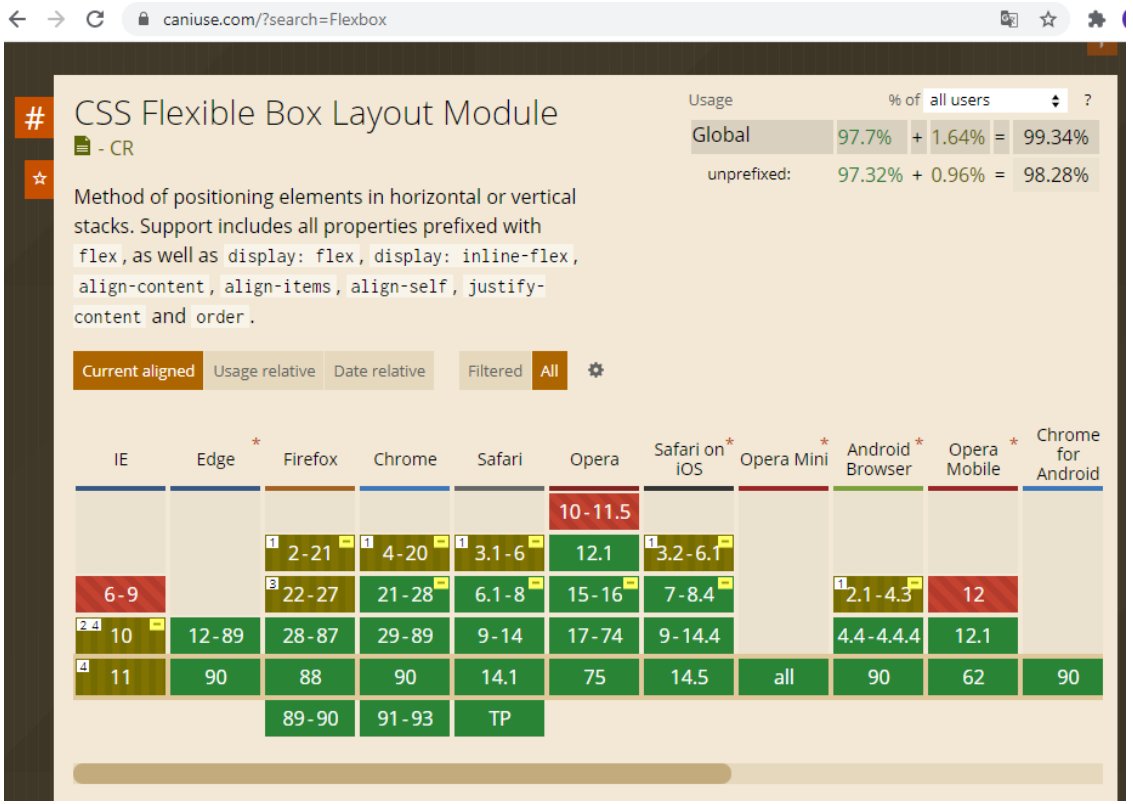
Flexbox

Модуль 7 (2 пары)

Модель Flexbox

- Долгое время верстальщики мечтали об инструменте, позволяющем им размещать по 2, 3, 4 и более колонки в один ряд очень простым способом. Сначала это было реализовано с помощью таблиц, затем с помощью свойства float: left или display: inline-block, но каждый из этих способов имел свои нюансы и ограничения. Поэтому w3.org в 2009 был опубликован черновик «Flexible Box Layout Module», в 2011 и 2012 году он претерпел изменения, и наконец, с 2014 года в стандарт CSS3 вошел в том виде, в котором мы используем его в данный момент.

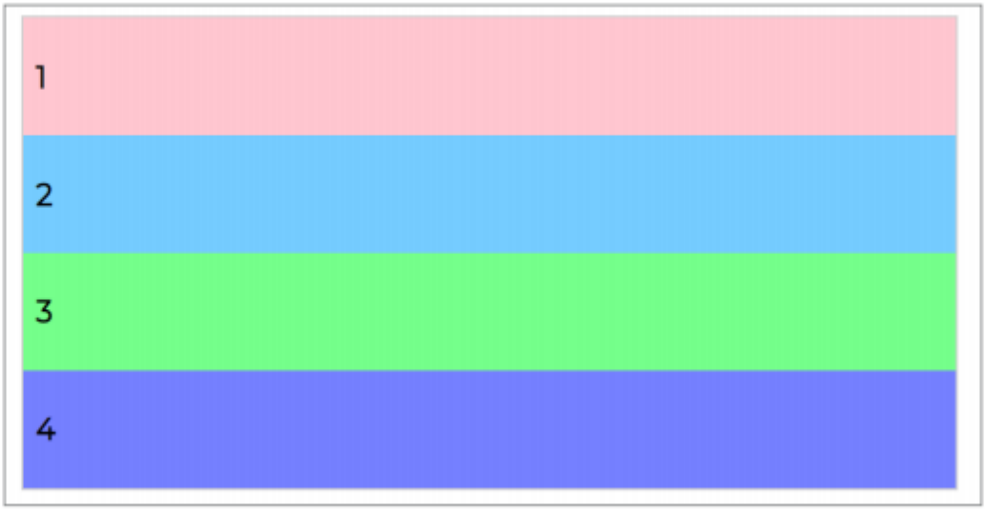
- Сейчас Flexbox уже прочно обосновался в арсенале верстальщиков и широко используется повсеместно, в том числе и в таких фреймворках, как Bootstrap-4 или Foundation-6. К его плюсам можно отнести очень хорошую поддержку всеми современными браузерами и частичную — теми версиями браузеров, которые были выпущены между 2011 и 2013-м годом. Для последних необходимо указывать ряд свойств с префиксами -webkit- (Chrome, Safari), -ms- (Internet Explorer 10), -moz- (Mozilla Firefox). Более подробную информацию по этому вопросу вы найдете на сайте caniuse.com



Что такое Flexbox?

