

CSS grid/float/ preprocessors/Reset/ Normalize

Vira Huskova

- Float
- Align
- Flex
- Grid
- CSS Reset/Normalize
- Line-height
- Preprocessors
- Bundler

Свойство "float"

Свойство float в CSS занимает особенное место. До его появления расположить два блока один слева от другого можно было лишь при помощи таблиц.

float: left | right | none | inherit;

При применении этого свойства происходит следующее:

1. Элемент позиционируется как обычно
2. Затем *вынимается из документа потока* и сдвигается влево (для left) или вправо (для right) до того **как коснётся либо границы родителя, либо другого элемента с float**.

Если пространства по горизонтали не хватает для того, чтобы вместить элемент, то он сдвигается вниз до тех пор, пока не начнёт помещаться.

Другие непозиционированные **блочные элементы без float** ведут себя так, как будто элемента с float нет, так как он убран из потока.

Строки (**inline-элементы**), напротив, «**знают**» о float и обтекают элемент по сторонам.

Элемент при наличии float получает display:block.

То есть, указав элементу, у которого display:inline свойство float: left/right, мы автоматически сделаем его блочным.

В частности, для него будут работать width/height.

Исключением являются некоторые редкие display наподобие inline-table.

Ширина float-блока определяется по содержимому.

Вертикальные отступы margin элементов с float не сливаются с отступами соседей, в отличие от обычных блочных элементов.

Одно из первых применений float, для которого это свойство когда-то было придумано – это вёрстка текста с картинками, отжатыми влево или вправо.

Винни-Пух

Винни-Пух (англ. Winnie-the-Pooh) — плюшевый мишка, персонаж повестей и стихов Алана Александра Милна (цикл не^{я и обычно тоже называется «Винни-Пух»,} из самых известных героев детской^{3 1960-е—1970-е годы, благодаря пересказу} сказки «Винни Пух и все все все», а затем и фильмам^{са и «Союзмультфильм», где мишку озвучивал Евгений Леонов, Винни-Пух стал очень популярен в Советском Союзе.}



руководством Фёдора Хитрука было создано три мультильма.

Сценарий написал Хитрук в соавторстве с Заходером; работа соавторов не всегда шла гладко, что стало в конечном счёте причиной прекращения выпуска мультильмов (первоначально планировалось выпустить сериал по всей книге). Текст и картинки взяты с Wikipedia

float: right



Первый перевод «Винни-Пуха» в СССР вышел в 1958 году в Литве (лит. Mikelis Pūkuotukas), его выполнил 20-летний литовский писатель Виргилюс Чепайтис, пользуясь польским переводом Ирене Тувим. Епоследствии Чепайтис, познакомившись с английским оригиналом, с^т переработал свой перевод, переиздававшийся неоднократно.

float: right

История Винни-Пуха в России начинается с того же 1958 года, когда с книгой познакомился Борис Владимирович Заходер.



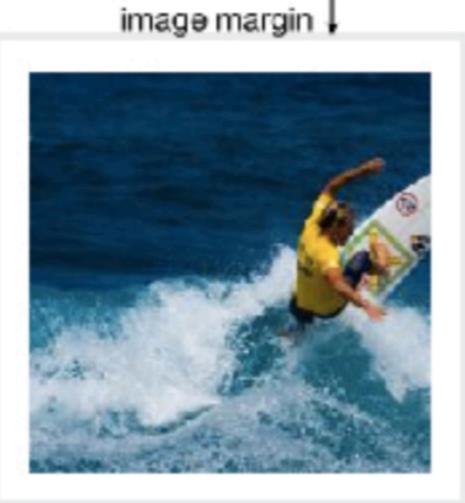
The Normal Document Flow



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent euismod ultrices ante, ac laoreet nulla vestibulum adipiscing. Nam quis justo in augue auctor imperdiet. Curabitur aliquet orci sit amet est posuere consectetur. Fusce nec leo ut massa viverra venenatis. Nam accumsan libero a elit aliquet quis ullamcorper arcu tincidunt. Praesent purus turpis, consectetur quis congue vel, pulvinar at lorem. Vivamus varius condimentum dolor, quis ultricies ipsum porta quis.

Right-Floating Image

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent euismod ultrices ante, ac laoreet nulla vestibulum adipiscing. Nam quis justo in augue auctor imperdiet. Curabitur aliquet orci sit amet est posuere consectetur. Fusce nec leo ut massa viverra venenatis. Nam accumsan libero a elit aliquet quis ullamcorper arcu tincidunt. Praesent purus turpis, consectetur quis congue vel, pulvinar at lorem. Vivamus varius condimentum dolor, quis ultricies ipsum porta quis.



Right-Floating Image

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent euismod ultrices ante, ac laoreet nulla vestibulum adipiscing. Nam quis justo in augue auctor imperdiet. Curabitur aliquet orci sit amet est posuere consectetur. Fusce nec leo ut massa viverra venenatis. Nam accumsan libero a elit aliquet quis ullamcorper arcu tincidunt. Praesent purus turpis, consectetur quis congue vel, pulvinar at lorem. Vivamus varius condimentum dolor, quis ultricies ipsum porta quis.



Если текста больше, то абзац будет обтекать картинку:

Right-Floating Image

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent euismod ultrices ante, ac laoreet nulla vestibulum adipiscing. Nam quis justo in augue auctor imperdiet. Curabitur aliquet orci sit amet est posuere consectetur. Fusce nec leo ut massa viverra venenatis. Nam accumsan libero a elit aliquet quis ullamcorper arcu tincidunt. Praesent purus turpis, consectetur quis congue vel, pulvinar at lorem. Vivamus varius condimentum dolor, quis ultricies ipsum porta quis.



Девять правил float-элементов:

