
МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

# Фронтенд и бэкенд разработка

## Flexbox, Grid

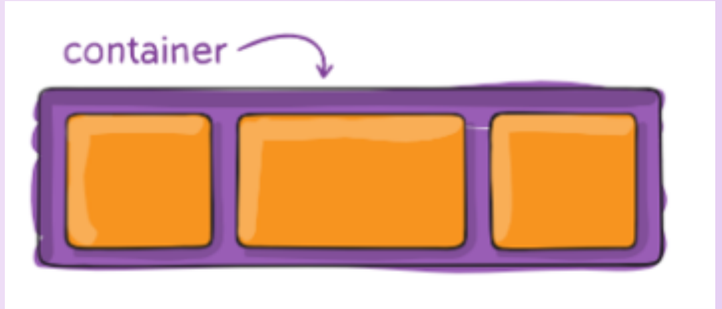
# Flexbox

Гибкая компоновка CSS, известная как Flexbox, представляет собой модель веб-макета CSS3. Гибкая компоновка позволяет автоматически упорядочивать адаптивные элементы внутри контейнера в зависимости от размера области просмотра (экрана устройства).

Долгое время единственными надёжными инструментами CSS вёрстки были такие способы как Float (обтекание) и позиционирование. С их помощью сложно или невозможно достичь следующих простых требований к макету:

- Вертикального выравнивания блока внутри родителя.
- Оформления всех детей контейнера так, чтобы они распределили между собой доступную ширину/высоту, независимо от того, сколько ширины/высоты доступно.
- Сделать все колонки в макете одинаковой высоты, даже если наполнение в них различно.

# Flexbox

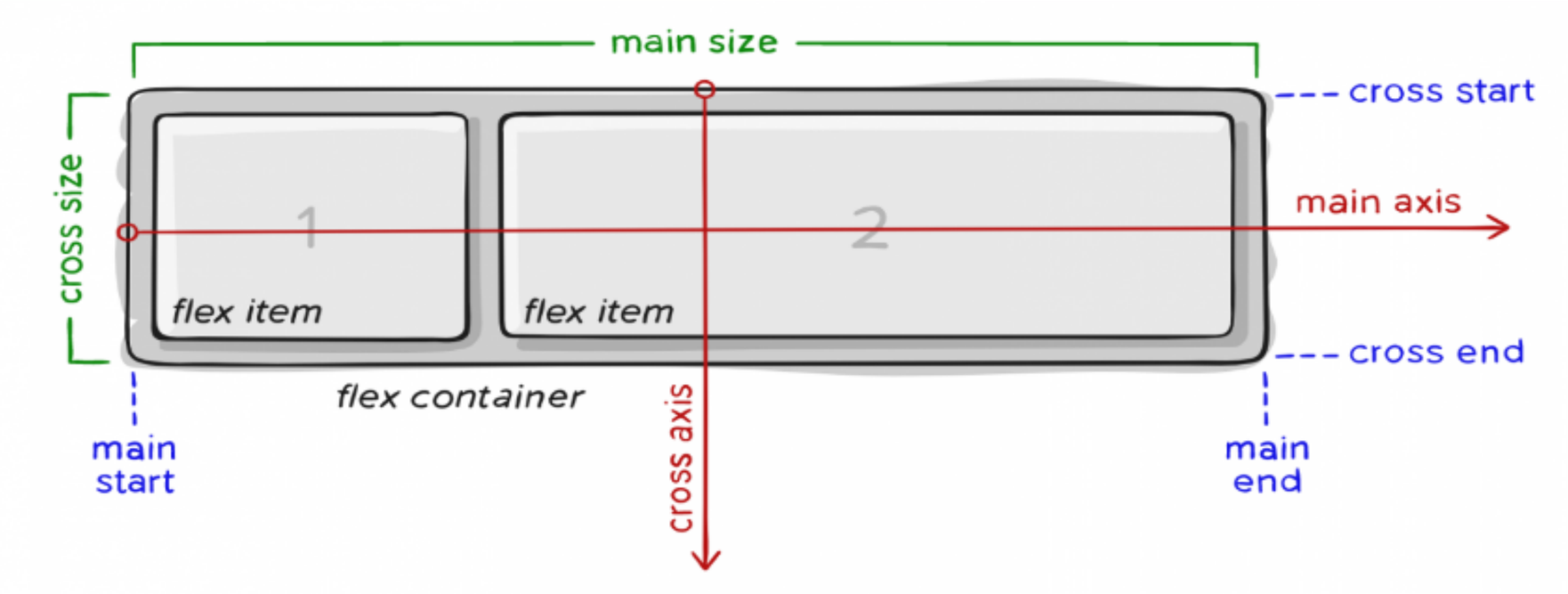



Flexbox разметка в CSS даёт один из наиболее эффективных способов расстановки, выравнивания и распределения места между элементами внутри контейнера, даже если их размер неизвестен или динамичен (собственно, по этому его и называют flex, от слова flexible, что по-английски имеет двойственное значение – гибкий и уступчивый, что очень сочетается с моделью поведения flexbox).

Основной целью Flexbox является предоставление возможности изменения своих элементов по ширине и высоте, для того, чтобы они максимально эффективно умещались в доступном месте родительского контейнера, в частности – это удобно в тех случаях, когда нужно соответствовать всем типам дисплеев устройств и размерам экранов. Flex контейнер расширяет вложенные элементы для того, чтобы заполнить доступное пространство или же урезает их, чтобы избежать переполнения.

# Flexbox

Поскольку flexbox – это целый модуль, а не одно свойство, он включает в себя множество элементов с набором свойств. Некоторые из них предназначены для установки в контейнере (родительский элемент принято называть «flex контейнер»), в то время как другие предназначены для установки в дочерних элементах (так называемые «flex элементы»).



# Flexbox

Элементы будут расположены либо в направлении главной оси (**main axis** от **main-start** до **main-end**) или в направлении поперечной оси (**cross axis** от **cross-start** до **cross-end**).
**main axis** – главная ось flex контейнера – это основная ось, вдоль которой располагаются flex элементы. Будьте внимательны, эта ось не обязательна горизонтальная; это зависит от **flex-direction** свойства (см. ниже).

**main-start | main-end** – flex элементы помещаются в контейнер, начиная с main-start и заканчивая main-end.

**main size** – ширина или высота flex элемента, в зависимости от того, что находится в основном измерении. Определяется основным размером flex элементов т.е. свойством 'width' или 'height', в зависимости от того, что находится в основном измерении.

