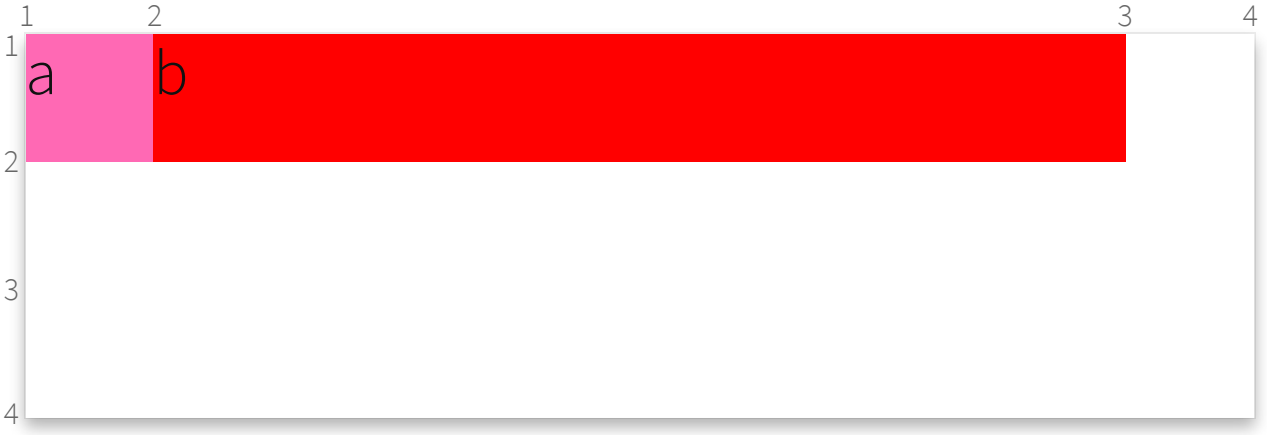
```
.container
{
    display: grid;
    grid-template-columns: 100px auto 100px; /* ширины столбцов */
    grid-template-rows: 100px 100px 100px;   /* высоты рядов    */
}
```



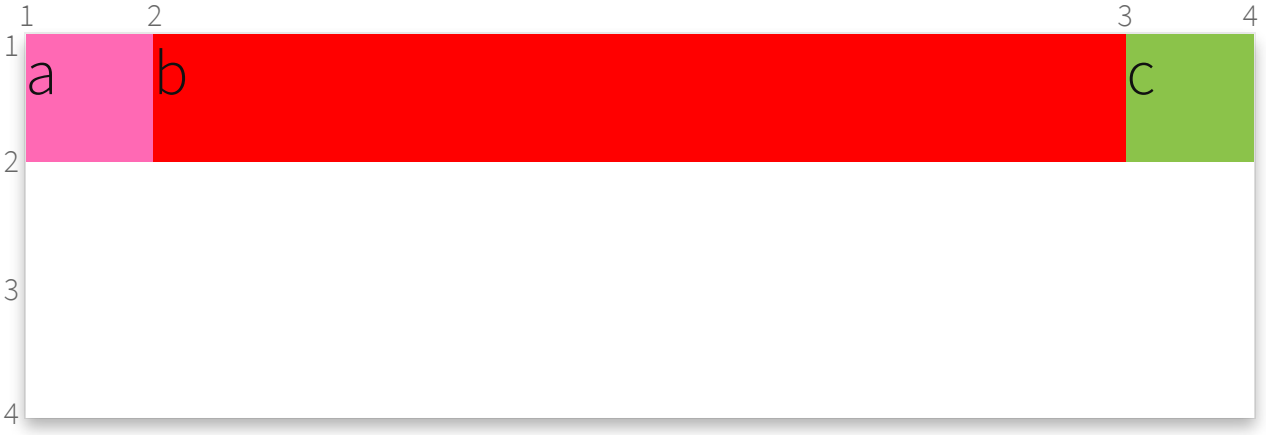
```
.container
{
    display: grid;
    grid-template-columns: 100px auto 100px; /* ширины столбцов */
    grid-template-rows: 100px 100px 100px;   /* высоты рядов    */
}
```