- Flexbox (CSS Flexible Box Layout) — это модель отображения содержимого страницы, при которой вы получаете набор гибких элементов внутри контейнера, имеющего свойство display со значением flex или inline-flex.
- Из данного определения следует, что у нас есть один родительский элемент и несколько вложенных, или дочерних элементов. Собственно, и свойства, которые существуют в модели Flexbox, подразделяются на свойства для элемента-контейнера и для дочерних элементов. Рассмотрим их последовательно.
- Предположим, что у нас есть 4 элемента внутри пятого , который является их родителем (или контейнером). Каждый из этих элементов имеет цифру от 1 до 4-х — порядковый номер, обозначающий последовательность их появления в html-разметке, высоту в 100px и свой цвет фона. У каждого div-а есть свойство display: block по умолчанию, поэтому они расположены друг под другом и занимают все свободное пространство внутри родительского div-а, обведенного серой рамкой.

С помощью свойств Flexbox модели мы можем изменить положение дочерних элементов внутри родительского контейнера, применив ряд соответствующих свойств.

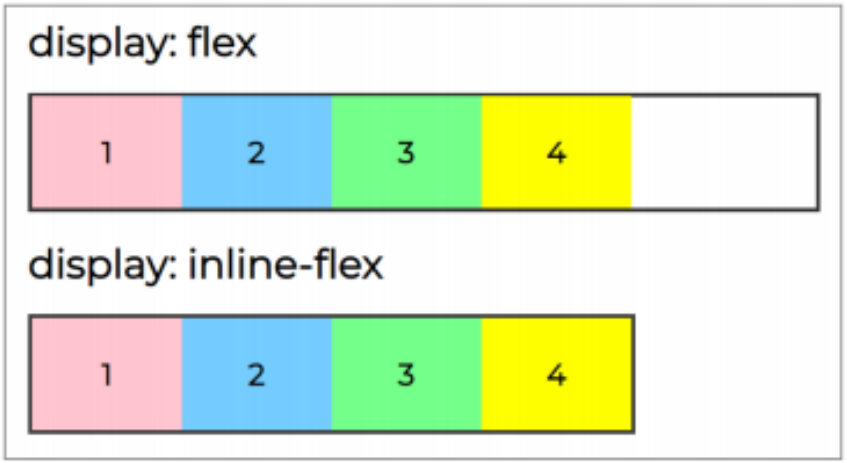


Свойства flex-контейнера

- Элемент, который содержит ряд вложенных элементов, мы будем называть flex-контейнером. Для него обязательно нужно задать свойство display:

```
display: flex | inline-flex
```

- Разница между двумя значениями заключается в том, что при display: flex размер контейнера по ширине равен всему доступному пространству внутри родительского элемента (аналогично display: block), а при display: inline-flex размер контейнера определяется размером вложенных элементов (аналогично display: inline-block).
- Обратите внимание на черную рамку, которая на первой картинке значительно длиннее, чем на второй.

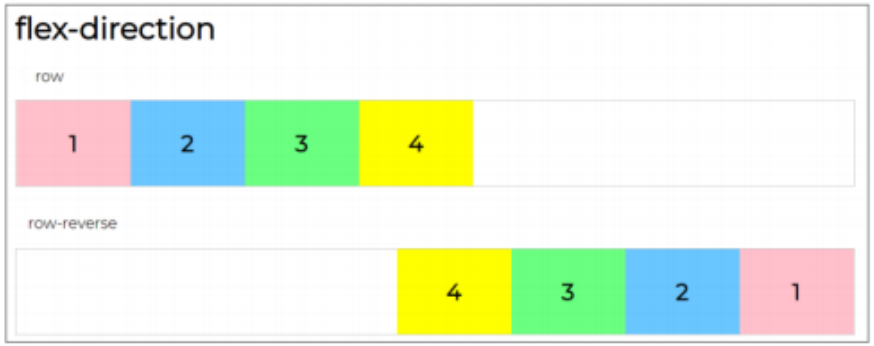


Главная и поперечная ось

- Как видно из рисунка выше, все элементы flex-контейнера выстроились в одну горизонтальную линию. Такое поведение задается еще одним свойством для flex-контейнера — flex-direction:

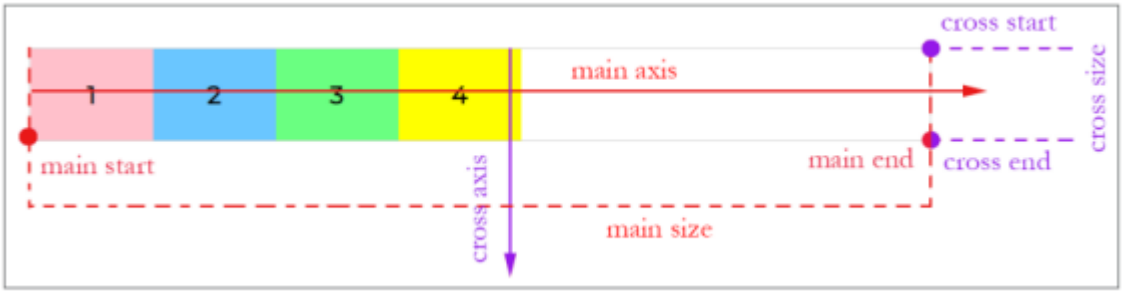
```
flex-direction: row | row-reverse | column | column-revers
```

- На рисунке видно, что вложенные элементы перестраиваются в зависимости от значения. Направление row или row-reverse выстраивает дочерние элементы по горизонтали в направлении слева направо или справа налево. Значения column или column-reverse меняют направление размещения блоков на вертикальное: сверху вниз или снизу вверх.



