CSS flexbox

CSS flexbox *(Flexible Box Layout Module)* — модуль макета гибкого контейнера — представляет собой способ компоновки элементов, в основе лежит идея оси.

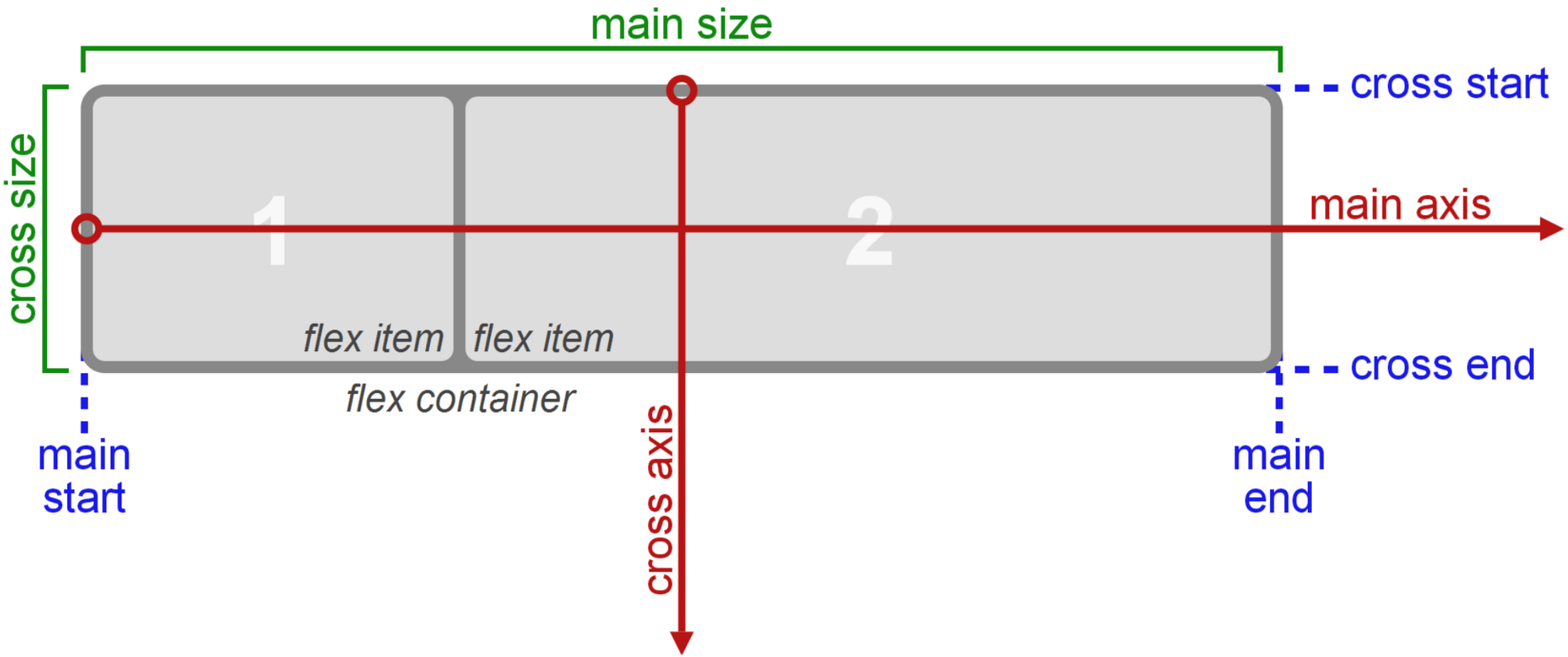
Flexbox состоит из **гибкого контейнера (flex container)** и **гибких элементов (flex items)**. Гибкие элементы могут выстраиваться в строку или столбик, а оставшееся свободное пространство распределяется между ними различными способами.

Flexbox используется для создания одномерных макетов, например, навигационной панели, так как flex-элементы можно размещать только по одной из осей.