**cross axis** – ось перпендикулярная главной оси, называется поперечной осью. Её направление зависит от направления главной оси.

**cross-start | cross-end** – flex строки заполняются элементами и помещаются в контейнер, начиная от cross-start flex контейнера по направлению к cross-end.

**cross size** – ширина или высота flex элемента. В зависимости от css свойства flex-direction, это ширина или высота элемента. Это всегда поперечный размер flex элементов.

# Flex-direction

В Flexbox есть свойство под названием **flex-direction**, которое определяет направление главной оси (в каком направлении располагаются flexbox-дочерние элементы) – по умолчанию ему присваивается значение row, т.е. располагать дочерние элементы в ряд слева направо (для большинства языков) или справа налево (для арабских языков)

Если добавить следующее CSS-свойство flex-direction: column, то мы увидим что элементы расположились в виде столбцов, также как было до того, как мы добавили CSS код.

# Flex-direction

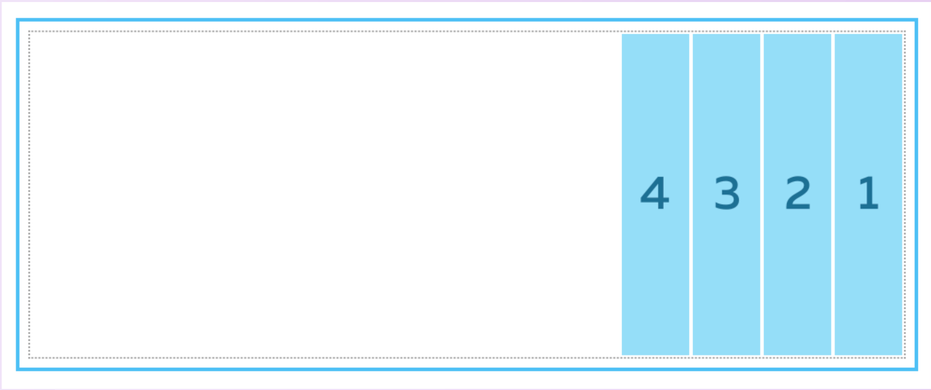
**row**

Главная ось направлена так же, как и ориентация текста, по умолчанию слева направо. Если значение dir задано как rtl, то направление оси идёт справа налево.


```
.parent {
  display: flex;
  flex-direction: row;
  height: 100%;
}
```

**row-reverse**

Похоже на значение row, но меняются местами начальная и конечная точки и главная ось направлена справа налево. Если значение dir задано как rtl, то направление оси идёт слева направо.

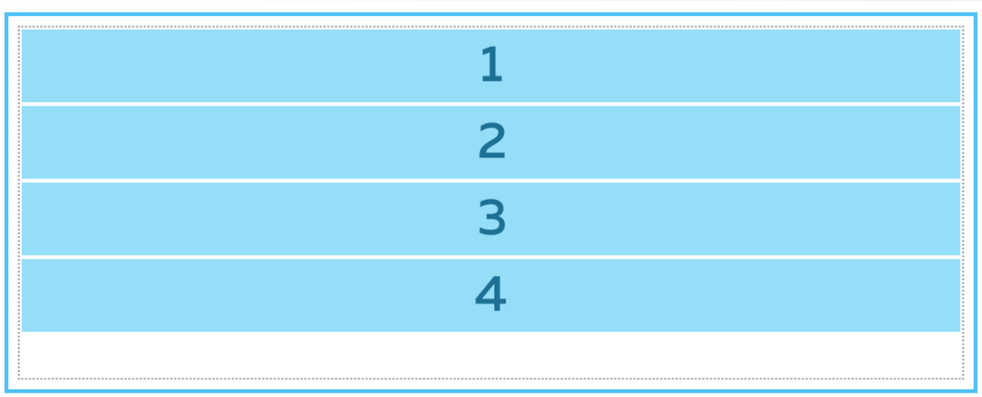



```
.parent {
  display: flex;
  flex-direction: row-reverse;
  height: 100%;
}
```

# Flex-direction

**column**

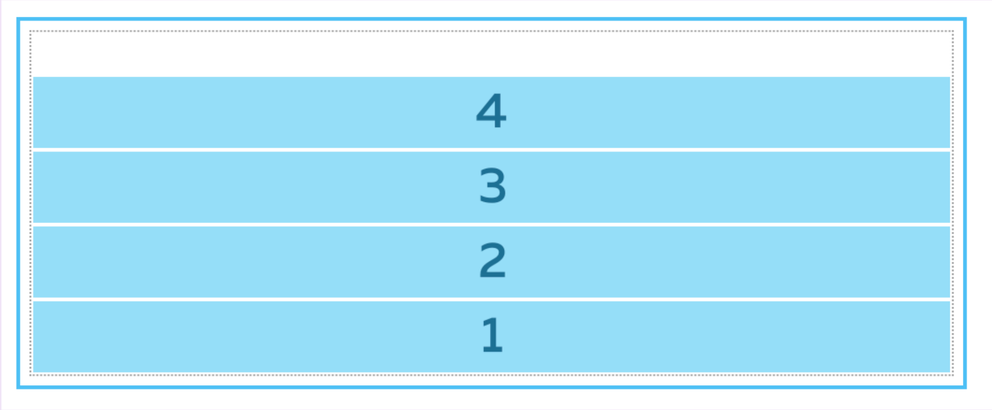
Главная ось располагается вертикально и направлена сверху вниз.




```
.parent {
  display: flex;
  flex-direction: column;
  height: 100%;
}
```

**column-reverse**

Главная ось располагается вертикально, но меняется положение начальной и конечной точек и ось направлена снизу вверх.




```
.parent {
  display: flex;
  flex-direction: column-reverse;
  height: 100%;
}
```

# Parent & Child

**:first-child, :last-child и :nth-child(n)**

Эти псевдоклассы выбирают элемент по его порядковому номеру. :first-child соответствует первому дочернему элементу родителя, :last-child – последнему. А псевдокласс :nth-child(n) указывает на n-й дочерний элемент. Например, с его помощью можно выбрать второй, пятый или предпоследний элемент. Вместо n в скобках указывается целое число или математическое выражение

```
<ul>
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ul>
```