Главная и поперечная ось

- В спецификации CSS для Flex-контейнера существует понятие главной и поперечной оси. Главная ось (main axis) направлена слева направо (для арабских языков (rtl) — справа налево), а поперечная (cross axis) — сверху вниз:

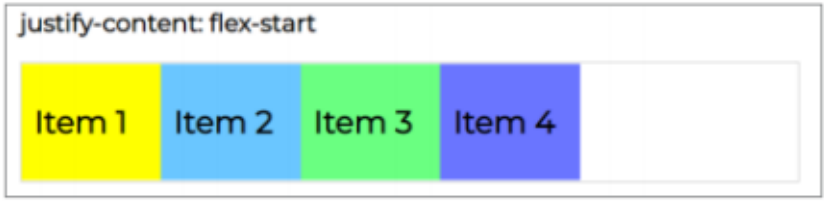


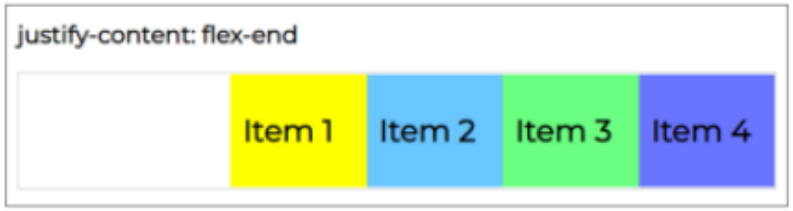
- Именно направление главной оси, а также ее начало и конец (main start и main end) определяет размещение вложенных flex-элементов при значении flex-direction, заданных как row или row-reverse. В случае изменения направления flex-direction на column или column-reverse, определяющей становится поперечная ось и ее начало или конец (cross start и cross end). Размер главной (main size) или поперечной оси (cross size) важен для расчета размеров вложенных элементов, особенно, если они заданы в %.
- От направления главной и поперечной оси зависят остальные свойства flex-контейнера. Мы будем из рассматривать в основном для flex-direction: row. Это значение по умолчанию, как правило, достаточно для размещения большинства flex-элементов

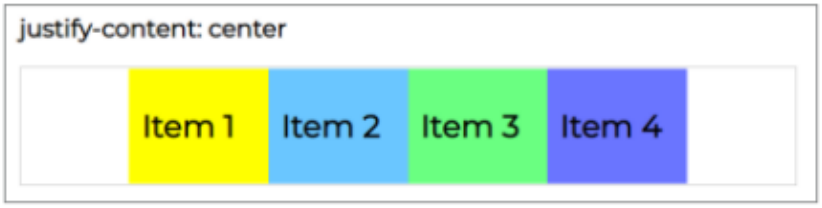
Свойство justify-content

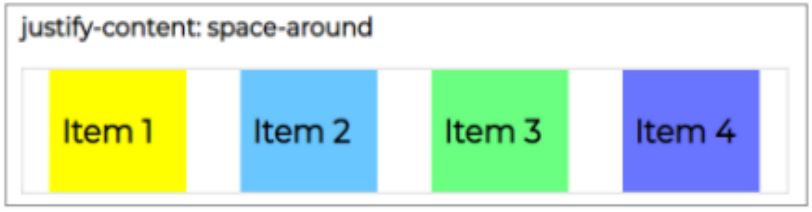
- Свойство justify-content определяет выравнивание flex-элементов вдоль главной оси. Оно часто используется при построении колонок, т.к. позволяет распределить контент внутри контейнера. Значения:

```
justify-content: flex-start | flex-end | center |
space-between | space-around |
space-evenly
```





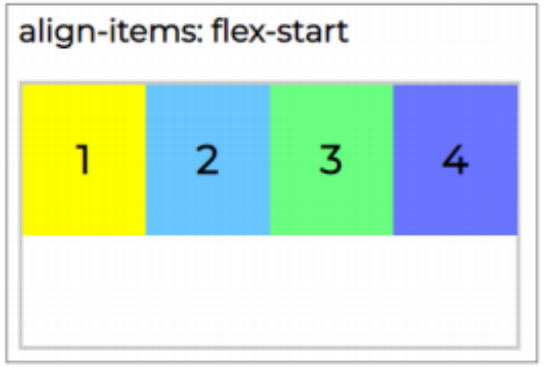


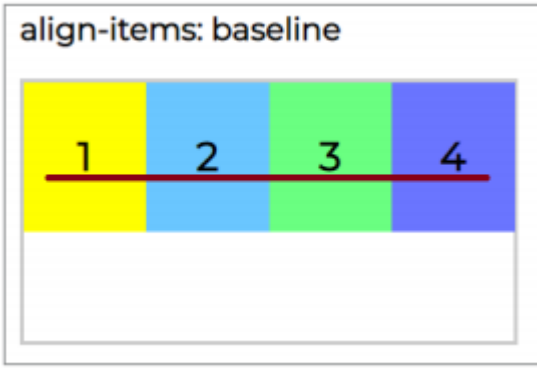


Свойство align-items

- Свойство align-items определяет, каким образом выравнивать flex-элементы вдоль поперечной оси. Для корректного использования этого свойства необходимо задать высоту контейнера больше, чем высота вложенных элементов (часто бывает в header). Значения:

```
align-items: flex-start | flex-end | center | stretch | baseline
```





Многострочность внутри flex-контейнера

- Далеко не всегда располагать элементы нужно только в одну строку. Поэтому в стандарте CSS предусмотрены свойства для многострочного flex-контейнера.