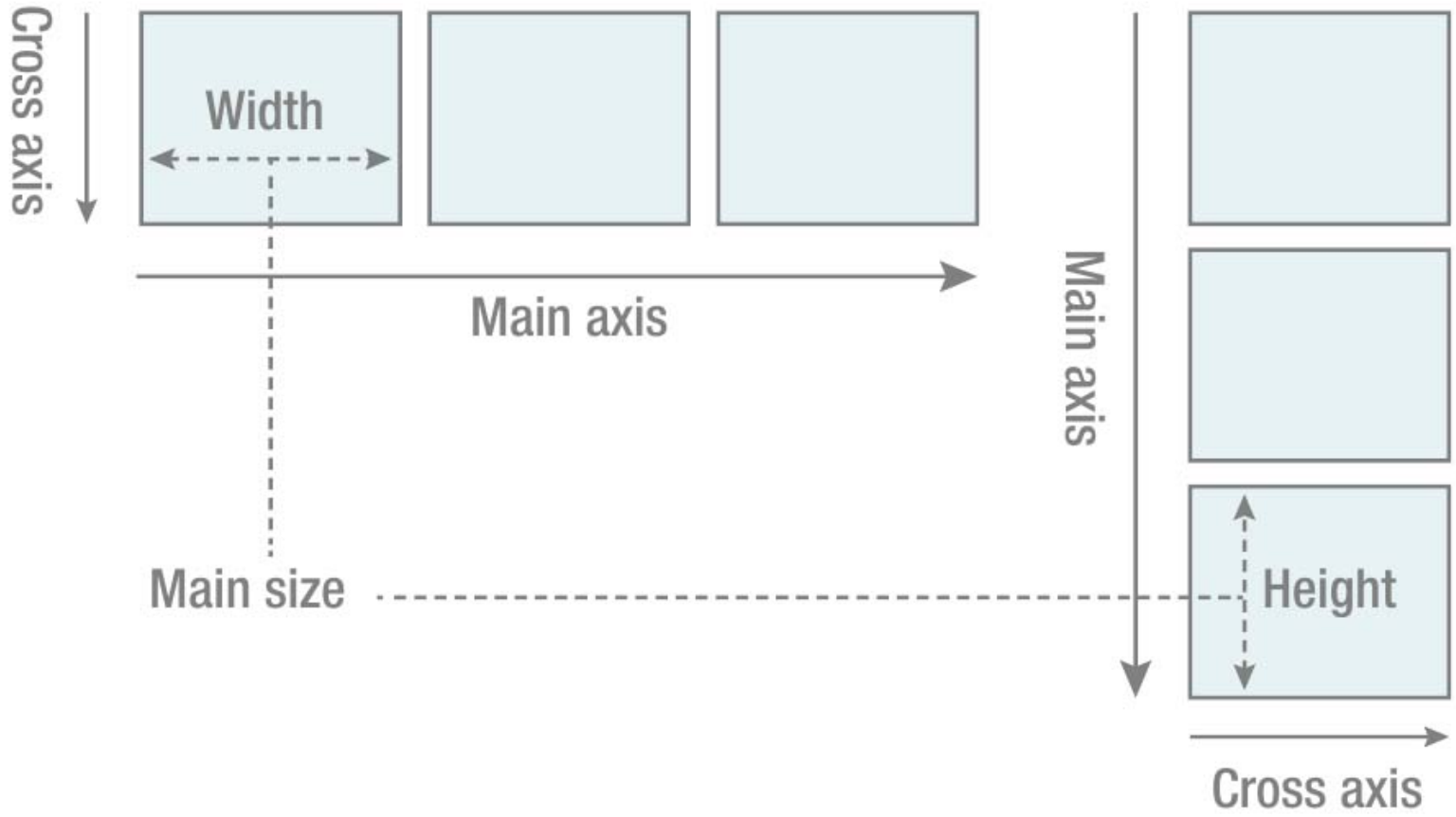
Модуль flexbox позволяет решать следующие задачи:

- Располагать элементы в одном из четырех направлений: слева направо, справа налево, сверху вниз или снизу вверх.
- Переопределять порядок отображения элементов.
- Автоматически определять размеры элементов таким образом, чтобы они вписывались в доступное пространство.
- Решать проблему с горизонтальным и вертикальным центрированием.
- Переносить элементы внутри контейнера, не допуская его переполнения.
- Создавать колонки одинаковой высоты.