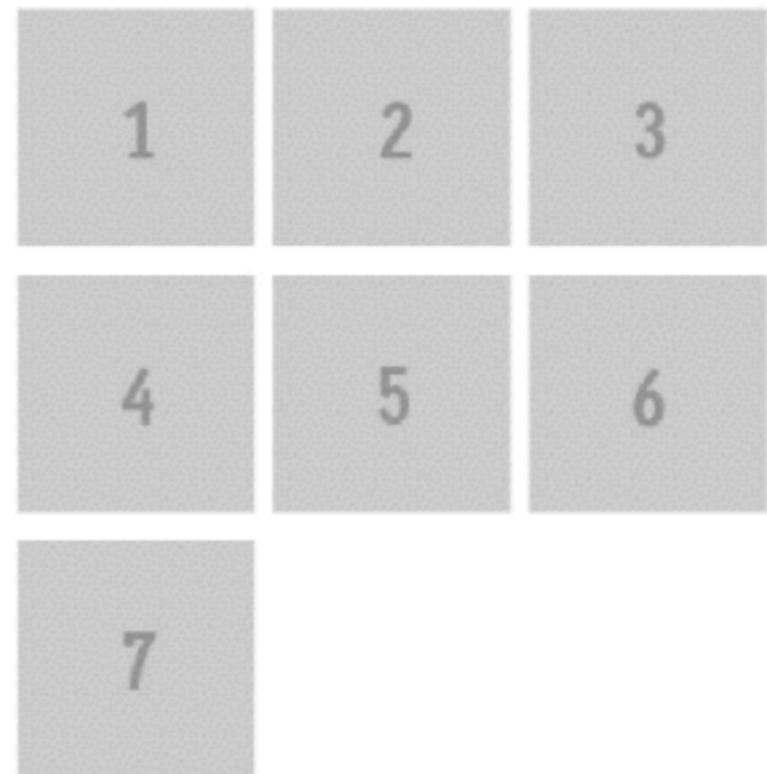
1. Плавающие элементы не могут выходить за край своего контейнера-родителя.
2. Каждый плавающий элемент будет отображаться справа или ниже от предыдущего при `float:left`, либо слева и ниже, при `float:right`.
3. Блок с `float:left` не может быть правее, чем блок с `float:right`.
4. Плавающий элемент не может выходить за пределы верхней границы своего контейнера.
5. Плавающий элемент не может располагаться выше, чем родительский блок или предыдущий плавающий элемент.
6. Плавающий элемент не может располагаться выше, чем предыдущая строка inline-элементов
 7. Плавающий блок должен быть расположен как можно выше.
8. Один плавающий элемент, следующий за другим, не может выходить за пределы своего контейнера – происходит перенос на следующую строку.
9. Блок с `float:left` должен быть расположен как можно левее, а с `float:right` – как можно правее.

Пример галереи

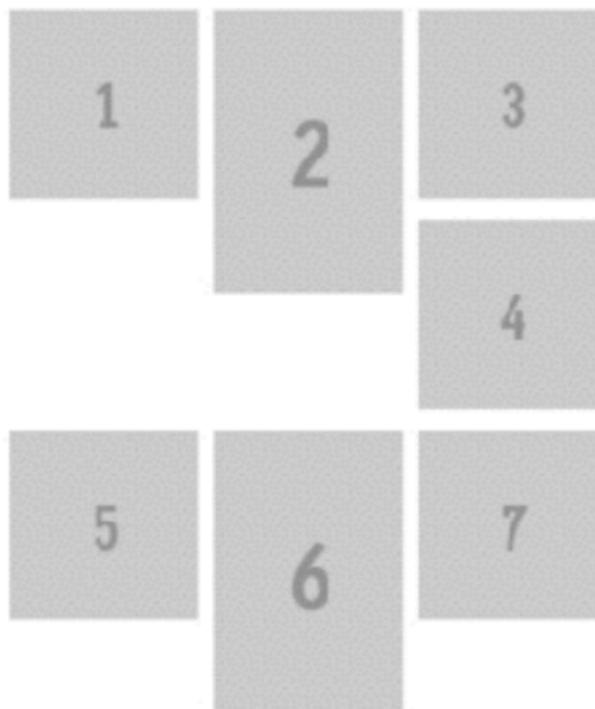
По умолчанию каждый элемент списка появится с новой строки. Если применить к каждому float:left, изображения встанут в один ряд с переносом строки:

```
<ul>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
</ul>
```

```
li {
  float: left;
  margin: 4px;
}
```

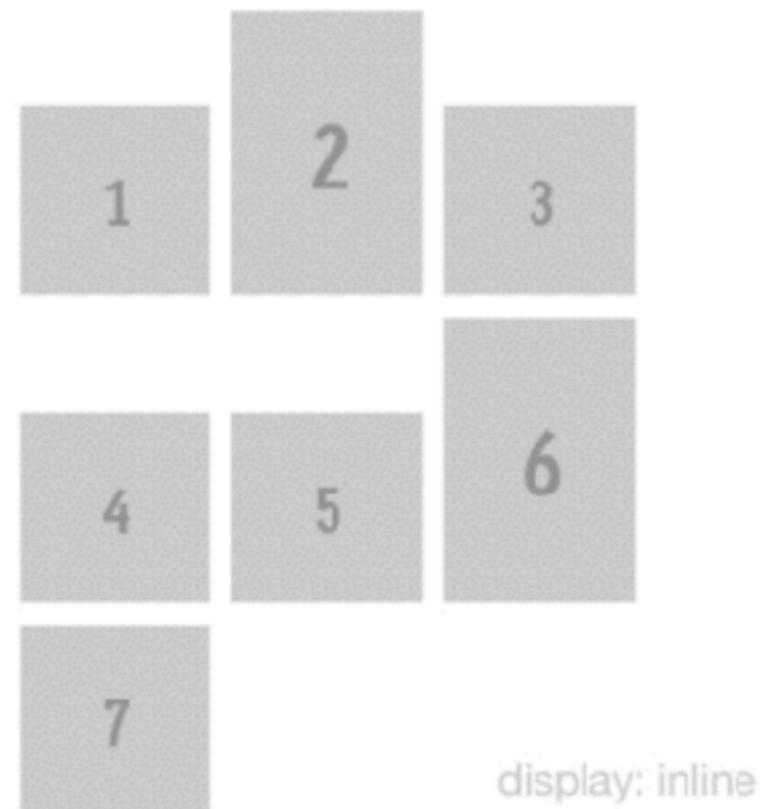


Если изображения разной высоты

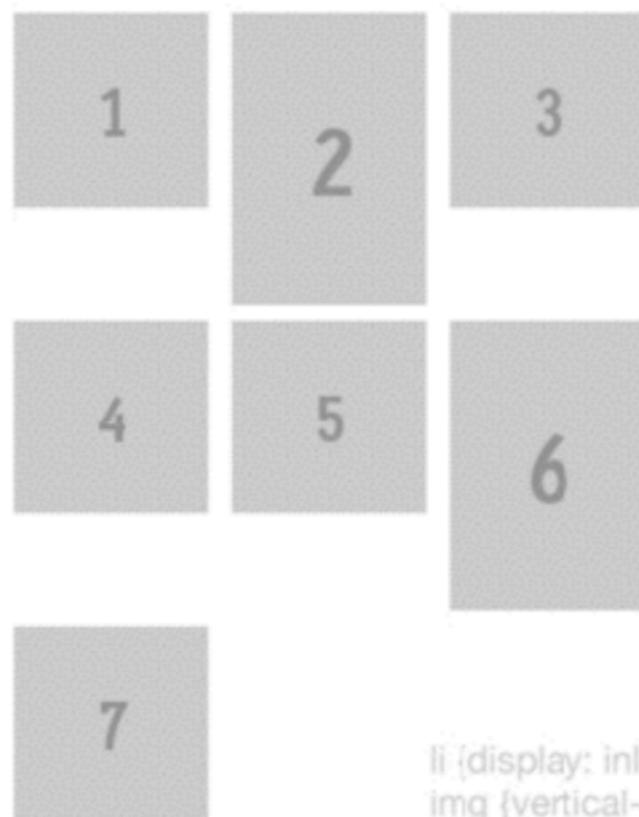


Добавим элементам списка отображение в одну строку

```
li {  
    display: inline;  
}
```



Выравниваем по вертикали



Свойство clear

```
clear: left | right | both;
```

```
li {display: inline}  
img {vertical-align: top}
```

CSS-текст

Горизонтальное выравнивание text-align

Свойство выравнивает строки текста внутри блока по ширине относительно его границ. Применяется только к блочным элементам, например, абзацам. Наследуется.