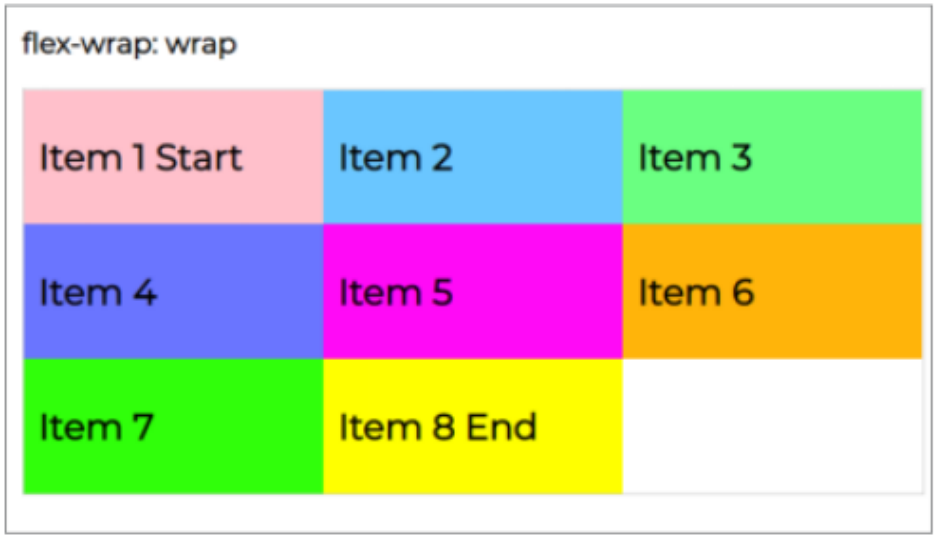
Свойство flex-wrap

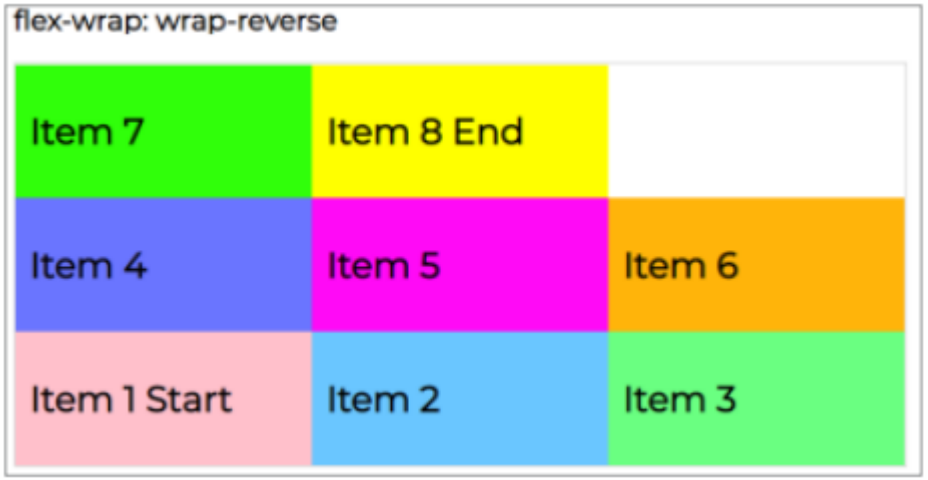
- Свойство flex-wrap отвечает за то, как будут размещаться вложенные flex-элементы внутри контейнера, если они не помещаются в одну строку. То есть это свойство управляет переносами строк внутри flex-контейнера.

```
flex-wrap: nowrap | wrap | wrap-reverse
```

Свойство flex-wrap

- При назначении flex-wrap в виде значения wrap элементы переносятся на новые строки в том порядке, в котором они расположены в html-разметке. Значение wrap-reverse оборачивает этот порядок, располагая первые дочерние элементы внизу, а последние — в верхнем ряду. Размер и количество строк во flex-контейнере будут также зависеть от значений свойств width или flex-basis для дочерних flex-элементов, которое мы рассмотрим чуть ниже.





Объединенное свойство flex-flow

- Свойство flex-flow предназначено для объединения свойств flex-direction и flex-wrap. По умолчанию имеет значение flex-flow: row nowrap. Имеет смысл использовать его при переопределении обоих этих значений или одного из них.

```
flex-flow: flex-direction || flex-wrap
```

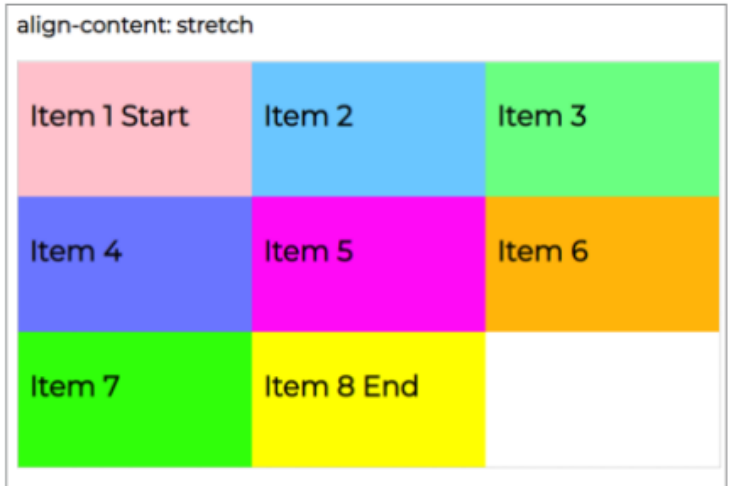
Свойство align-content

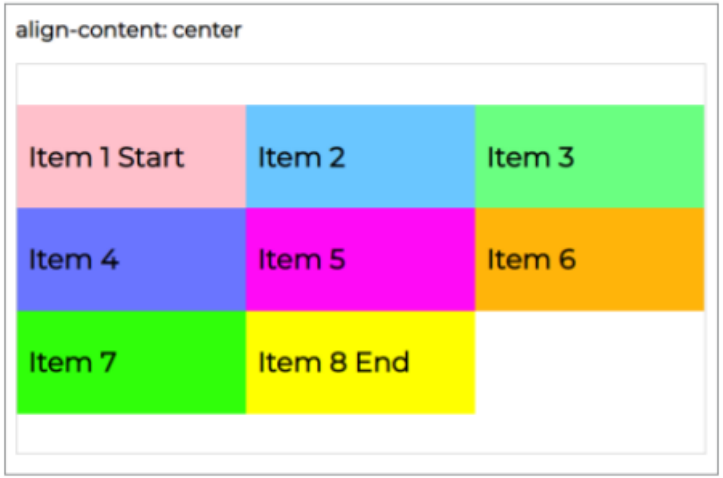
- Свойство align-content используется только в многострочном режиме (т.е. в случае, когда свойство flex-wrap имеет значение wrap или wrap-reverse). Оно определяет, каким образом ряды flex-элементы будут выровнены по вертикали, а также то, как они будут делить между собой все пространство flex-контейнера. Варианты значений свойства:

```
align-content: stretch | flex-start | flex-end | center | space-around | space-between
```