Flexbox Layout



Flexbox Layout: режим строки и колонки



Flex-контейнер

Flex-контейнер устанавливает новый гибкий контекст форматирования для его содержимого. Flex-контейнер не является блочным контейнером, поэтому для дочерних элементов не работают такие CSS-свойства, как float, clear, vertical-align.
Модель flexbox-разметки связана с определенным значением CSS-свойства display родительского html-элемента, содержащего внутри себя дочерние блоки. Для управления элементами с помощью этой модели нужно установить свойство display следующим образом:

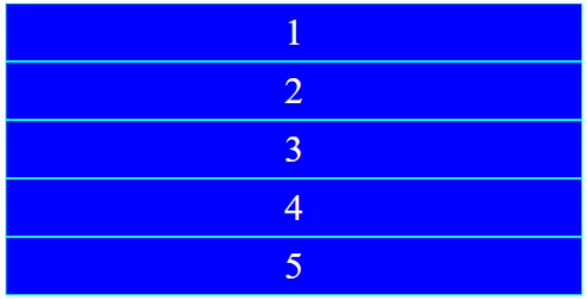
```
.flex-container {
/*генерирует flex-контейнер уровня блока*/
  display: flex;
}
.flex-container {
/*генерирует flex-контейнер уровня строки*/
  display: inline-flex;
}
```

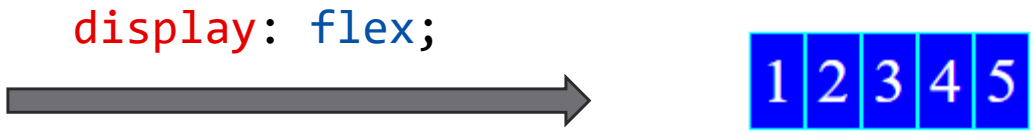
После установки данных значений свойства каждый дочерний элемент автоматически становится flex-элементом, выстраиваясь в один ряд (вдоль главной оси). При этом блочные и строчные дочерние элементы ведут себя одинаково, т.е. ширина блоков равна ширине их содержимого с учетом внутренних полей и рамок элемента.
Если родительский блок содержит текст или изображения без оберток, они становятся анонимными flex-элементами. Текст выравнивается по верхнему краю блока-контейнера, а высота изображения становится равной высоте блока, т.е. оно деформируется.

Создание Flex контейнера

```
 <div class="flex-container">
    <div class="flex-item">1</div>
    <div class="flex-item">2</div>
    <div class="flex-item">3</div>
    <div class="flex-item">4</div>
    <div class="flex-item">5</div>
</div>
```

```
.flex-item{
    background-color: blue;
    border: 1px solid cyan;
    color: white;
    padding: 5px;
    font-size: 2em;
    text-align: center;
}
.flex-container{
    display: flex;
}
```





Направление главной оси: flex-direction

Свойство относится к flex-контейнеру. Управляет направлением главной оси, вдоль которой укладываются flex-элементы, в соответствии с текущим режимом записи.

- `flex-direction: row;` /* Значение по умолчанию, слева направо (в rtl справа налево). Flex-элементы выкладываются в строку. Начало (main-start) и конец (main-end) направления главной оси соответствуют началу (inline-start) и концу (inline-end) оси строки (inline-axis).*/
- `flex-direction: row-reverse;` /* Направление справа налево (в rtl слева направо). Flex-элементы выкладываются в строку относительно правого края контейнера (в rtl — левого).*/
- `flex-direction: column;` /* Направление сверху вниз. Flex-элементы выкладываются в колонку.*/
- `flex-direction: column-reverse;` /* Колонка с элементами в обратном порядке, снизу вверх.*/

Выравнивание по главной оси: justify-content

Свойство выравнивает flex-элементы по главной оси flex-контейнера, распределяя свободное пространство, незанятое flex-элементами. Когда элемент преобразуется в flex-контейнер, flex-элементы по умолчанию сгруппированы вместе (если для них не заданы поля margin). Промежутки добавляются после расчета значений margin и flex-grow. Если какие-либо элементы имеют ненулевое значение flex-grow или margin: auto;, свойство не будет оказывать влияния.