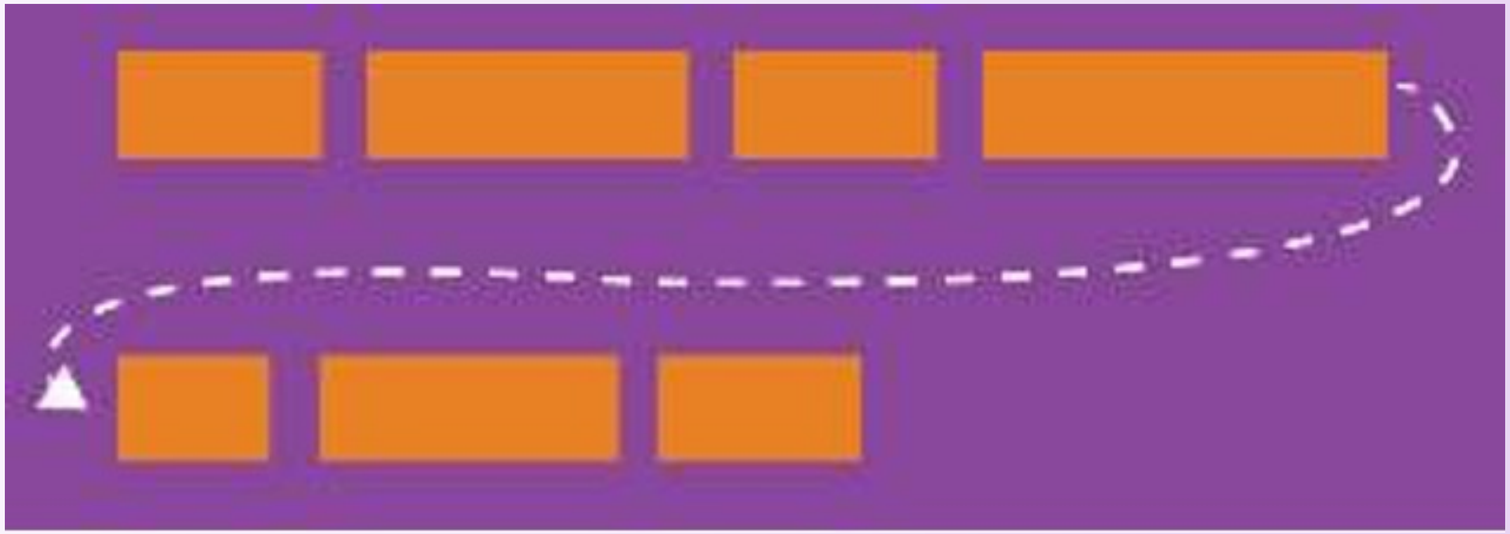
```
/* выберет первый элемент – HTML*/
li:first-child {
  font-weight: 700;
}

/* выберет последний элемент – JavaScript*/
li:last-child {
  text-decoration: underline;
}

/* выберет второй элемент – CSS*/
li:nth-child(2) {
  font-weight: 700;
}
```

# Flex-wrap = перенос строк внутри контейнера

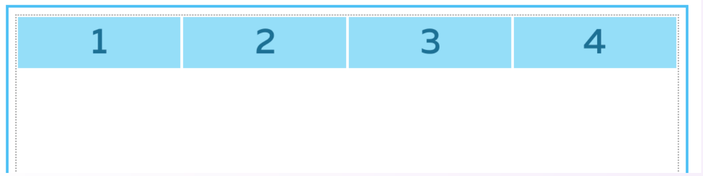
Свойство flex-wrap Указывает, следует ли флексам располагаться в одну строку или можно занять несколько строк. Если перенос строк допускается, то свойство также позволяет контролировать направление, в котором выкладываются строки.
**Применяется к**: flex контейнерам.
**Значение по умолчанию**: nowrap



# Flex-wrap = перенос строк внутри контейнер

**nowrap**

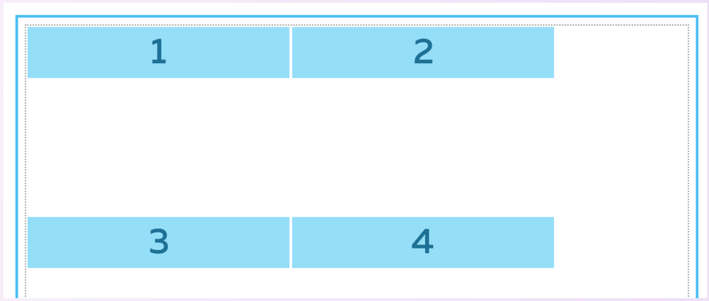
Флексы выстраиваются в одну линию.




```
.parent {
  display: flex;
  align-items: flex-start;
  flex-wrap: nowrap;
  height: 100%;
}
.child {
  width: 40%;
}
```

**wrap**

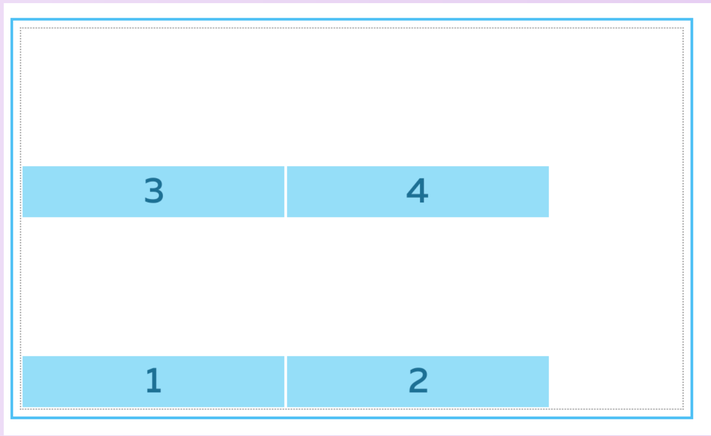
Флексы выстраиваются в несколько строк, их направление задаётся свойством flex-direction.




```
flex-wrap: wrap;
```