Свойство align-content

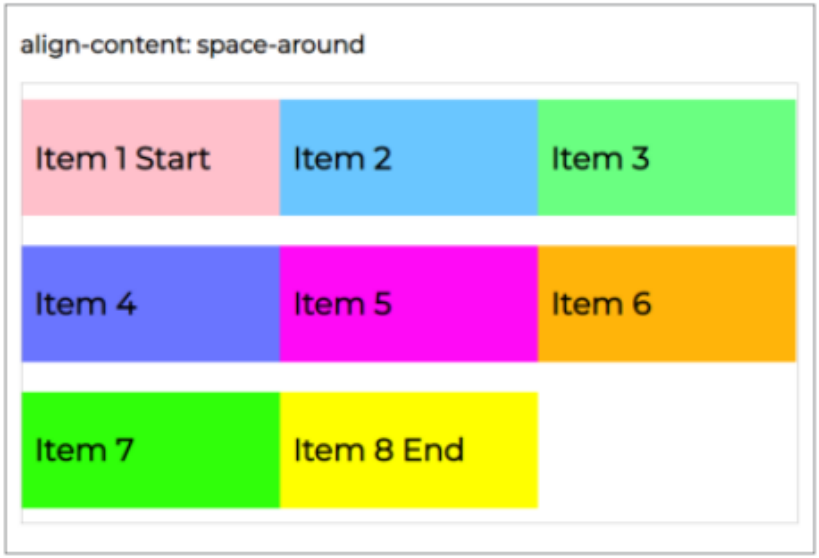
- По умолчанию свойство align-content имеет значение stretch, поэтому все flex-элементы вытянуты по высоте flex-контейнера. При изменении значений на flex-start или flex-end, они подтягиваются к верхней или нижней границе flex-контейнера

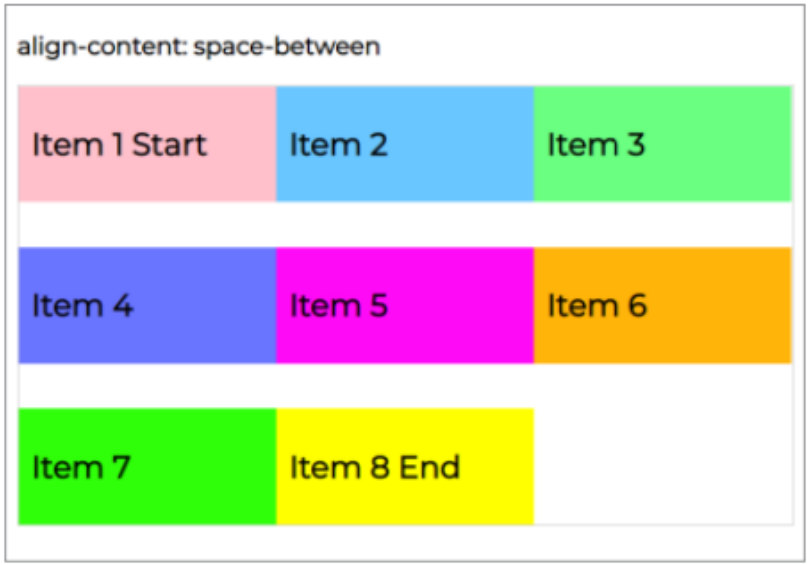




Свойство align-content

- Значение свойства align-content: center располагает flex-элементы по вертикальному центру flex-контейнера, оставляя сверху и снизу равные промежутки





- Остальные значения свойства align-content перераспределяют свободное пространство внутри контейнера так, чтобы оно делилось на равные промежутки между всеми линиями flex-элементов (значение space-between) или с добавлением половины промежутка сверху и половины снизу (значение space-around).

Свойства flex-элементов

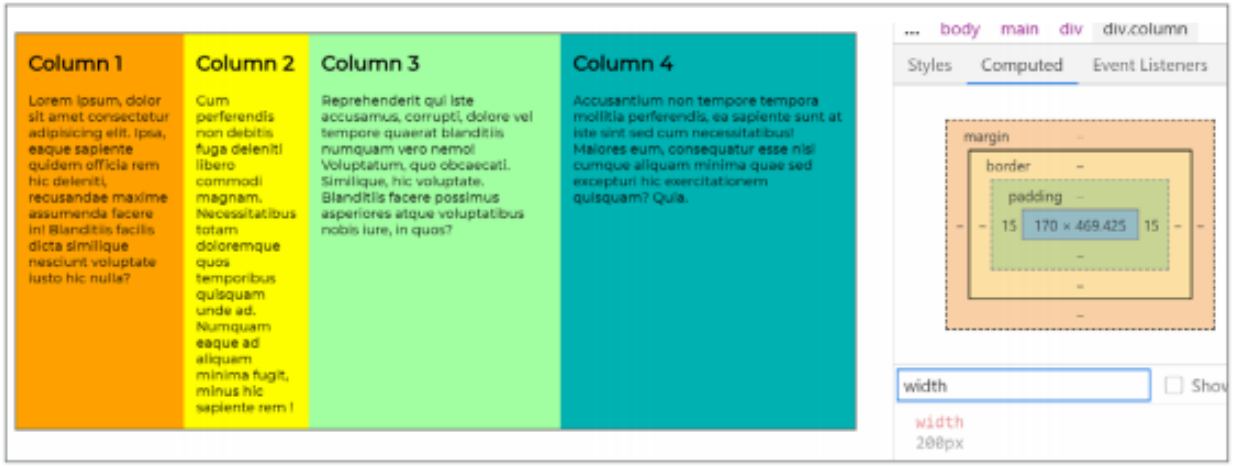
- Свойства flex-элементов позволяют управлять каждым из элементов внутри flex-контейнера по отдельности. Это, в первую очередь, свойства, которые управляют размерами дочерних элементов, их растяжимостью и сжимаемостью, порядком отображения вне зависимости от html-разметки, а также возможностью переопределить размещение flex-элементов, заданное свойствами flex-контейнера.