<https://codepen.io/anon/pen/OqVzzx>

left Выравнивание по левому краю элемента. Значение по умолчанию для языков, в которых чтение происходит слева-направо.

Right Выравнивание по правому краю элемента.

Center Выравнивание по центру элемента, управляет выравниванием содержимого, а не самих элементов. Центрирует каждую строчку текста элемента.

justify Выравнивание по ширине элемента. В выровненном по ширине тексте оба конца строки прижимаются к левому и правому краям родительского элемента. Пробелы между словами и буквами распределяются таким образом, чтобы длина всех строк была равна. Разные браузеры могут увеличить как отступы между словами, так и интервалы между буквами.

initial Устанавливает значение свойства в значение по умолчанию.

inherit Наследует значение свойства от родительского элемента.

```
p {text-align: left; }  
p {text-align: right; }  
p {text-align: center; }  
p {text-align: justify; }
```

Отступ text-indent

Устанавливает отступ (выступ) в первой строке элемента, создавая иллюзию структурированного текста. Применяется к любому блочному элементу, значение по умолчанию 0. Если в первой строке блочного элемента присутствует изображение, то оно сдвинется вместе с остальным текстом. Наследуется.

длина Значение задаётся в единицах длины em, px и т.д. Смещает первую строку текста. Можно использовать как положительные, так и отрицательные значения (см. пример ниже).

% Величина смещения задаётся в процентах, вычисляется относительно ширины родительского блока.

initial Устанавливает значение свойства в значение по умолчанию.

inherit Наследует значение свойства от родительского элемента.

```
p {text-indent: 5px;}  
p {text-indent: 2%;}
```

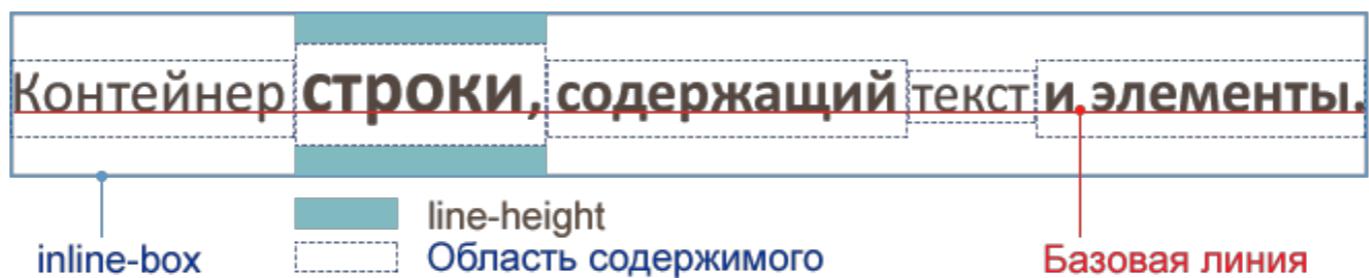
Отступ первой строки этого абзаца задан с помощью свойства text-indent: 25px.
Остальные строки абзаца не будут иметь отступ, независимо от длины абзаца.

Для этого абзаца задан выступ (отрицательный отступ) первой строки text-indent: -25px.
Остальные строки абзаца не будут иметь отступ, независимо от длины абзаца.



Отрицательный отступ text-indent: -30px также можно использовать для выделения первой строки абзаца с помощью изображений.

Высота строки line-height



длина Значение задаётся в единицах длины, создавая фиксированное значение высоты строки. Если задать значение меньше единицы, слова в смежных строках будут находить друг на друга.

% Вычисляется относительно текущего значения свойства font-size.

число Целое или дробное числовое значение, на которое будет умножен текущий размер шрифта.

normal Значение браузера по умолчанию.

initial Устанавливает значение свойства в значение по умолчанию.

inherit Наследует значение свойства от родительского элемента.

```
h1 {line-height: 20px;}  
h1 {line-height: 200%;}  
h1 {line-height: 1.2;}  
h1 {line-height: normal;}
```

Межстрочный интервал позволяет управлять расстоянием между строками текста. В данном примере при размере шрифта `{font-size: 1em}` задан межстрочный интервал `{line-height: 0.8em}`.

Для данного абзаца при размере шрифта `{font-size: 1em}` задан межстрочный интервал `{line-height: 1}`.

В данном абзаце при размере шрифта `{font-size: 1em}` установлен межстрочный интервал `{line-height: normal}`.

Оптимальным значением межстрочного интервала при размере шрифта `{font-size: 1em}` будет `{line-height: 1.4em}`.

Вертикальное выравнивание vertical-align

Применяется только к строчным элементам, к изображениям и полям форм. Не выравнивает содержимое блочного элемента. Не наследуется.

baseline Выравнивает базовую линию элемента по базовой линии его родителя, совмещая среднюю линию элемента со средней линией родительского элемента.

sub Делает элемент подстрочным (аналогично с тегом `<sub>`). Величина понижения элемента может меняться в зависимости от браузера пользователя.

super Делает элемент надстрочным (аналогично с тегом `<sup>`). При этом значения `sup` и `super` не меняют размер шрифта, по умолчанию текст надстрочного и подстрочного элемента имеет такой же размер, как и текст родительского элемента.

top Верхний край элемента совмещается с верхним краем самого высокого элемента в линии.

middle Средняя линия элемента (обычно изображения) совмещается с линией, проходящей через середину родительского элемента.

bottom Нижний край элемента совмещается с нижним краем самого низкого элемента в линии.