**wrap-reverse**

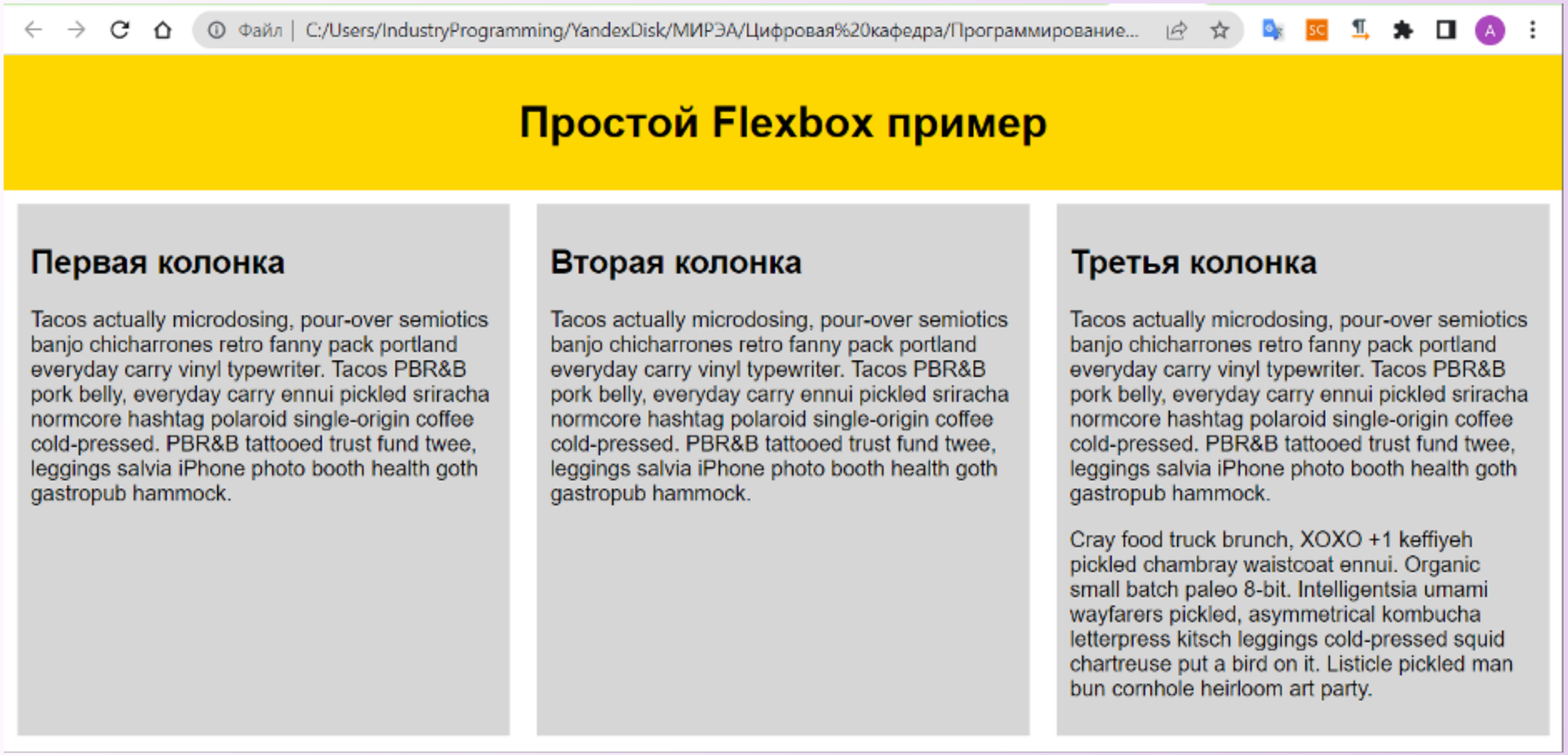
Флексы выстраиваются в несколько строк, в направлении, противоположном flex-direction.




```
flex-wrap: wrap-reverse;
```

# Пример 1.1

Создать простой пример трехколоночного сайта с использованием Flexbox



# Пример 1.2

Настроить перенос строк с помощью flex-wrap

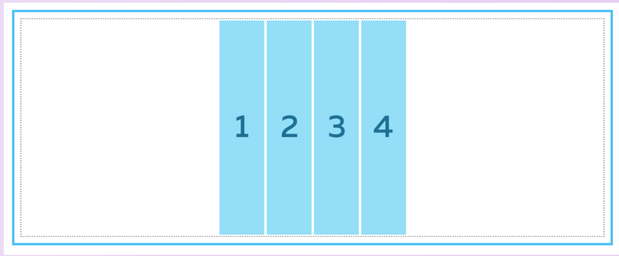

```css
section{
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
}

article {
  flex: 200px;
}
```

# Justify-content

Свойство justify-content определяет, как браузер распределяет пространство вокруг флекс-элементов вдоль главной оси контейнера. Это делается после того, как применяются размеры и автоматические отступы, за исключением ситуации, когда по крайней мере у одного элемента flex-grow больше нуля. При этом не остаётся никакого свободного пространства для манипулирования.

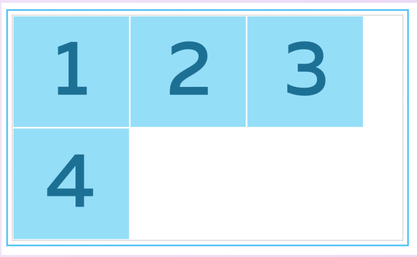
**Применяется к**: flex контейнерам.
**Значение по умолчанию**: flex-start.

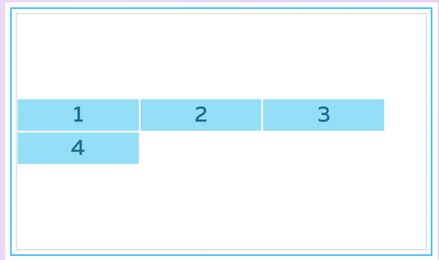




# Align-content

Свойство align-content задаёт тип выравнивания строк внутри flex контейнера по поперечной оси при наличии свободного пространства.

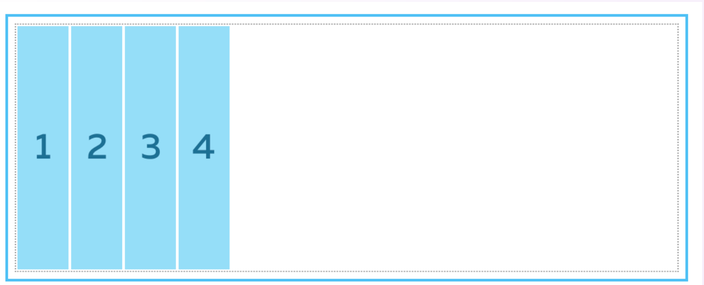
**Применяется к**: flex контейнерам.
**Значение по умолчанию**: stretch.