- `justify-content: flex-start;` – значение по умолчанию. Flex-элементы выкладываются в направлении, идущем от начала flex-контейнера.
- `justify-content: flex-end;` – flex-элементы размещаются в конце flex-контейнера.
- `justify-content: center;` – flex-элементы выравниваются по центру flex-контейнера.
- `justify-content: space-between;` – flex-элементы равномерно распределяются по линии. Первый flex-элемент помещается вровень с краем начальной линии, последний flex-элемент — вровень с краем конечной линии, а остальные flex-элементы на линии распределяются так, чтобы расстояние между любыми двумя соседними элементами было одинаковым. Если оставшееся свободное пространство отрицательно или в строке присутствует только один flex-элемент, это значение идентично параметру flex-start.
- `justify-content: space-around;` – flex-элементы на линии распределяются так, чтобы расстояние между любыми двумя смежными flex-элементами было одинаковым, а расстояние между первым / последним flex-элементами и краями flex-контейнера составляло половину от расстояния между flex-элементами.

Выравнивание по главной оси: justify-content

justify-content: start;

justify-content: end;

justify-content: center;

justify-content: space-between;

justify-content: space-around;

Выравнивание по поперечной оси: align-items и align-self

Flex-элементы можно выравнивать по поперечной оси текущей строки flex-контейнера. align-items устанавливает выравнивание для всех элементов flex-контейнера, включая анонимные flex-элементы. align-self позволяет переопределить это выравнивание для отдельных flex-элементов. Если любое из поперечных margin flex-элемента имеет значение auto, align-self не имеет никакого влияния.

- `align-items: flex-start;` верхний край flex-элемента помещается вплотную с flex-линией (или на расстоянии, с учетом заданных полей margin и рамок border элемента), проходящей через начало поперечной оси.
- `align-items: flex-end;` нижний край flex-элемента помещается вплотную с flex-линией (или на расстоянии, с учетом заданных полей margin и рамок border элемента), проходящей через конец поперечной оси.
- `align-items: center;` поля flex-элемента центрируется по поперечной оси в пределах flex-линии.
- `align-items: stretch;` если поперечный размер flex-элемента вычисляется как auto и ни одно из поперечных значений margin не равно auto, элемент растягивается. Значение по умолчанию.

Выравнивание по поперечной оси: align-items и align-self

align-items: stretch;

align-items: flex-start;

align-items: flex-end;

align-items: center;

Управление многострочностью flex-контейнера: flex-wrap

Свойство определяет, будет ли flex-контейнер однострочным или многострочным, а также задает направление поперечной оси, определяющее направление укладки новых линий flex-контейнера.

`flex-wrap: nowrap;` /* Значение по умолчанию. Flex-элементы не переносятся, а располагаются в одну линию слева направо (в rtl справа налево).*/

`flex-wrap: wrap;` /* Flex-элементы переносятся, располагаясь в несколько горизонтальных рядов (если не помещаются в один ряд) в направлении слева направо (в rtl справа налево).*/

`flex-wrap: wrap-reverse;` /* Flex-элементы переносятся на новые линии, располагаясь в обратном порядке слева-направо, при этом перенос происходит снизу вверх.*/

Краткая запись направления и многострочности: flex-flow

Свойство позволяет определить направления главной и поперечной осей, а также возможность переноса flex-элементов при необходимости на несколько строк. Представляет собой сокращённую запись свойств `flex-direction` и `flex-wrap`.