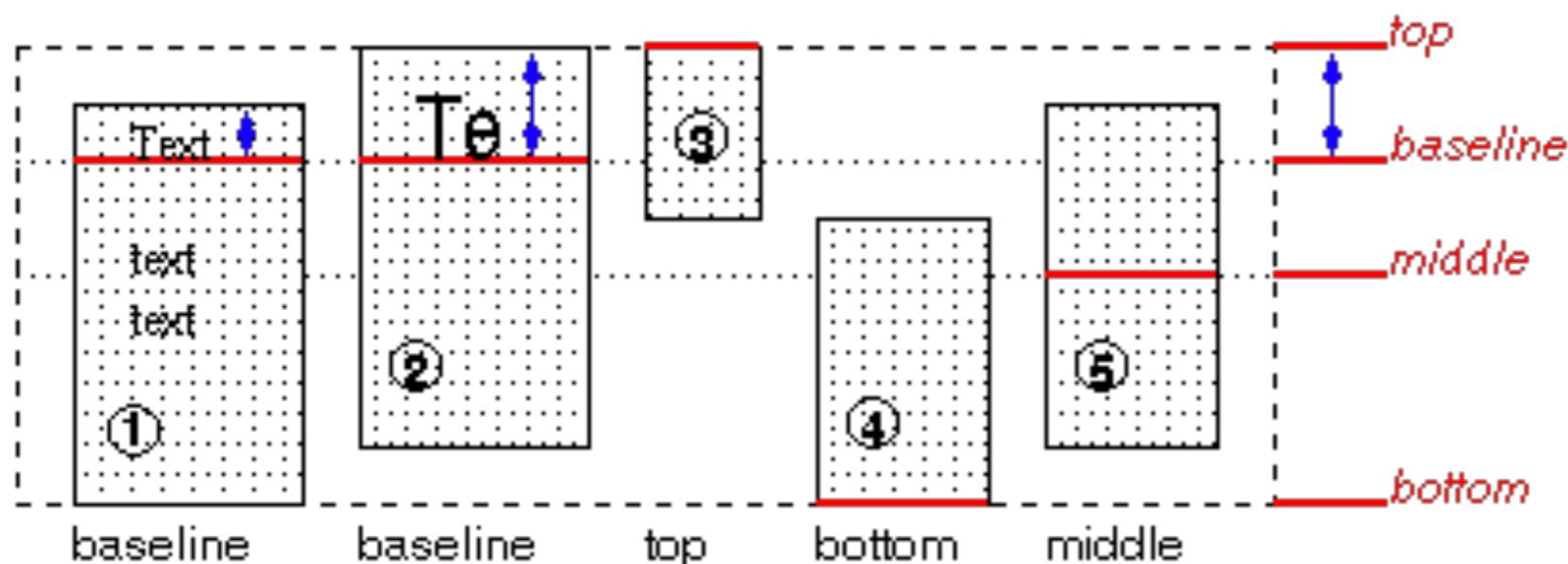
длина Устанавливает значение в единицах длины, перемещая элемент на заданное расстояние.

% Не позволяет устанавливать middle, вычисляется как часть line-height элемента, а не его родителя, т.е. если установить значение vertical-align, равное 50% для элемента с line-height равным 20px, то базовая линия элемента поднимется на 10px.

initial Устанавливает значение свойства в значение по умолчанию.

inherit Наследует значение свойства от родительского элемента.

```
img {vertical-align: baseline;}  
img {vertical-align: sub;}  
img {vertical-align: super;}  
img {vertical-align: top;}  
img {vertical-align: text-top;}  
img {vertical-align: middle;}  
img {vertical-align: bottom;}  
img {vertical-align: text-bottom;}  
img {vertical-align: 5px;}  
img {vertical-align: 100%;}
```



Обрезка строки `text-overflow`

Свойство позволяет ограничивать длину текста в случае, когда он не умещается в контейнер, визуально обрезая его или отображая многоточием. Свойство работает только при задании следующих условий: должна быть определена ширина контейнера, элемент должен иметь значения `overflow: hidden` и `white-space: nowrap`. Не наследуется.

clip Значение по умолчанию. Текст обрезается в пределе области содержимого, при этом может отобразиться лишь часть символа.

Ellipsis Замещает текст, не уместившийся в блок, с помощью многоточия.

initial Устанавливает значение свойства в значение по умолчанию.

inherit Наследует значение свойства от родительского элемента.

```
p {  
width: 300px;  
overflow: hidden;  
white-space: nowrap;  
text-overflow: ellipsis;  
}
```

Текст обрезан в пределах области содержим

Текст в блоке обрезан с помощью многото...

Перенос внутри слов word-break

В обычной ситуации слова переносятся на другую строку в местах пробелов или дефисов («мягкий перенос»), или же в случае принудительного переноса с использованием элемента `
`. Данное свойство позволяет установить правила переноса внутри слов для того, чтобы длинные строки полностью заполняли пространство внутри контейнера. Не используется для СЖК-языков (китайский-японский-корейский). Наследуется.

<https://codepen.io/Aortan/pen/bqNYQV>

normal Значение по умолчанию. Текст в контейнере отображается в привычном для нас виде в соответствии с правилами, установленными в языке.

break-all Пробелы в словах могут быть оформлены между любыми двумя буквами.

keep-all Запрещает разрывы между парами букв слова.

initial Устанавливает значение свойства в значение по умолчанию.

inherit Наследует значение свойства от родительского элемента.

```
p {word-break: normal;}  
p {word-break: break-all;}  
p {word-break: keep-all;}
```

Hello! This is a veryvery
veryveryveryverylongw
ord.
word-break: break-all

Hello! This is a
veryveryveryveryveryverylongword.
word-break: normal

Hello! This is a
veryveryveryveryveryverylongword.
word-break: keep-all

Перенос слов в строке word-wrap

Применяется в случаях, когда текст не умещается в свой контейнер и выходит за его границу. Позволяет разрывать длинные слова и переносить их на следующую строку. Наследуется. Срабатывает лишь в случае, если значение `white-space` разрешает разрыв строк.

normal Значение по умолчанию. Слова разрываются только по правилам, принятым в языке.

break-word Разрыв слов и перенос на следующую строку происходит в месте границы контейнера.

initial Устанавливает значение свойства в значение по умолчанию.

inherit Наследует значение свойства от родительского элемента.

```
p {word-wrap: normal;}  
p {word-wrap: break-word;}
```

Word-wrap

<https://codepen.io/elvirasantos/pen/YXqqKM?page=2>

Supercalifragilisticexpialidocious

Supercalifragili
sticexpialidoci
ous

Центрирование горизонтальное и вертикальное

В CSS есть всего несколько техник центрирования элементов.