# Justify-content

**flex-start**
Флексы прижаты к началу строки.

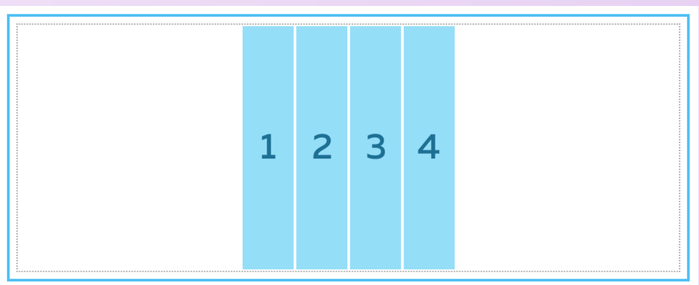



```
.parent {
  display: flex;
  justify-content: flex-start;
  height: 100%;
}
```

**flex-end**
Флексы прижаты к концу строки.


```
.parent {
  display: flex;
  justify-content: flex-end;
  height: 100%;
}
```

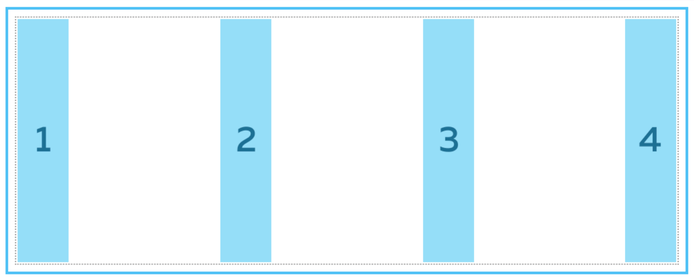
**center**
Флексы выравниваются по центру строки.




```
.parent {
  display: flex;
  justify-content: center;
  height: 100%;
}
```

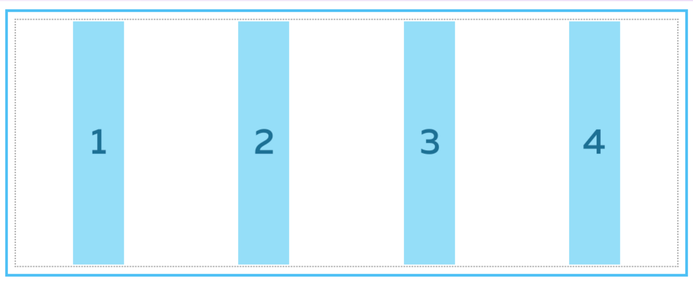
# Justify-content

**space-between**
Флексы равномерно распределяются по всей строке. Первый и последний элемент прижимаются к соответствующим краям контейнера.



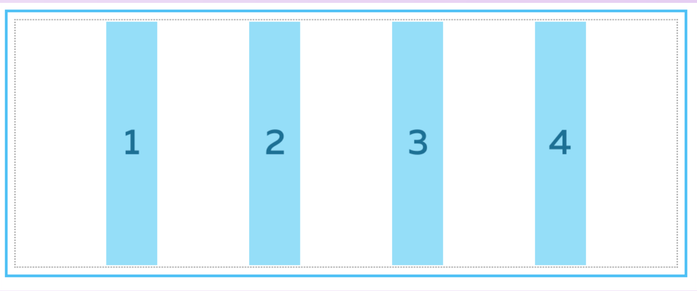

```
.parent {
  display: flex;
  justify-content: space-between;
  height: 100%;
}
```

**space-around**
Флексы равномерно распределяются по всей строке. Пустое пространство перед первым и после последнего элементов равно половине пространства между двумя соседними элементами.




```
.parent {
  display: flex;
  justify-content: space-around;
  height: 100%;
}
```

**space-evenly**
Флексы распределяются так, что расстояние между любыми двумя соседними элементами, а также перед первым и после последнего, было одинаковым.




```
.parent {
  display: flex;
  justify-content: space-evenly;
  height: 100%;
}
```

# Align-items

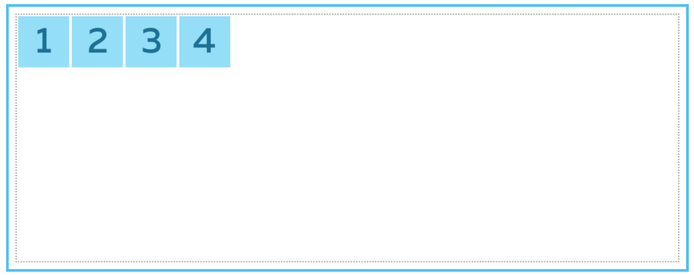
Свойство align-items выравнивает флекс-элементы внутри контейнера в перпендикулярном направлении.

**Применяется к**: flex контейнерам.
**Значение по умолчанию**: stretch.

# Align-items

**flex-start**

Флексы выравниваются в начале поперечной оси контейнера.




```
.parent {
  display: flex;
  align-items: flex-start;
  height: 100%;
}
```

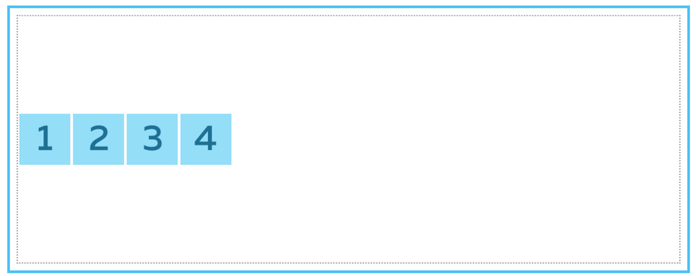
**flex-end**

Флексы выравниваются в конце поперечной оси контейнера.


```
.parent {
  display: flex;
  align-items: flex-end;
  height: 100%;
}
```

**center**

Флексы выравниваются по линии поперечной оси.