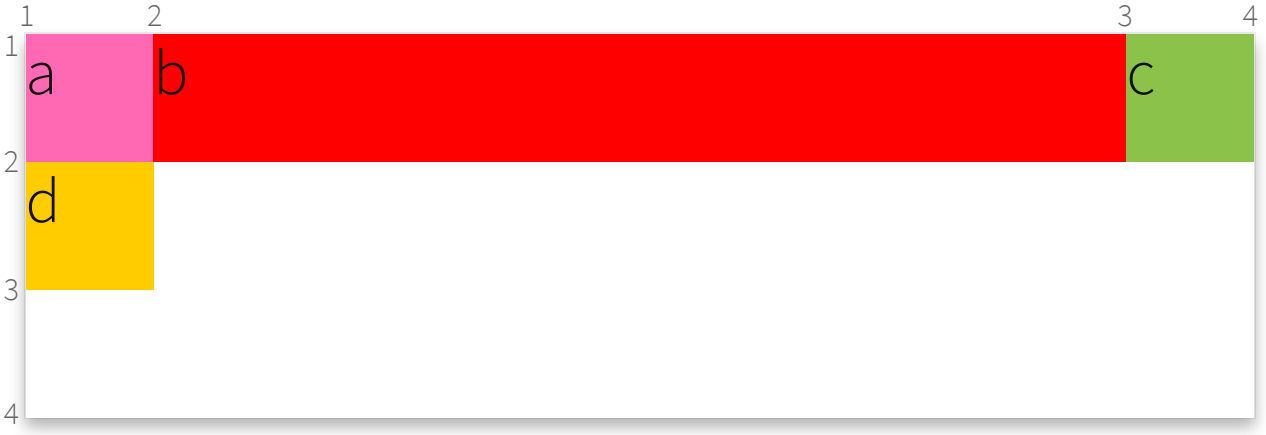
```css
.container
{
    display: grid;
    grid-template-columns: 100px auto 100px; /* ширины столбцов */
    grid-template-rows: 100px 100px 100px;   /* высоты рядов    */
}
```



```
.container
{
    display: grid;
    grid-template-columns: 100px auto 100px; /* ширины столбцов */
    grid-template-rows: 100px 100px 100px;   /* высоты рядов    */
}
```



```
.container
{
    display: grid;
    grid-template-columns: 100px auto 100px; /* ширины столбцов */
    grid-template-rows: 100px 100px 100px;   /* высоты рядов    */
}
```



```
.container
{
    display: grid;
    grid-template-columns: 100px auto 100px; /* ширины столбцов */
    grid-template-rows: 100px 100px 100px;   /* высоты рядов    */
}
```

```
.a
{
    grid-row: 1 / 2;
    grid-column: 1 / 4;
}
```

```css
.a
{
    grid-row: 1 / 2;
    grid-column: 1 / 4;
}
```

```css
.b
{
    grid-row: 2 / 4;
    grid-column: 1 / 2;
}
```

```
.a
{
    grid-row: 1 / 2;
    grid-column: 1 / 4;
}
```

```
.b
{
    grid-row: ⇨ 1 / 4;
    grid-column: 1 / 2;
}
```

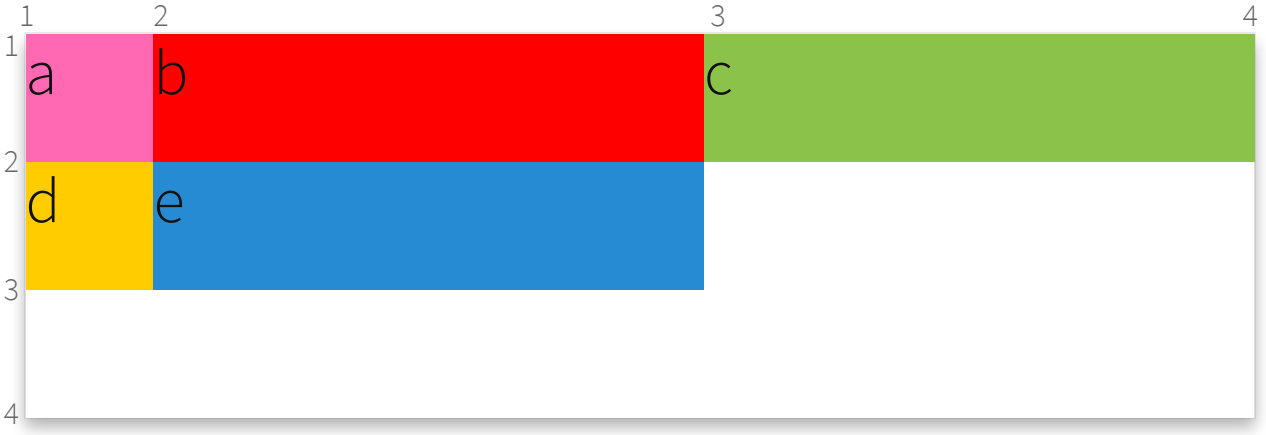
```
.a
{
    grid-row: 1 / 2;
    grid-column: 1 / 4;
    z-index: 1;
}
```

```
.b
{
    grid-row: 1 / 4;

    grid-column: 1 / 2;
}
```