Горизонтальное

text-align

Для центрирования инлайновых элементов – достаточно поставить родителю
text-align: center (left/right);

```
1 <style>
2   .outer {
3     text-align: center;
4     border: 1px solid blue;
5   }
6 </style>
7
8 <div class="outer">Текст</div>
```

Текст

Для центрирования блока это уже не подойдёт, свойство просто не подействует. Например:

```
1 <style>
2   .outer {
3     text-align: center;
4     border: 1px solid blue;
5   }
6   .inner {
7     width: 100px;
8     border: 1px solid red;
9   }
10 </style>
11
12 <div class="outer">
13   <div class="inner">Текст</div>
14 </div>
```

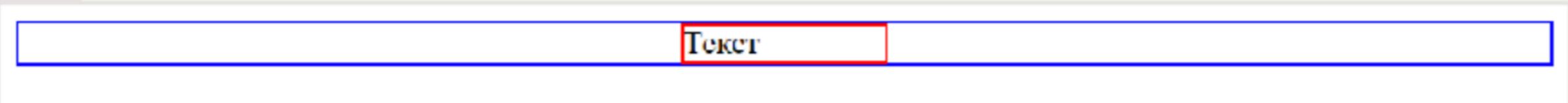
Текст

margin: auto

<https://codepen.io/anon/pen/OqVQKe>

Блок по горизонтали центрируется margin: auto

```
1 <style>
2   .outer {
3     border: 1px solid blue;
4   }
5   .inner {
6     width: 100px;
7     border: 1px solid red;
8     margin: auto;
9   }
10 </style>
11
12 <div class="outer">
13   <div class="inner">Текст</div>
14 </div>
```



В отличие от width/height, значение auto для margin само не появляется.
Обычно margin равно конкретной величине для элемента, например 0 для DIV.
Нужно поставить его явно.

Вертикальное

Вертикальное же изначальное не было предусмотрено в спецификации CSS

position:absolute + margin

```
1 <style>
2   .outer {
3     position: relative;
4     height: 5em;
5     border: 1px solid blue;
6   }
7
8   .inner {
9     position: absolute;
10    top: 50%;
11    border: 1px solid red;
12  }
13 </style>
14
15 <div class="outer">
16   <div class="inner">Текст</div>
17 </div>
```

Текст

Конечно, это не совсем центр. По центру находится верхняя граница. Нужно ещё приподнять элемент на половину своей высоты.

Высота центрируемого элемента должна быть известна. Родитель может иметь любую высоту.

Если мы знаем, что это ровно одна строка, то её высота равна line-height.

Приподнимем элемент на пол-высоты при помощи margin-top:

```
1 <style>
2   .outer {
3     position: relative;
4     height: 5em;
5     border: 1px solid blue;
6   }
7
8   .inner {
9     position: absolute;
10    top: 50%;
11    margin-top: -0.625em;
12    border: 1px solid red;
13  }
14 </style>
15
16 <div class="outer">
17   <div class="inner">Текст</div>
18 </div>
```

Текст

transform: translate(-50%, -50%)

```
1 <style>
2   .outer {
3     position: relative;
4     height: 5em;
5     border: 1px solid blue;
6   }
7
8   .inner {
9     position: absolute;
10    top: 50%;
11    transform: translate(-50%, -50%);
12    border: 1px solid red;
13  }
14 </style>
15
16 <div class="outer">
17   <div class="inner">Tekst</div>
18 </div>
```

Tekst

Одна строка: line-height

Вертикально отцентрировать одну строку в элементе с известной высотой height можно, указав эту высоту в свойстве line-height:

```
1 <style>
2   .outer {
3     height: 5em;
4     line-height: 5em;
5     border: 1px solid blue;
6   }
7 </style>
8
9 <div class="outer">
10  <span style="border:1px solid red">Текст</span>
11 </div>
```

Текст

Это работает, но лишь до тех пор, пока строка одна, а если содержимое вдруг переносится на другую строку, то контент начинает сдвигаться.

Если центрирование должно работать для любой высоты родителя и центрируемого элемента, то обычно используют таблицы или `display:table-cell` с `vertical-align`.

Если центрируются не-блочные элементы, например `inline` или `inline-block`, то `vertical-align` может решить задачу без всяких таблиц. Правда, понадобится вспомогательный элемент (можно через `:before`).

```
1 <style>
2 .before {
3   display: inline-block;
4   height: 100%;
5   vertical-align: middle;
6 }
7
8 .inner {
9   display: inline-block;
10  vertical-align: middle;
11 }
12 </style>
13
14 <div class="outer" style="height:100px; border:1px solid blue">
15   <span class="before"></span>
16   <span class="inner" style="border:1px solid red">
17     Центрированный<br>Элемент
18   </span>
19 </div>
```

Центрированный
Элемент

Перед центрируемым элементом помещается вспомогательный inline-блок `before`, занимающий всю возможную высоту.

Центрируемый блок выровнен по его середине.

```
1 <style>
2 .outer:before {
3   content: '';
4   display: inline-block;
5   height: 100%;
6   vertical-align: middle;
7 }
8
9 .inner {
10  display: inline-block;
11  vertical-align: middle;
12 }
13
14 /* добавим горизонтальное центрирование */
15 .outer {
16   text-align: center;
17 }
18 </style>
19
20 <div class="outer" style="height:100px; width: 100%; border:1px solid black">
21   <span class="inner" style="border:1px solid red">
22     Центрированный<br>Элемент
23   </span>
24 </div>
```