Значение по умолчанию `flex-flow: row nowrap;`

```
section {
  display: flex;
  flex-flow: row nowrap;
}
```

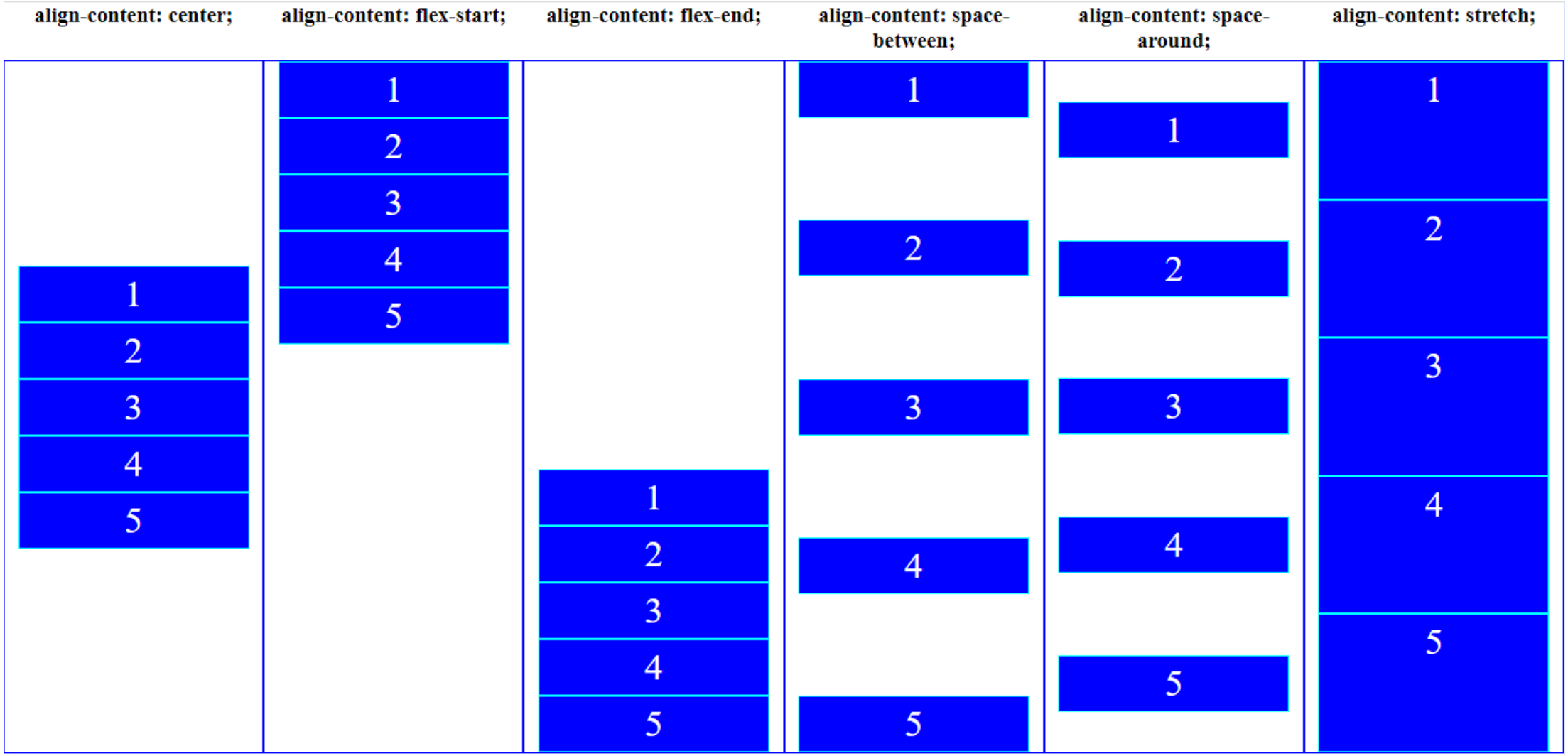
Выравнивание строк flex-контейнера: align-content

Свойство `align-content` устанавливает тип выравнивания строк флекс-элементов по вертикали внутри флекс-контейнера, позволяя управлять свободным пространством.

Свойство `align-content` работает только в случае, если разрешен перенос строк и указано направление `flex-flow: row/row-reverse/column/column-reverse wrap/wrap-reverse;` и высота flex-контейнера.