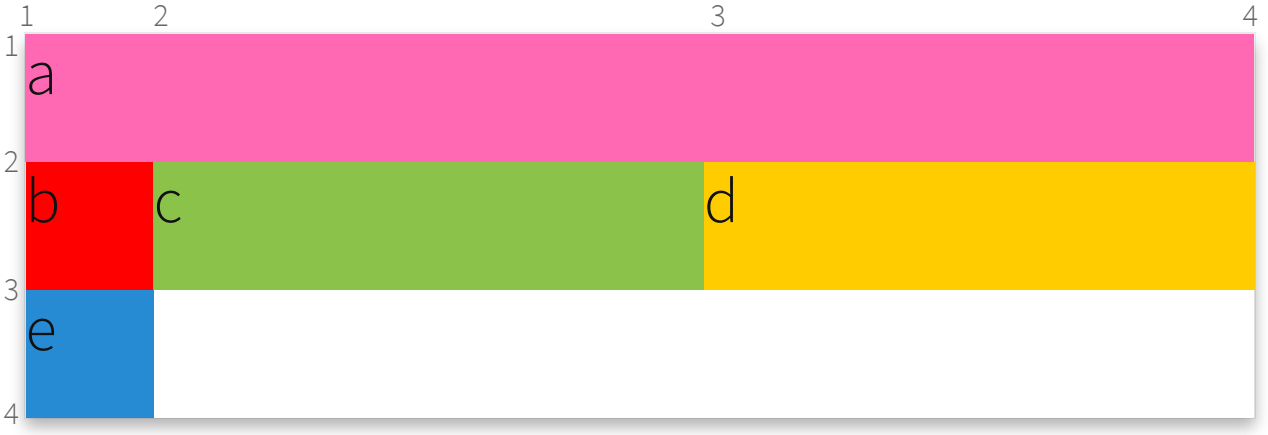
Размер дорожки можно указывать в fr – долях свободного места.

Свободное место вычисляется после расположения всех элементов фиксированных размеров.

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

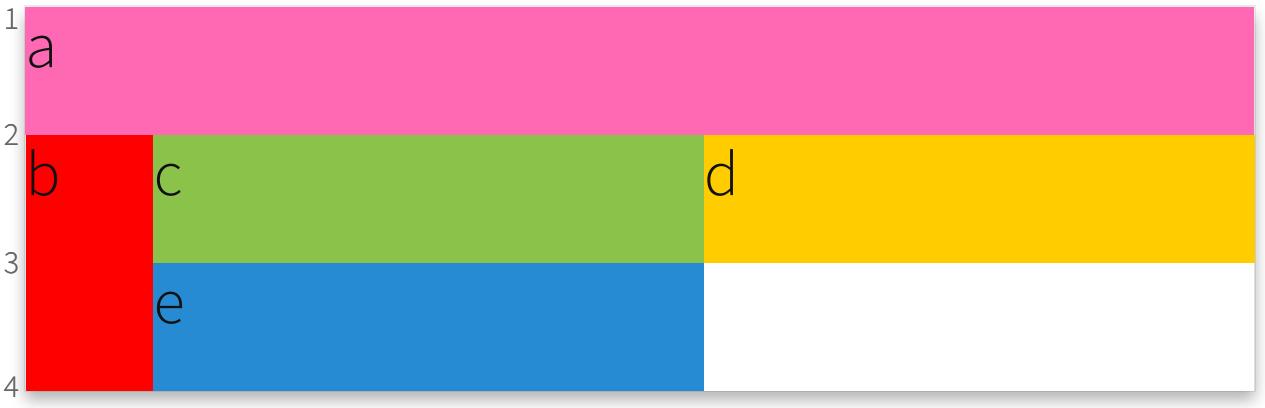


```
.a
{
    grid-column: span 3;
}
```



```
.a
{
    grid-column: span 3;
}
```

```
.b
{
    grid-row: span 2;
}
```