Свойство flex-basis

- Свойства flex-basis позволяет назначить базовый размер flex-элемента по основной оси. Вы можете задать его в любых единицах (5em, 4rem, 48%, 200px и др.) или указать, как auto (значение по умолчанию), но реальный размер flex-элемента будет зависеть от 2-х коэффициентов — flex-grow и flex-shrink. Также реальная ширина flex-элемента будет зависеть от min-width и max-width, назначенных для flex-элемента.

Свойство flex-basis

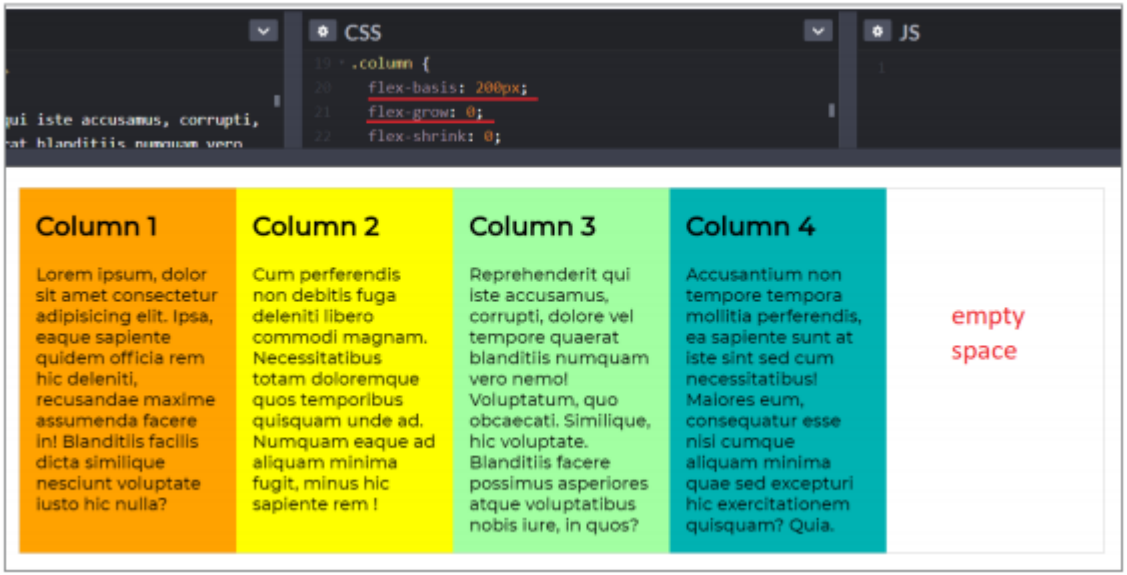
- Значение flex-basis будет корректироваться в зависимости от минимальной и максимальной ширины, т.е. ширина элемента не может быть меньше min-width и не может быть больше maxwidth. То же относится к размерам по вертикали (height, min-height, max-height), если свойство flex-direction имеет значение column



- На скриншоте выше ширина flex-контейнера составляет 1000px. Первый столбец имеет flex-basis: 200px, второй — flex-basis: 10%, т.е. должен быть шириной 100px, однако текста в нем больше, поэтому реальная ширина его 149px. В 3-м столбце flex-basis: 10%, но задано еще свойство min-width: 300px, поэтому его ширина составляет 300px. Для четвертого столбца задано свойство flex-grow: 1, поэтому его ширина увеличилась до 350.875px.

Свойство flex-grow

- Свойство flex-grow определяет фактор (коэффициент) растяжимости flex-элемента при наличии избыточного пространства во flex-контейнере. По умолчанию flex-grow имеет значение 0, т.е. все flex-элементы будут того размера, который им задан с помощью width или flex-basis, и часть flex-контейнера останется незанятой

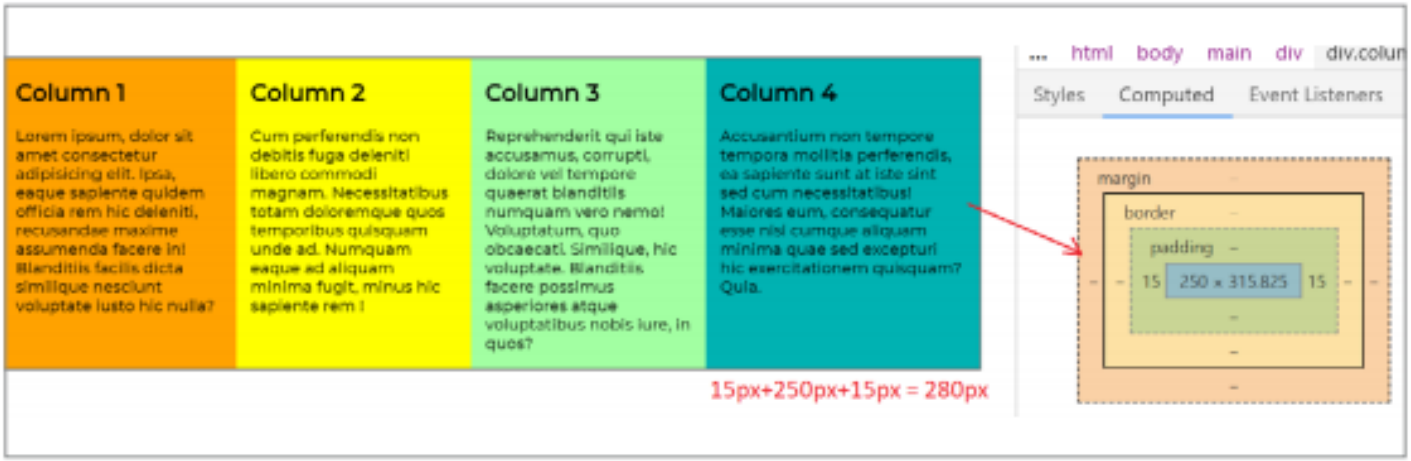


Свойство flex-grow

- Например, flex-контейнер с шириной 1000px содержит 4 flex-элемента с шириной 200px, т.е. 200px остаются незаполненными. Если каждому из этих элементов назначить flex-grow: 1, то 200px избыточного пространства нужно разделить на сумму этих коэффициентов, т.е. 200px:4 = 50px, а потом добавить результат к ширине каждого элемента. В результате этих действий ширина каждого flex-элемента станет 250px