- `align-content: flex-start;` строки укладываются по направлению к началу flex-контейнера. Край первой строки помещается вплотную к краю flex-контейнера, каждая последующая — вплотную к предыдущей строке.
- `align-content: flex-end;` строки укладываются по направлению к концу flex-контейнера. Край последней строки помещается вплотную к краю flex-контейнера, каждая предыдущая — вплотную к последующей строке.
- `align-content: center;` строки укладываются по направлению к центру flex-контейнера. Строки расположены вплотную друг к другу и выровнены по центру flex-контейнера с равным расстоянием между начальным краем содержимого flex-контейнера и первой строкой и между конечным краем содержимого flex-контейнера и последней строкой.
- `align-content: space-between;` строки равномерно распределены в flex-контейнере. Если оставшееся свободное пространство отрицательно или в flex-контейнере имеется только одна flex-линия, это значение идентично flex-start. В противном случае край первой строки помещается вплотную к начальному краю содержимого flex-контейнера, край последней строки — вплотную к конечному краю содержимого flex-контейнера. Остальные строки распределены так, чтобы расстояние между любыми двумя соседними строками было одинаковым.
- `align-content: space-around;` строки равномерно распределены в flex-контейнере с половинным пробелом на обоих концах. Если оставшееся свободное пространство отрицательно, это значение идентично center. В противном случае строки распределяются таким образом, чтобы расстояние между любыми двумя соседними строками было одинаковым, а расстояние между первой / последней строками и краями содержимого flex-контейнера составляло половину от расстояния между строками.
- `align-content: stretch;` значение по умолчанию. Строки флекс-элементов равномерно растягиваются, заполняя все доступное пространство.

Выравнивание строк flex-контейнера: align-content



Гибкое изменение размеров flex элементов

Коэффициент роста: flex-grow

Свойство определяет коэффициент роста одного flex-элемента относительно других flex-элементов в flex-контейнере при распределении положительного свободного пространства. Свободное пространство — разница между размером flex-контейнера и размером всех его flex-элементов вместе. Если все элементы имеют одинаковый коэффициент flex-grow, то все они получат одинаковую долю свободного пространства, в противном случае оно распределяется в соответствии с соотношением, определённым различными коэффициентами flex-grow.

Коэффициент сжатия: flex-shrink

Свойство указывает коэффициент сжатия flex-элемента относительно других flex-элементов при распределении отрицательного свободного пространства. Умножается на базовый размер flex-basis. Отрицательное пространство распределяется пропорционально тому, насколько элемент может сжаться, поэтому, например, маленький flex-элемент не уменьшится до нуля, пока не будет заметно уменьшен flex-элемент большего размера.

Базовый размер: flex-basis

Свойство устанавливает начальный основной размер flex-элемента до распределения свободного пространства в соответствии с коэффициентами гибкости. Для всех значений, кроме auto и content, базовый гибкий размер определяется так же, как width в горизонтальных режимах записи. Процентные значения определяются относительно размера flex-контейнера, а если размер не задан, используемым значением для flex-basis являются размеры его содержимого.

Гибкое изменение размеров flex элементов

Определяющим аспектом гибкого макета является возможность «сгибать» flex-элементы, изменяя их ширину/высоту (в зависимости от того, какой размер находится на главной оси), чтобы заполнить доступное пространство в основном измерении. Это делается с помощью свойства `flex`.

Свойство `flex` является сокращённой записью свойств `flex-grow`, `flex-shrink` и `flex-basis`.

Значение по умолчанию: `flex: 0 1 auto;`. Можно указывать как одно, так и все три значения свойств.

Flex-элемент будет полностью «негибок», если его значения flex-grow и flex-shrink равны нулю, и «гибкий» в противном случае.