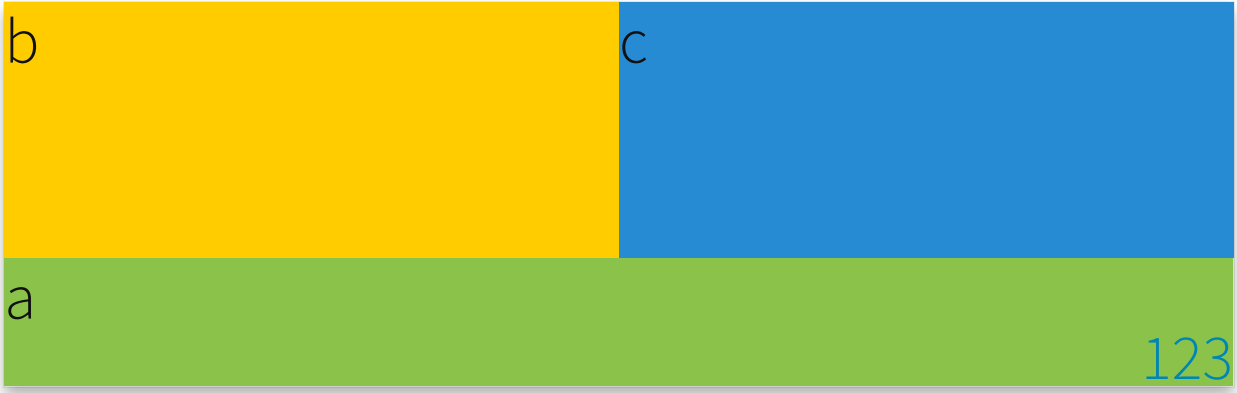


В контейнере можно создавать именованные области с помощью свойства grid-template-areas.

Поместить элемент в область можно свойством grid-area.

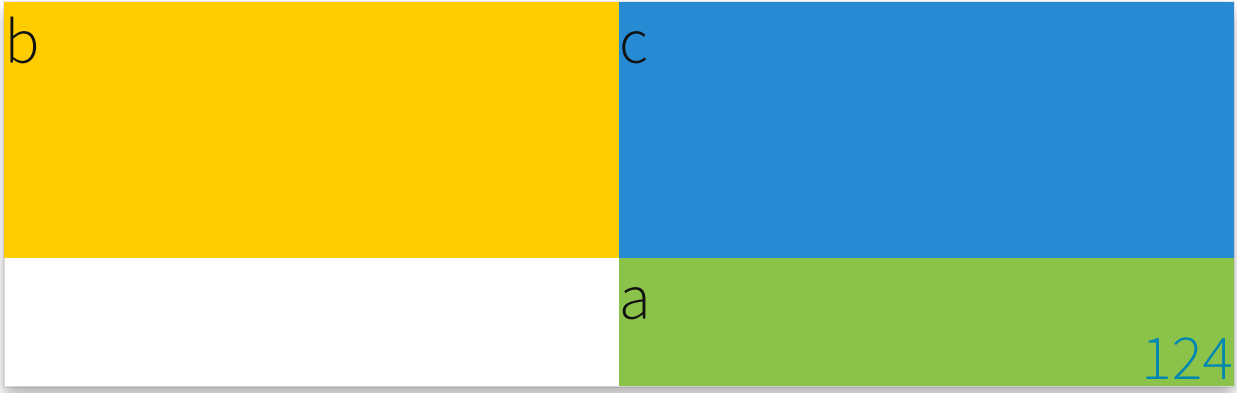
```
.beach {
    display: grid;
    grid-template-columns: 1fr 1fr;
    grid-template-rows: 100px 100px 100px;
    grid-template-areas: 'sand  water'

                         'sand  water'
                         'grass grass';
}
```

```
.a {
    grid-area: grass;
}
.b {
    grid-area: sand;

}
.c {
   grid-area: water;
}
```



```
.beach {
    display: grid;
    grid-template-columns: 1fr 1fr;
    grid-template-rows: 100px 100px 100px;
    grid-template-areas: 'sand  water'

                         'sand  water'
                         '.     grass';
}
```

```
.a {
    grid-area: grass;
}
.b {
    grid-area: sand;

}
.c {
   grid-area: water;
}
```



Короче!