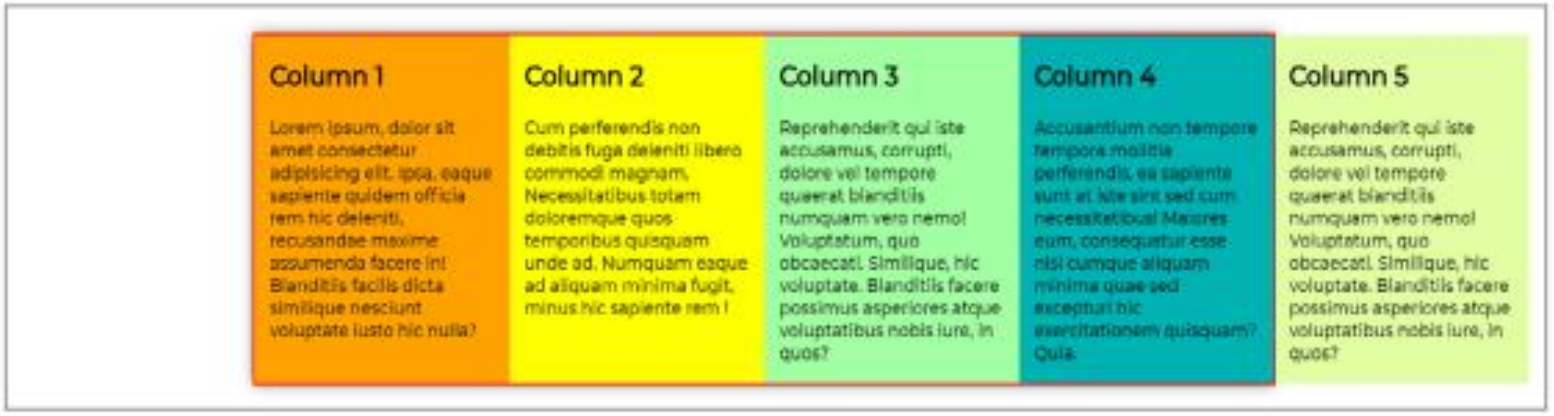
Свойство flex-grow

- Если же какой-то один flex-элемент получит свойство flex-grow: 2, то расчет будет уже другим. 200px: 5 = 40px. Цифра 5 — это сумма flex-grow всех элементов (1+1+1+2 = 5). Затем к 3-м из наших элементов нужно добавить по 40px, и их размер станет 240px, а к тому, у которого flex-grow: 2, мы добавим 80px (40px*2) — и получим 280px. Общая сумма всех ширин будет по-прежнему 1000px



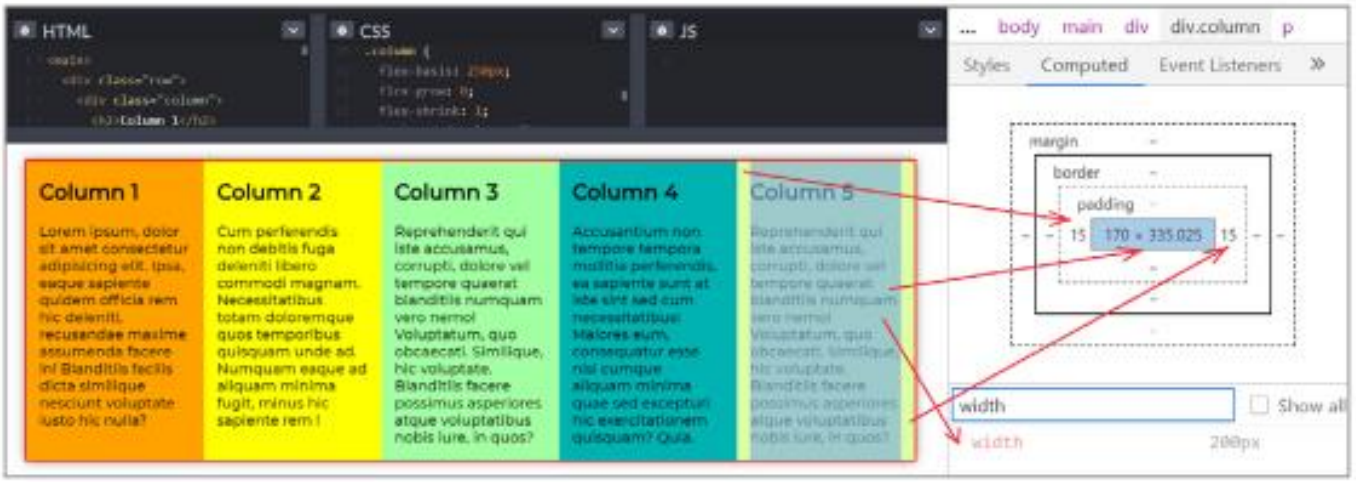
Свойство flex-shrink

- Свойство flex-shrink является фактором (коэффициентом) уменьшения ширины (высоты в случае flexdirection: column) flex-элемента, если во flex-контейнере недостаточно места. По умолчанию он равен 1, т.е. ширина всех элементов будет уменьшена, если ширина контейнера недостаточна для размещения всех элементов или для каждого них задан размер, который подразумевает большую ширину flex-контейнера