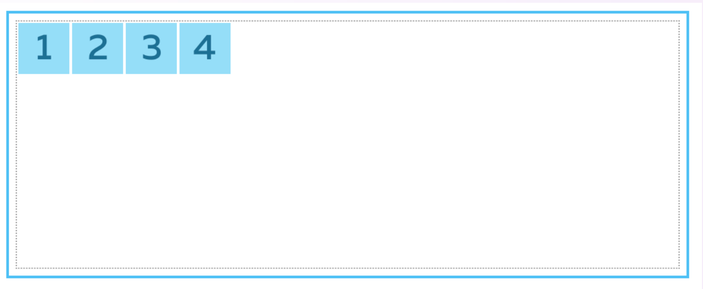



```
.parent {
  display: flex;
  align-items: center;
  height: 100%;
}
```

# Align-items

**baseline**

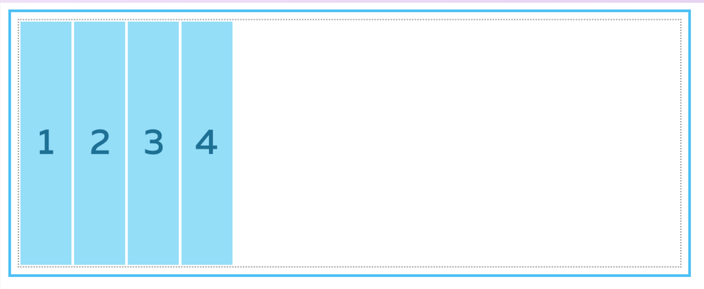
Флексы выравниваются по их базовой линии.




```
.parent {
  display: flex;
  align-items: baseline;
  height: 100%;
}
```

**Stretch**

Флексы растягиваются таким образом, чтобы занять всё доступное пространство контейнера.




```
.parent {
  display: flex;
  align-items: stretch;
  height: 100%;
}
```

# Flexbox

Шпаргалка: https://tpverstak.ru/flex-cheatsheet/

# Адаптив на основе Flexbox

**@-правила CSS (at-правила, директивы CSS)**
Директивы – это конструкции, которая позволяет создавать в CSS инструкции для изменения отображения либо поведения элементов страницы. Директива начинается со знака @, за которым следует одно из служебных слов. Это общий синтаксис, которому следуют все директивы.

**@import**
При помощи директивы **@import** можно импортировать один файл со стилями в другой файл со стилями.

```
1 @import "fonts.css";
2 @import "buttons.css";
3
4 /*  Остальной CSS-код */
```

При этом равнозначны:

```
1  @import "file.css";
2  @import url("file.css");
```

# Адаптив на основе Flexbox

Можно указать, для каких типов устройств должны применяться стили из импортируемого файла:

```
1 @import "print-styles.css" print;
2 @import "screen-styles.css" screen;
```

CSS поддерживает следующие виды устройств вывода:

| Тип | Описание |
|---|---|
| all | Все типы. Это значение используется по умолчанию. |
| aural | Речевые синтезаторы, а также программы для воспроизведения текста вслух. Сюда, например, можно отнести речевые браузеры. |
| braille | Устройства, основанные на системе Брайля, которые предназначены для слепых людей. |
| handheld | Наладонные компьютеры и аналогичные им аппараты. |
| print | Печатающие устройства вроде принтера. |
| projection | Проектор. |
| screen | Экран монитора. |
| tv | Телевизор. |

# Адаптив на основе Flexbox

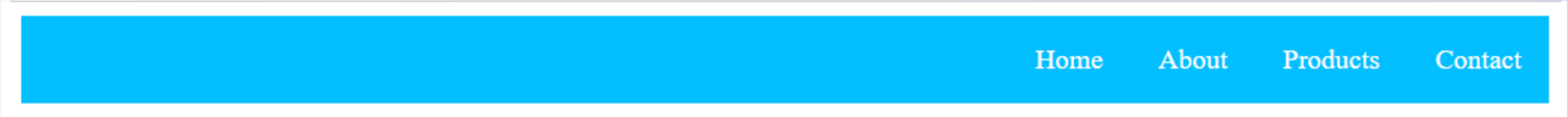
Подключение/импорт стороннего шрифта:

Пример 3.2 > # style.css > ...

```
1   @font-face {
2     font-family: "Novartis";
3     src: url("Novartis-Deco.ttf");
4     font-weight: normal;
5   }
6
7   p {
8     font-family: "Novartis", sans-serif;
9     font-size: 30px;
10  }
```

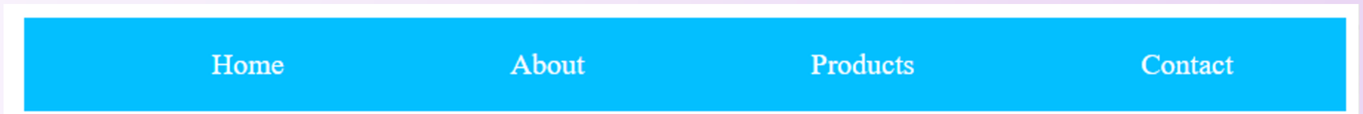
# Пример: меню для трех версий утройств

Исходное меню:




Добавляем код:
<= 800px

```
22  @media all and (max-width: 800px) {
23    .navigation {
24      justify-content: space-around;
25    }
26  }
```



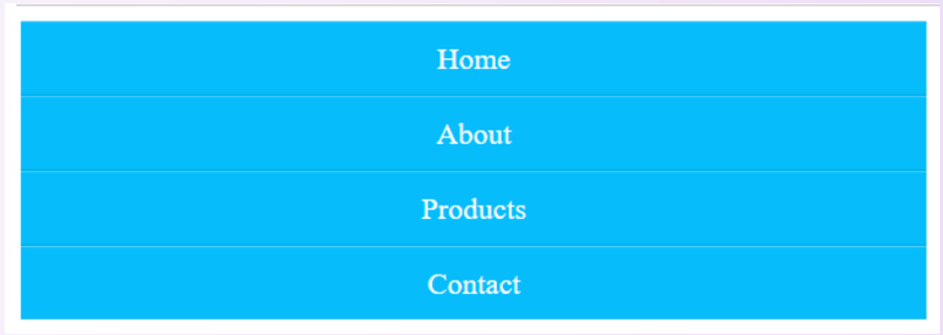

Это медиа-запрос. Внутри него прописываются стилевые правила (строки 23-25), которые срабатывают, если устройство для просмотра страницы удовлетворяет параметрам, описанным в строке 22. Первый параметр – это тип устройства (в данном случае all означает, что правила будут применяться на всех устройствах). Второй параметр означает максимальную ширину области просмотра (ширина окна браузера). **max-width: 800px;** означает, что стилевые правила будут срабатывать при ширине окна браузера не более 800px.