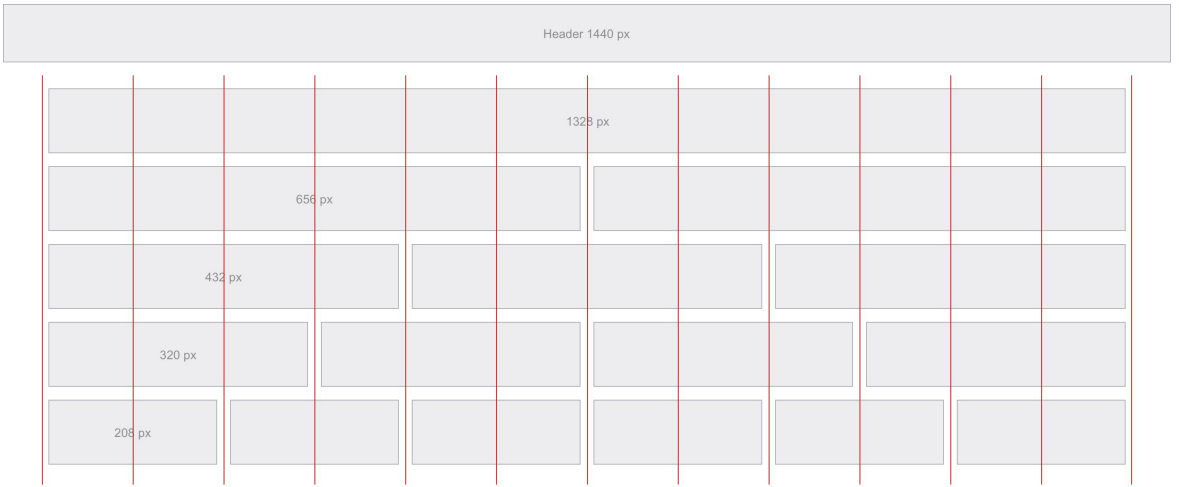
Свойство grid-template:

```
.container
{
    display: grid;
    grid-template: 'header  header'  100px
                   'sidebar content' 100px
                   'footer  content'  100px
                    / 100px auto;
}
```

Используя свойство grid-gap, можно управлять размером интервалов между линиями.

Интервал только между рядами — grid-row-gap, только между столбцами — grid-column-gap.

```
.container
{
    grid-column-gap: 1%;

    grid-row-gap: 16px;
}
```


Header 1440 px
1328 px
656 px
432 px
320 px
208 px

Писать руками?

```
.container
{
    grid-template-columns: 1fr 1fr 1fr 1fr 1fr 1fr 1fr 1fr...;
}
```

НЕТ. Есть функция repeat()

```
.container
{
    grid-template-columns: repeat(12, 1fr);
}
```

caniuse.com/#feat=css-grid

CSS Grid Layout (level 1) - CR

Method of using a grid concept to lay out content, providing a mechanism for authors to divide available space for layout into columns and rows using a set of predictable sizing behaviors. Includes support for all `grid-*` properties and the `fr` unit.

Usage: Global — % of all users: 91.89% + 1.95% = 93.84%
unprefixed: 91.89%

Current aligned | Usage relative | Date relative | Apply filters | Show all | ?

IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Opera Mobile	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baic Brows
		2-39	4-28												
		3 40-51	1 29-56		10-27										
6-9	2 12-15	4 52-53	4 57	3.1-10	1 28-43	3.2-10.2							4-5.4		
2 10	16-17	54-69	58-77	10.1-12.1	44-63	10.3-13.1		2.1-4.4.4	12-12.1				6.2-9.2		
2 11	18	70	78	13	64	13.2	all	76	46	78	68	12.12	10.1	1 1.2	7.1
	76	71-72	79-81	TP		13.3									

Notes | Known issues (3) | Resources (13) | Feedback

See also support for subgrids

1 Enabled in Chrome through the "experimental Web Platform features" flag in chrome://flags

2 Partial support in IE refers to supporting an older version of the specification.

3 Enabled in Firefox through the `layout.css.grid.enabled` flag

4 There are some bugs with overflow (1356820, 1348857, 1350925)

Полезные ссылки

1. Grid Explained In 7 Minutes
2. Гайд по гриду
3. Grid By Example
4. Игра для изучения грида


Flexbox или grid?

Flexbox

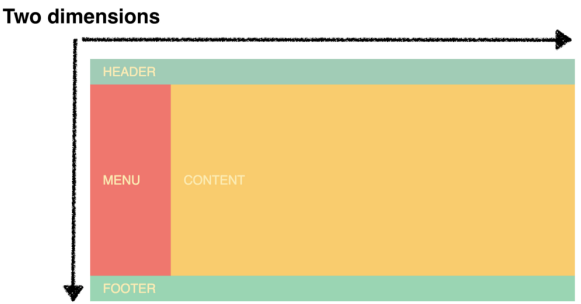
```
.container
{
    display: flex;
}

.column-i
{
    flex: auto;
}
```

Grid

```
.container
{
    display: grid;
    grid-template-columns:
                1fr 1fr 1fr;
}
```

Flexbox

Grid



Flexbox

- одно измерение
  — ряд или столбец
- content-first

Grid

- два измерения
  — ряды и столбцы
- layout-first

```
<header class="header">

    <div class="home">Home</div>
    <div class="search">Search</div>
    <div class="logout">Logout</div>
</header>
```

Flexbox

```
.header
{
    display: flex;
    align-items: center;
}
```


```
.logout
{
    margin-left: auto;
}
```

Grid

```
.header
{
    display: grid;
    grid-template-columns:
                repeat(10, 1fr);
}
```


```
.logout
{
    grid-column: 10;
}
```

Подходы можно совмещать

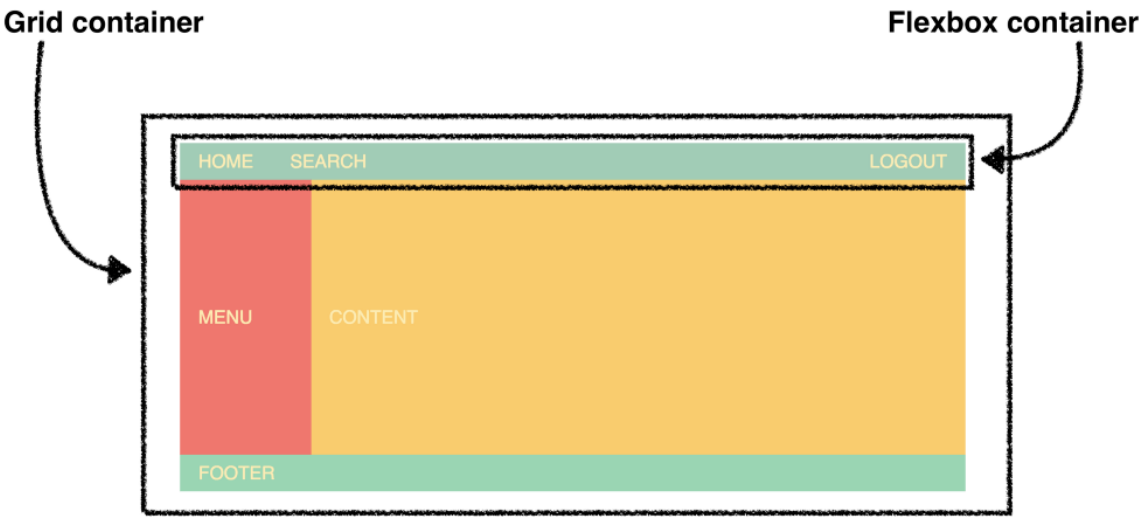
Подходы можно совмещать

```
<div class="container">
  <header class="header">HEADER</header>
  <aside class="menu">MENU</aside>

  <main class="content">CONTENT</main>
  <footer class="footer">FOOTER</footer>
</div>
```