W3C рекомендует использовать сокращённую запись, так как она правильно сбрасывает любые неуказанные компоненты, чтобы подстроиться под типичное использование.

```
article:nth-of-type(1){
  flex: 1 0 200px;
}
article:nth-of-type(2){
  flex: 2 1 200px;
}
article:nth-of-type(3){
  flex: 3 1 200px;
}
```

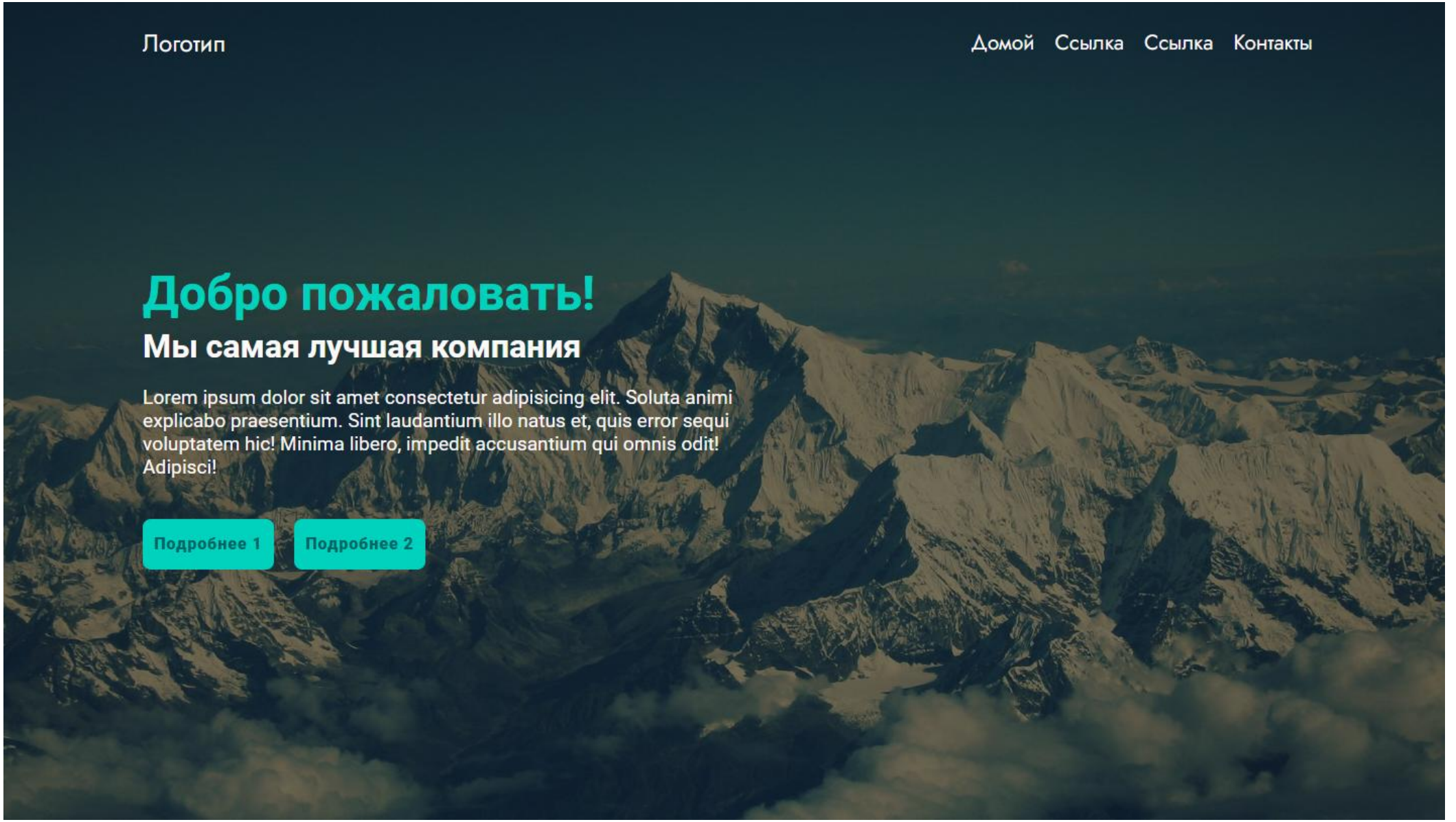
Порядок отображения flex-элементов: order

Свойство определяет порядок, в котором flex-элементы отображаются и располагаются внутри flex-контейнера. Применяется к flex-элементам.