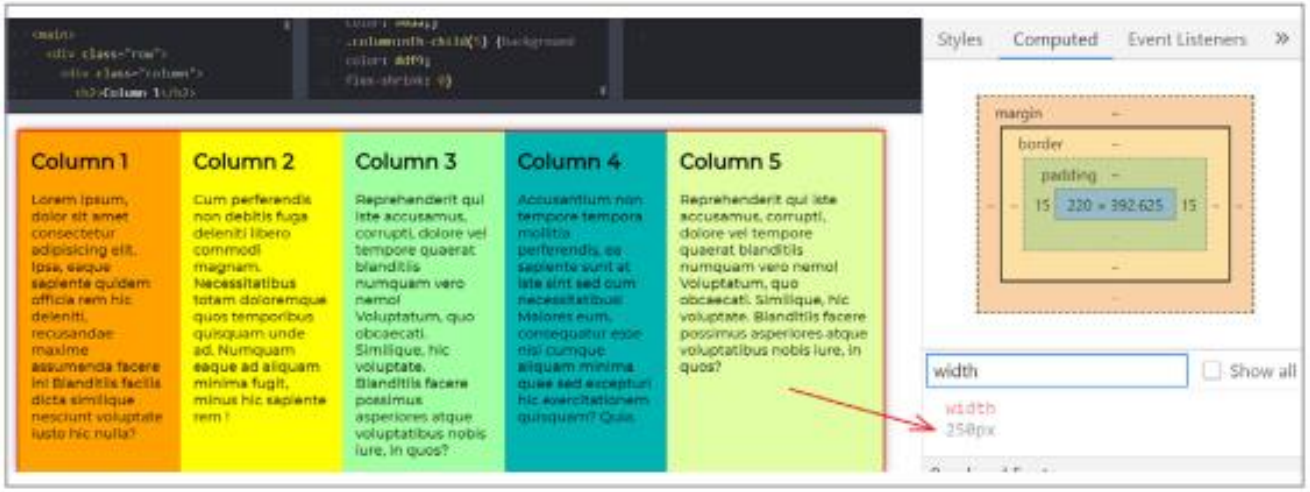
Свойство flex-shrink

- Например, ширина flex-контейнера равна 1000px, а ширина каждого из 5 flex-элементов 250px. Получится, что один из элементов должен выйти за пределы flex-контейнера. Однако, этого не происходит из-за flex-shrink: 1. Ширина каждого элемента уменьшится на 50px: 250px: 5 = 50px, и станет равна 200px: 250px50px=200px



Свойство flex-shrink

- Если добавить какому-либо одному элементу flexshrink:0, то он перестанет уменьшаться, оставаясь размером 250px. А нехватка в 250px распределится уже между 4 элементами, т.е. 250px: 4 = 62,5px. Затем браузер отнимет эту величину от ширины каждого элемента: 250px – 62.5px = 187.5px. Эта величина может быть изменена 23 Модель Flexbox в зависимости от контента (текста, картинок) внутри flex-элемента



Свойство flex-shrink

- Если же у одного из элементов flex-shrink:2, то этот элемент станет уже, чем его соседи. Расчет сжимания будет несколько сложнее для восприятия, т.к. 250px: 6 = 41.66px, т.к. цифра 6 берется путем складывания коэффициентов flex-shrink для всех flex-элементов, т.е. 1+1+1+1+2 = 6.
- А затем от ширины каждого элемента браузер отнимет 41.66px: 250px – 41.66px = 208.34, а для элемента с flexshrink:2 уменьшение будет больше, а ширина — меньше: 250px – 41.66*2 = 166.68px. Опять же — из-за размеров контента эта величина может изменяться

Обобщенное свойство flex

- Свойство flex позволяет задать сразу 3 предыдущих значения через пробел в такой последовательности:

```
flex: flex-grow flex-shrink flex-basis;
```

- Значение по умолчанию: flex: 0 1 auto; Варианты написания:

```
/* Одно значение, число без размерности: flex-grow */
flex: 2;
/* Одно значение, ширина/высота: flex-basis */
flex: 250px;
/* Два значения: flex-grow | flex-basis */
flex: 1 30px;
/* Два значения: flex-grow | flex-shrink */
flex: 2 2;
/* Три значения: flex-grow | flex-shrink | flex-basis */
flex: 2 1 25%;
```

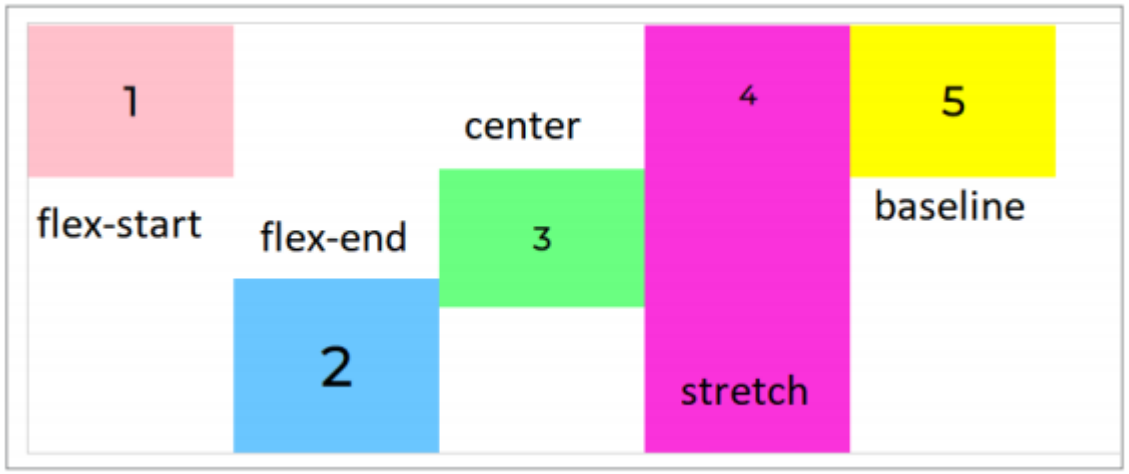
Свойство order

- Свойство order позволяет изменить последовательность расположения элементов. Это свойство определяет порядок следования flex-элементов и назначается в виде целочисленного значения как положительного, так и отрицательного. По умолчанию order имеет значение `0`, и элементы размещаются в той последовательности, в которой они расположены в html-разметке. Если вы назначаете какому-либо элементу значение order:1, он переместится в конец, если order: -1 — в начало.

Свойство align-self

- Свойство align-self меняет выравнивание отдельно взятого flex-блока по поперечной оси по сравнению со свойством flex-контейнера align-items. Варианты значений:

```
align-self: stretch | flex-start | flex-end | center | baseline
```



Спасибо за внимание.