# Пример: меню для трех версий утройств

Добавляем код:
<= 600px

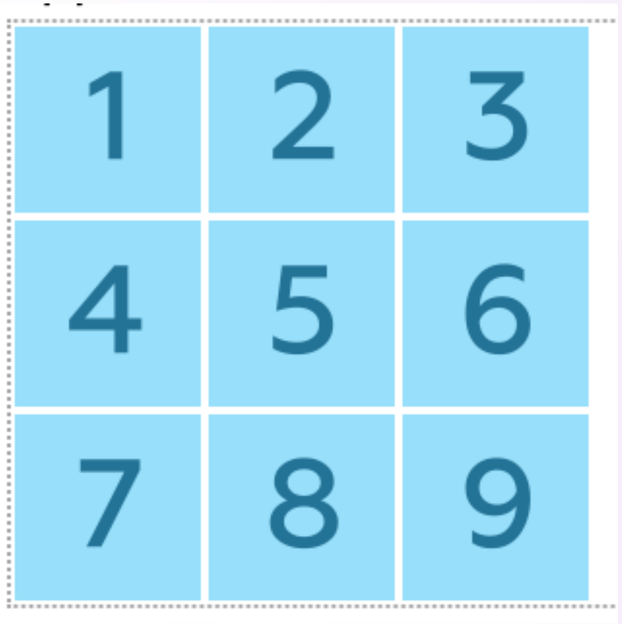
```css
@media all and (max-width: 600px) {
  .navigation {
    flex-flow: column wrap;
    padding: 0;
  }
  .navigation a {
    text-align: center;
    padding: 10px;
    border-top: 1px solid rgba(255, 255, 255,0.3);
    border-bottom: 1px solid rgba(0, 0, 0, 0.1);
  }
  .navigation li:last-of-type a {
    border-bottom: none;
  }
}
```

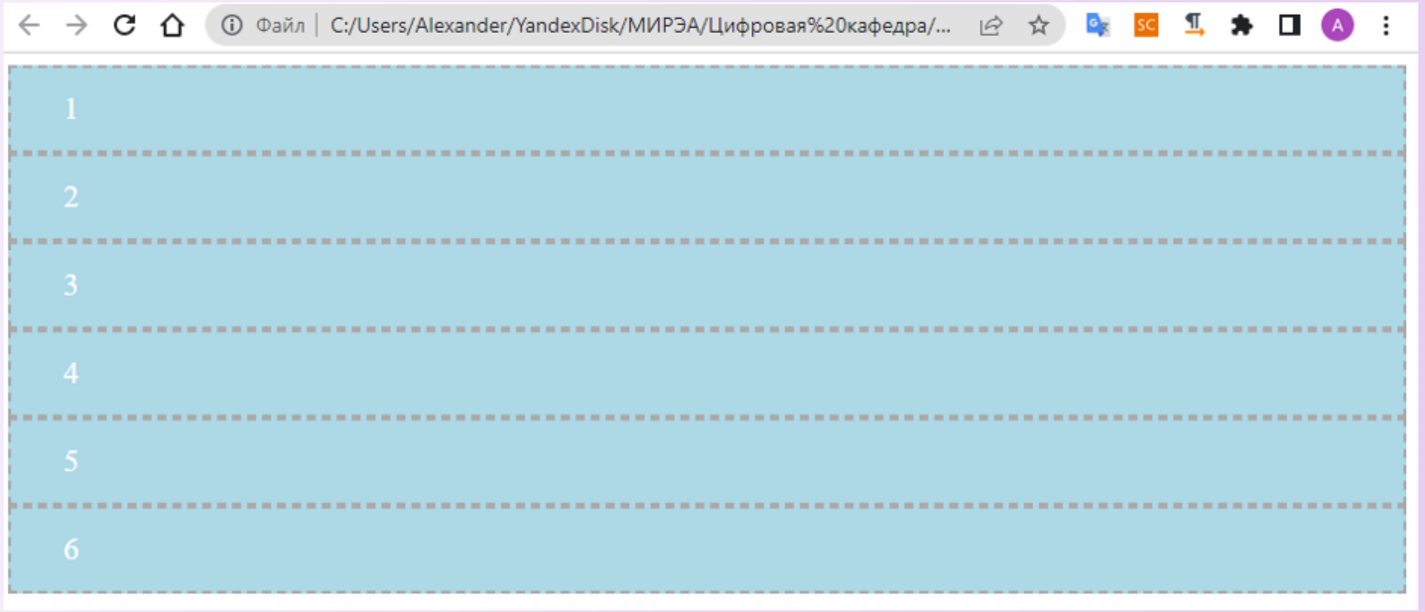
Эти правила будут срабатывать при ширине области просмотра до 600px:



# Grid Layout

CSS Grid Layout (далее просто Grid) – это способ двумерной раскладки. Именно ДВУмерной, в отличие от Flexbox. Flexbox позволяет полноценно управлять элементами только по одной оси.



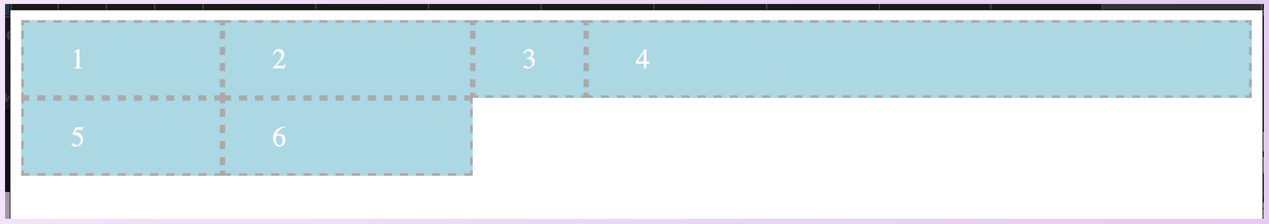
# Grid Layout

**grid-template-columns** определяет количество колонок и может задавать ширину каждой из них. В данном случае количество колонок равно 4, поскольку перечислены 4 параметра

При этом ширина колонок может задаваться в разных единицах. Так px – это известные нам пиксели. vw - представляет собой процент ширины корневого элемента. Один VW равен 1% ширины области просмотра. Третья колонка ужимается под контент (свойство auto). А 1fr здесь используется, чтоб занять всё оставшееся доступное место


```css
.el {
  padding: 15px 35px;
  font-size: 21px;
  color: #fff;
  background: lightblue;
  border: 2px dashed #aaa;
}
```