Первоначально все flex-элементы имеют order: 0;. При указании значения от -1 для элемента он перемещается в начало сроки, значение 1 — в конец. Если несколько flex-элементов имеют одинаковое значение order, они будут отображаться в соответствии с исходным порядком.

```
article:nth-of-type(2){
        order: -1;
}
```

Пример использования Flex box



Пример использования Flex box

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ООО Ромашка</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header class="header">
        <a href="#" class="logo">Логотип</a>
        <nav class="navbar">
            <a href="#" class="active">Домой</a>
            <a href="#">Ссылка</a>
            <a href="#">Ссылка</a>
            <a href="#">Контакты</a>
        </nav>
    </header>
    <main class="home">
        <div class="home-content">
            <h1>Добро пожаловать!</h1>
            <h2>Мы самая лучшая компания</h2>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Soluta animi explicabo praesentium. Sint laudantium illo natus et, quis
error sequi voluptatem hic! Minima libero, impedit accusantium qui omnis odit! Adipisci!</p>
            <div class="home-btn-group">
                <a href="#" class="btn">Подробнее 1</a>
                <a href="#" class="btn">Подробнее 2</a>
            </div>
        </div>
    </main>

</body>
</html>
```

Пример использования Flex box

```css
:root{
    --home-bg-color: rgb(0, 100, 100);
    --home-main-color: rgb(255, 250, 250);
    --home-add-color: rgb(0, 210 , 190);
}

* {
    margin: 0;
    padding: 0;
    font-family: 'Roboto', sans-serif;
    box-sizing: border-box;
}
body{
    background-color: var(--home-bg-color);
    color: var(--home-main-color);
    background: linear-gradient(rgba(0, 0, 0, 0.6), rgba(0, 0, 0, 0.5)) ,  url(bg.jpg);
    background-size: cover;
    background-position: center;
}
.header{
    position: absolute;
    width: 100%;
    min-height: 80px;
    padding: 0 10%;
    background-color: transparent;
    display: flex;
    justify-content: space-between;
    align-items: center;
}
.logo{
    font-size: 1.5rem;
    color:var(--home-main-color);
    text-decoration: none;
    font-family: 'Jost', sans-serif;
    transition: 0.3s;
}
```

```css
.navbar{
    display: flex;
    flex-wrap: nowrap;
    background-color: transparent;
    justify-content: space-between;
    align-items: center;
    height: 100%;
}
.navbar a{
    color:var(--home-main-color);
    text-decoration: none;
    font-size: 1.3rem;
    font-family: 'Jost', sans-serif;
    transition: 0.3s;
    padding: 0 10px;
}
.navbar a:hover, .logo:hover{
    color:var(--home-add-color);
}
.home{
    height: 100vh;
    padding: 0 10%;
    display: flex;
    align-items: center;
}
.home-content{
    max-width: 600px;
}
.home-content h1{
    font-size: 3rem ;
    font-weight: 700;
    color: var(--home-add-color);
    margin-bottom: 5px;
}
```

```css
.home-content h2{
    font-size: 2rem ;
    font-weight: 700;
    margin-bottom: 20px;
}
.home-content p{
    font-size: 1.2rem;
    margin-bottom:  40px ;
}

.home-btn-group{
    height: 50px;
    width: 280px;
    display: flex;
    justify-content: space-between;
}
.home-btn-group .btn{
    text-decoration: none;
    background-color:  var(--home-add-color);
    color:  var(--home-bg-color);
    border: 2px solid var(--home-add-color);
    border-radius: 8px;
    font-size: 1rem;
    font-weight: 900;
    letter-spacing: 1px;
    width: 130px;
    height: 100%;
    display: inline-flex;
    align-items: center;
    justify-content: center;
    transition: 0.3s;
}
.home-btn-group .btn:hover{
    background-color:  transparent;
    color: var(--home-add-color);
}
```