Подходы можно совмещать

```
.container
{
    display: grid;
    grid-template-columns: repeat(12, 1fr);

    grid-template-rows: 50px 350px 50px;
}

.header
{
    display: flex;
    align-items: center;
}
.logout
{
    margin-left: auto;
}
```

```
.header,
.footer
{
    grid-column: span 12;

}
.menu
{
    grid-column: span 2;
}
.content
{
    grid-column: span 10;
}
```


Grid container
Flexbox container
HOME
SEARCH
LOGOUT
MENU
CONTENT
FOOTER

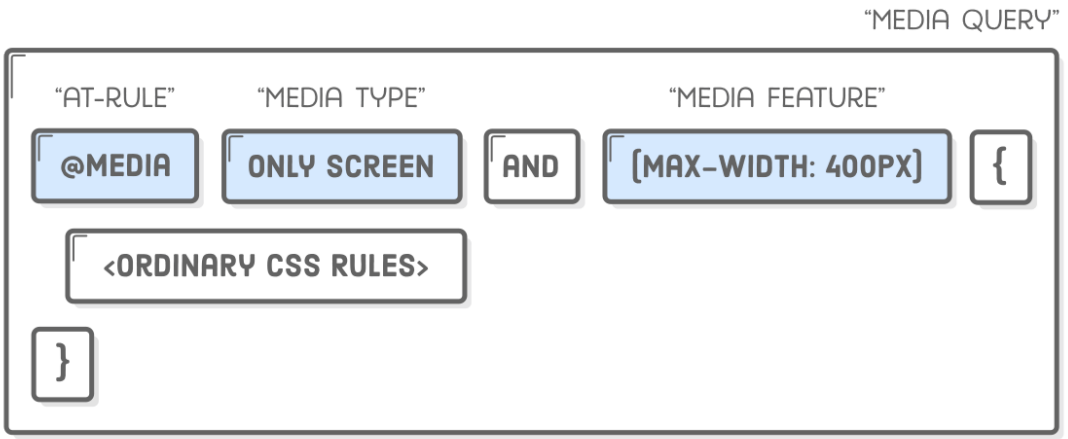
Адаптивный дизайн

Медиа-выражения –

условное применение CSS-правил.

Одна разметка, разные наборы стилей.

Задаются с помощью at-правила @media, за которым следует условие:



Media Type задает тип устройства

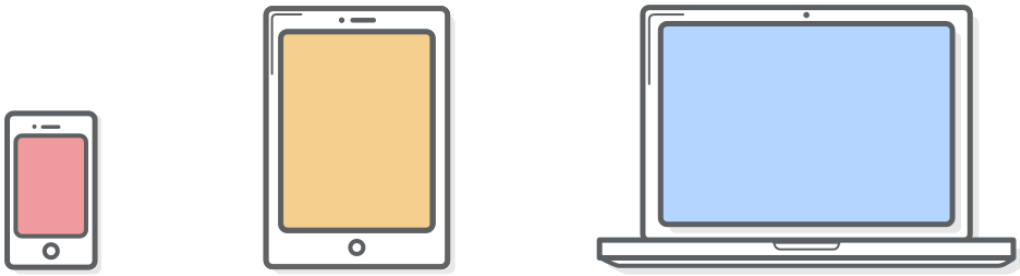
- all – все устройства (по умолчанию)
- print — принтеры и режим предпросмотра
- screen — устройства с экраном
- speech — скринридеры

Media Features задают технические характеристики устройства

- width – ширина вьюпорта
- height — высота вьюпорта
- orientation — ориентация вьюпорта
- resolution — разрешение устройства вывода

Ширина вьюпорта

```
/* для мобильных */
body {
    background: red;
}


/* для планшетов */
@media screen and (min-width: 768px) {
    body {
        background: yellow;
    }
}

/* для десктопов */
@media screen and (min-width: 1280px) {
    body {
        background: blue;
    }
}
```


MOBILE
TABLET
DESKTOP

Полезные ссылки

1. @media на MDN
2. Responsive Design Tutorial

Итоги

- Для раскладки одномерных элементов удобно применять flex
- Для двумерных — grid
- Совмещая эти подходы, можно сверстать всё, что угодно
- И сделать адаптивно с помощью media-запросов

Спасибо!