Центрированный
Элемент

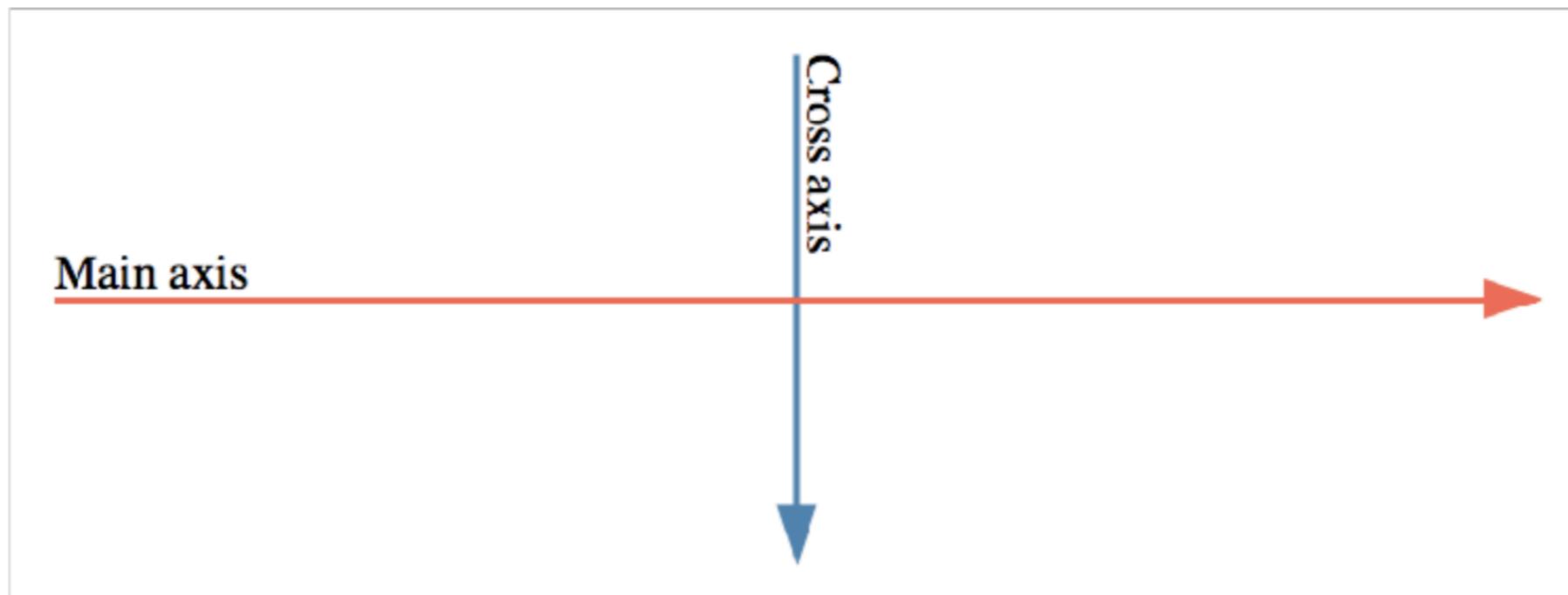
Display: flex

Flexbox – это новый способ располагать блоки на странице. Это технология, созданная именно для раскладки элементов, в отличие от float-ов.

С помощью **Flexbox** можно легко выравнивать элементы по горизонтали и по вертикали, менять направление и порядок отображение элементов, растягивать блоки на всю высоту родителя или прибивать их к нижнему краю.

Спецификация: <https://www.w3.org/TR/css-flexbox-1/>

Для начала надо знать, что flex-элементы располагаются по осям. По умолчанию элементы располагаются по горизонтали – вдоль **main axis** – главной оси.



При использовании **Flexbox** для внутренних блоков не работают float, clear и vertical-align

Curabitur ac
vestibulum mi

In viverra
dapibus

Fusce tincidunt
diam et

Nulla in dui vel
est

at diam in
lobortis

Родительскому элементу добавляем `display: flex;`.

Внутренние div-ы выстраиваются в ряд (вдоль главной оси) колонками одинаковой высоты, независимо от содержимого.

`display: flex;` делает все дочерние элементы резиновыми — `flex`, а не блочными — `block`, как было изначально.

Если родительский блок содержит картинки или текст без оберток, они становятся анонимными flex-элементами.

<https://codepen.io/lostsource41216364/pen/KGQjLW>

Curabitur ac
vestibulum mi

lenderit
id magna

In viverra
dapibus

Fusce
tincidunt
diam et

Nulla in
dui vel est



at diam in
lobortis

Свойство flex принимает два значения:

Flex - ведет себя как блочный элемент

Inline-flex - ведет себя как строчный элемент

flex – ведёт себя как блочный элемент. При расчете ширины блоков приоритет у раскладки (при недостаточной ширине блоков контент может вылезать за границы). **Ширина внутренних блоков может выходить за рамки родителя (если размер родителя фиксированно задан).**



inline-flex — ведёт себя как инлайн-блочный. Приоритет у содержимого (контент растягивает блоки до необходимой ширины, чтобы строчки, по возможности, поместились). **Ширина родительского блока подстраивается под ширину дочерних.**

Hello world

Curabitur ac vestibulum nisi

In viverra dapibus

Flex-direction

Направление раскладки блоков управляетя свойством **flex-direction**.

Возможные значения:

row — строка (значение по умолчанию);

row-reverse — строка с элементами в обратном порядке;

column — колонка;

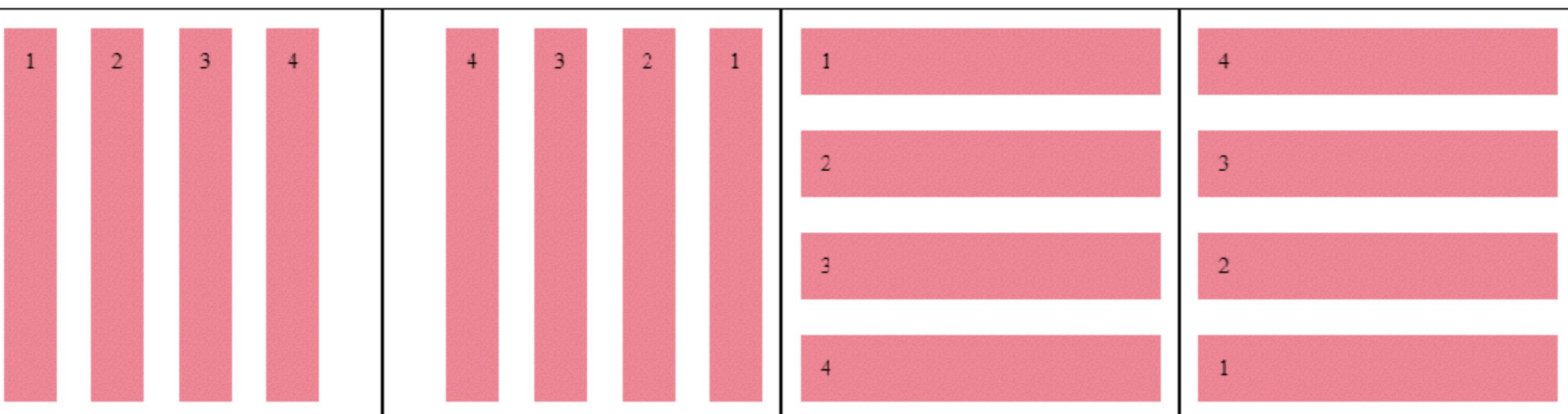
column-reverse — колонка с элементами в обратном порядке.

row и **row-reverse**

<https://jsfiddle.net/kisnows/1ryjp43c/>

<https://jsfiddle.net/kisnows/1ryjp43c/>

```
1 * .wrap {
2   display: flex;
3 }
4 * .container {
5   flex: 1;
6   display: flex;
7   border: 1px solid #000;
8   background: #FFF;
9 }
10 * .item {
11   padding: 14px;
12   margin: 12px;
13   background: #ED8890;
14 }
15 * .row {
16   flex-direction: row;
17 }
18 * .row-reverse {
19   flex-direction: row-reverse;
20 }
21 * .column {
22   flex-direction: column;
23 }
24 * .column-reverse {
25   flex-direction: column-reverse;
26 }
```



```
flex-grow: 0  
flex-shrink: 1
```

Свойство CSS `flex-grow` определяет, какую часть свободного пространства может занять контейнер, в соотношении с другими контейнерами.

Например, если все flex-блоки внутри flex-контейнера имеют `flex-grow:1`, то они будут одинакового размера. Если один из них имеет `flex-grow:2`, то он будет в 2 раза больше, чем все остальные.

<https://codepen.io/anon/pen/YgXagz>

Свойство `flex-shrink` определяет то, насколько flex-блок будет уменьшаться относительно соседних элементов внутри flex-контейнера в случае недостатка свободного места.