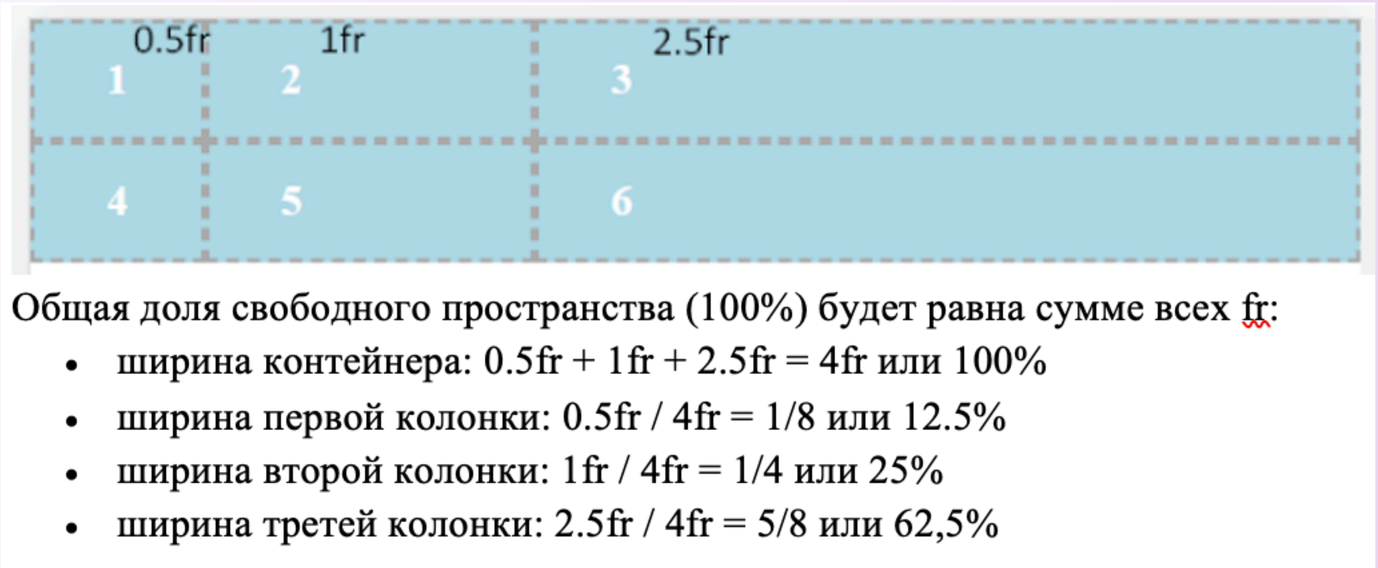
```css
.container {
  display: grid;
  grid-template-columns: 150px 20vw auto 1fr;
}
```



# Grid Layout

**fr (fractional unit)** всегда занимает свободное пространство. Когда несколько столбцов с ширинами в fr, цифрами мы указываем, какую часть свободного пространства должны делить между собой строки / колонки.

Эту единицу измерения можно использовать даже с дробными значениями. Например, строка **grid-template-columns: 0.5fr 1fr 2.5fr;** даст следующий результат:

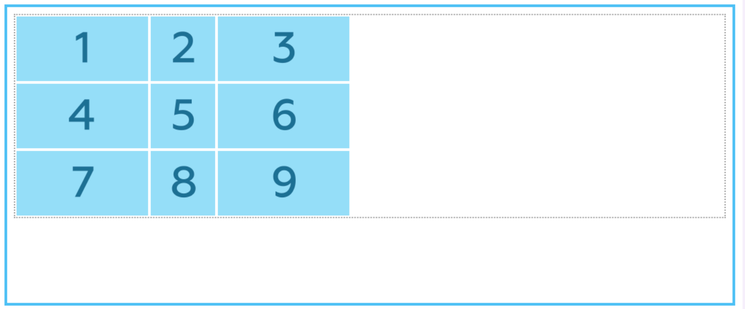



Общая доля свободного пространства (100%) будет равна сумме всех fr:

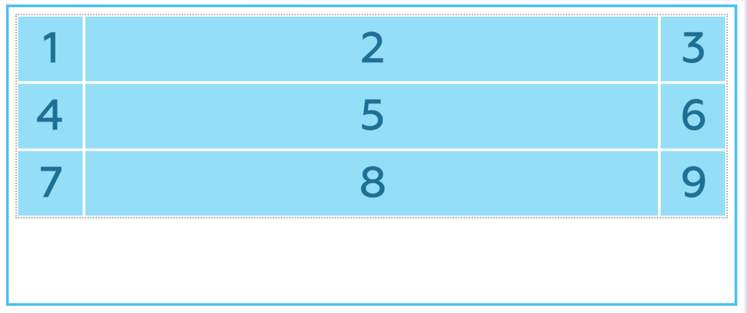
- ширина контейнера: 0.5fr + 1fr + 2.5fr = 4fr или 100%
- ширина первой колонки: 0.5fr / 4fr = 1/8 или 12.5%
- ширина второй колонки: 1fr / 4fr = 1/4 или 25%
- ширина третей колонки: 2.5fr / 4fr = 5/8 или 62,5%

# grid-template-columns

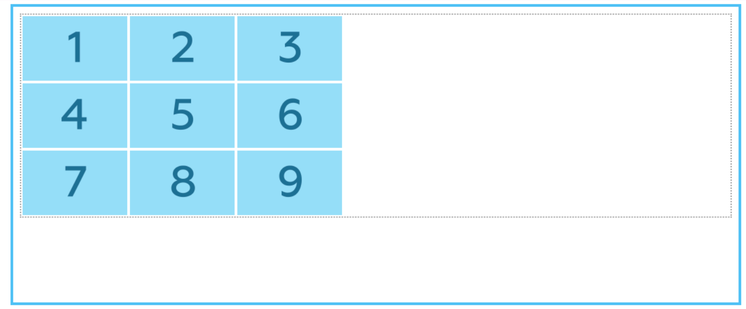
## Также работает и с grid-template-rows




```
.parent {
  display: grid;
  grid-template-columns: 100px 50px 100px;
}
```




```
.parent {
  display: grid;
  grid-template-columns: 50px 1fr 50px;
}
```




```
.parent {
  display: grid;
  grid-template-columns: repeat(3, 80px)
}
```