Например, если все flex-блоки внутри flex-контейнера имеют `flex-shrink:1`, то они будут одинакового размера. Если один из них имеет `flex-shrink:2`, то он будет в 2 раза меньше, чем все остальные.

<https://codepen.io/anon/pen/OqVvKb>

Flex-wrap

В одной строке может быть много блоков. Переносятся они или нет определяет свойство `flex-wrap`.

Возможные значения:

`nowrap` – блоки не переносятся (значение по умолчанию);

`wrap` – блоки переносятся;

`wrap-reverse` – блоки переносятся и располагаются в обратном порядке.

<https://jsfiddle.net/kisnows/3f7odnc6/>

```
27  * .nowrap {  
28      flex-wrap: nowrap;  
29  }  
30  
31  * .wrap {  
32      flex-wrap: wrap;  
33  }  
34  
35  * .wrap-reverse {  
36      flex-wrap: wrap-reverse;  
37  }  
38
```

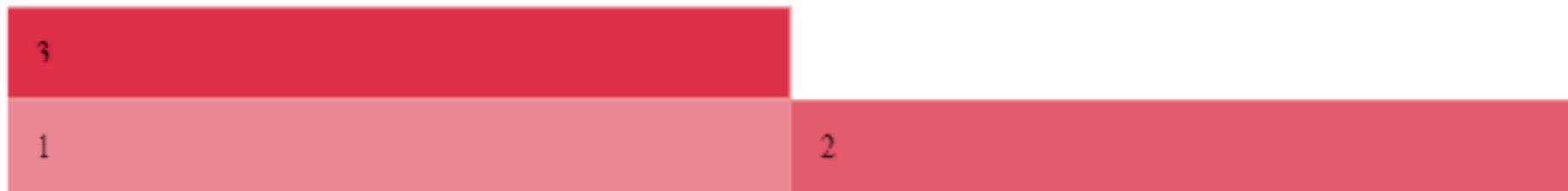
nowrap



wrap



wrap-reverse



Для короткой записи свойств flex-direction и flex-wrap существует свойство: flex-flow.

Возможные значения: можно задавать оба свойства или только какое-то одно.

Например:
flex-flow: column;

flex-flow: wrap-reverse;

flex-flow: column-reverse wrap;

<http://jsfiddle.net/mitrosin/w8rwmdww/>

```
1 .flex-row {  
2     -webkit flex flow: row wrap;  
3     -ms-flex-flow: row wrap;  
4     flex flow: row wrap;  
5 }  
6  
7 .flex-column {  
8     -webkit-flex-flow: column wrap;  
9     -ms-flex-flow: column wrap;  
10    flex-flow: column wrap;  
11 }  
12  
13 .flex-container {  
14     width: 60%;  
15     margin: 0 auto;  
16     border: solid 1px #000000;  
17     display: -webkit-flex;  
18     display: -ms-flexbox;  
19     display: flex;  
20 }  
21  
22 .flex item {  
23     border: solid 2px #FF0000;
```

Flex Flow : Row and Wrapping



Flex Flow : Row and Wrapping



Order

Для управления порядком блоков служит свойство `order`.

Возможные значения: числа.

Чтобы поставить блок самым первым, задайте ему `order: -1;`

<https://jsfiddle.net/blayderunner123/h09geqz2/>

<https://jsfiddle.net/blayderunner123/h09geqz2/>

```
153 .flex-item:nth-child(1){  
154     display:block;  
155     width:33.333%;  
156     text-align:center;  
157     background: gold;  
158     order:1;  
159 }  
160  
161 .flex-item:nth-child(2){  
162     display:block;  
163     width:33.333%;  
164     text-align:center;  
165     background: hotpink;  
166     order:2;  
167 }  
168  
169 .flex-item:nth-child(3){  
170     display:block;  
171     width:33.333%;  
172     text-align:center;  
173     background: orange;  
174     order:3;  
175 }  
176 }
```



Для выравнивания элементов есть несколько свойств:

justify-content, align-items и align-self.

justify-content и align-items применяются к родительскому контейнеру,

align-self – к дочерним.

justify-content отвечает за выравнивание по главной оси.

Возможные значения justify-content:

flex-start – элементы выравниваются от начала главной оси (значение по умолчанию);

flex-end – элементы выравниваются от конца главной оси;

center – элементы выравниваются по центру главной оси;

space-between – элементы выравниваются по главной оси, распределяя свободное место между собой;

space-around – элементы выравниваются по главной оси, распределяя свободное место вокруг себя.

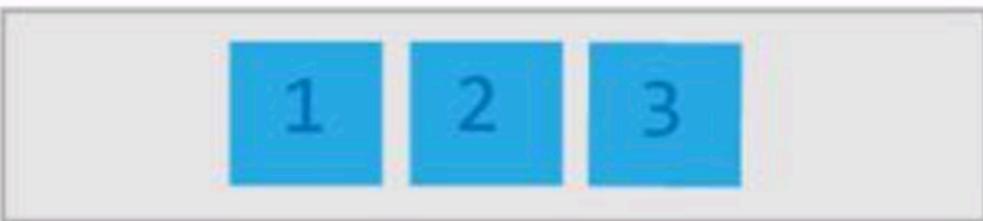
flex-start



flex-end



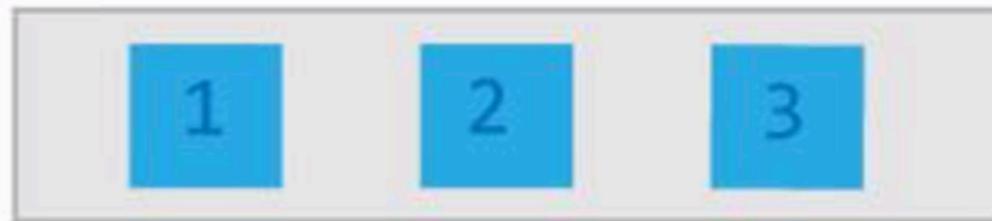
center



space-between



space-around



Align-items

align-items отвечает за выравнивание по перпендикулярной оси.

Возможные значения align-items:

flex-start – элементы выравниваются от начала перпендикулярной оси;

flex-end – элементы выравниваются от конца перпендикулярной оси; center – элементы выравниваются по центру;

baseline – элементы выравниваются по базовой линии;

stretch – элементы растягиваются, занимая все пространство по перпендикулярной оси (значение по умолчанию).

flex-start



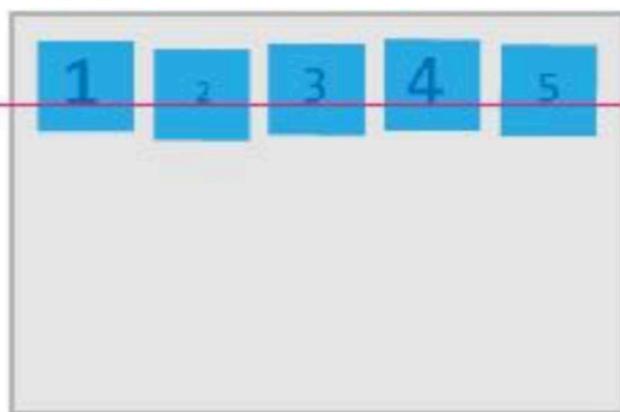
flex-end



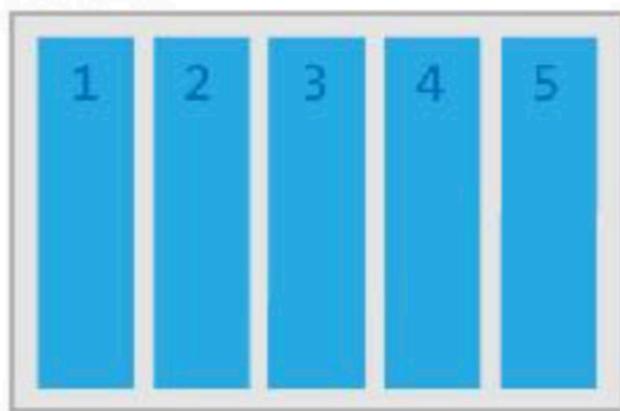
center



baseline



stretch



Align-Items

Align-self

align-self также отвечает за выравнивание по перпендикулярной оси, но задается отдельным flex-элементом.

Возможные значения align-self:

auto — значение по умолчанию. Означает, что элемент использует align-items родительского элемента;

flex-start — элемент выравнивается от начала перпендикулярной оси;

flex-end — элемент выравнивается от конца перпендикулярной оси;

center — элемент выравнивается по центру;

baseline — элемент выравнивается по базовой линии;

stretch — элемент растягивается, занимая все пространство по высоте.

Align-self



Align-content

Для управления выравниванием внутри многострочного flex-контейнера есть свойство align-content.

Возможные значения:

flex-start — элементы выравниваются от начала главной оси;

flex-end — элементы выравниваются от конца главной оси;

center — элементы выравниваются по центру главной оси;

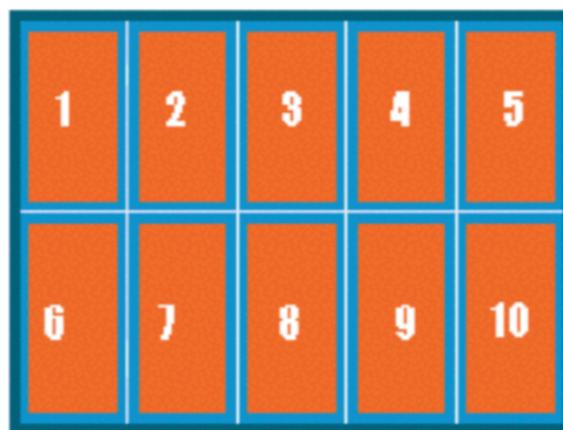
space-between — элементы выравниваются по главной оси, распределяя свободное место между собой;

space-around — элементы выравниваются по главной оси, распределяя свободное место вокруг себя;

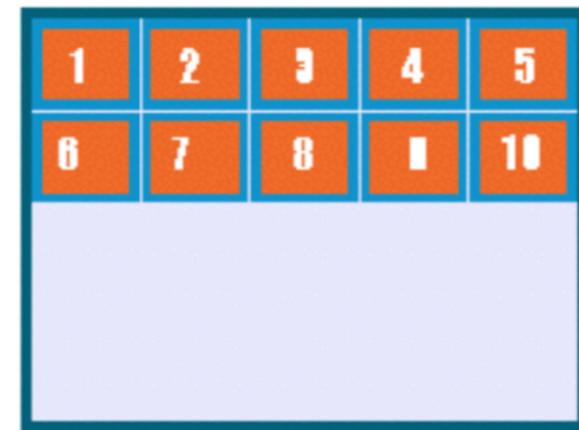
stretch — элементы растягиваются, заполняя всю высоту (значение по умолчанию).

align-content

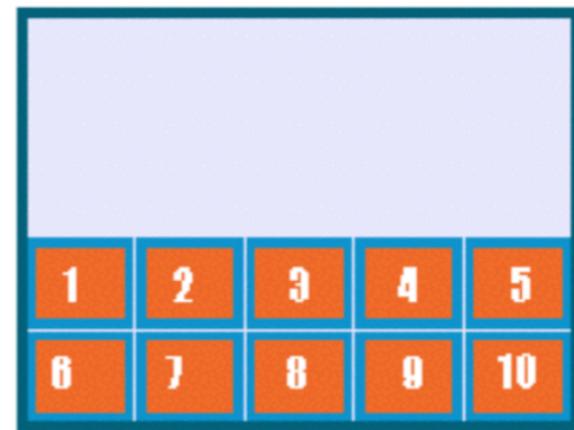
stretch
(по умолчанию)



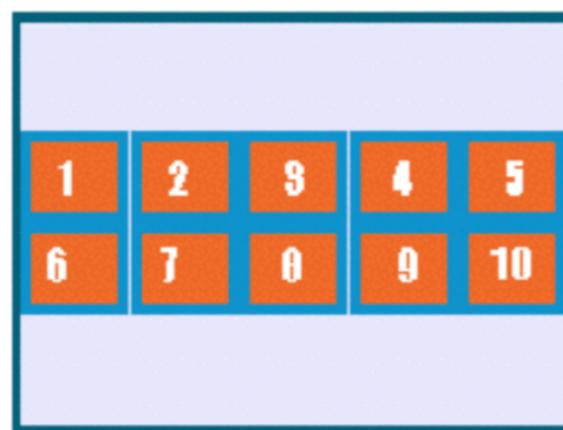
flex-start



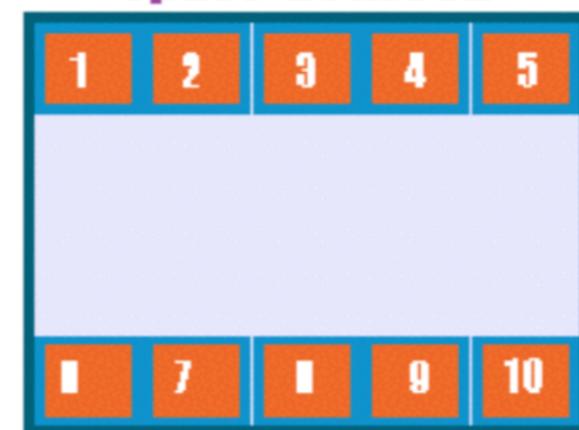
flex-end



center



space-between



space-around



Flex generator

<http://bennettfeely.com/flexplorer/>

<https://flexboxfroggy.com/#ru>

<https://cssgridgarden.com/#ru>