

Тема урока: Flexbox

Flexbox — это один из основных инструментов для создания адаптивных веб-страниц (наравне с [CSS Grid](#)), поэтому вот уже более 10 лет он остаётся мастхэв-технологией для верстальщиков и фронтенд-разработчиков.

В этом уроке мы расскажем о концепциях Flexbox, свойствах flex-контейнеров и flex-элементов и покажем их действие на простых примерах. А также поделимся лайфхаками для работы с Flexbox в Google Chrome и ссылками на бесплатные тренажёры.

Модуль Flexbox Layout (Flexible Box) (W3C Candidate Recommendation от октября 2017 г.) направлен на обеспечение более эффективного способа размещения, выравнивания и распределения пространства между элементами в контейнере, даже если их размер неизвестен и / или динамичен (Flex значит «гибкий»).

Основная идея flex layout состоит в том, чтобы дать контейнеру возможность изменять ширину / высоту его элементов (и порядок), чтобы наилучшим образом заполнить доступное пространство (главным образом, для отображения на всех типах устройств с любым размером экрана). Flex контейнер расширяет элементы, чтобы заполнить доступное свободное пространство, или сжимает их, чтобы предотвратить переполнение.

Наиболее важно то, что макет flexbox не зависит от направления, в отличие от обычных макетов (block на вертикальной основе и inline на горизонтальной основе). Хотя они хорошо работают для страниц, им не хватает гибкости (без каламбура :-)) для поддержки больших или сложных приложений (особенно когда речь идет об изменении ориентации, изменении размера, растяжении, сжатии и т.д.).

Предыстория:

Flexbox (от англ. flex — гибкий) — это модуль CSS, который позволяет удобно управлять расположением, порядком, размерами и отступами между элементами веб-страницы. Сайты, свёрстанные «флексами», получаются адаптивными, то есть выглядят хорошо на разных устройствах: ПК, ноутбуках, планшетах и смартфонах.

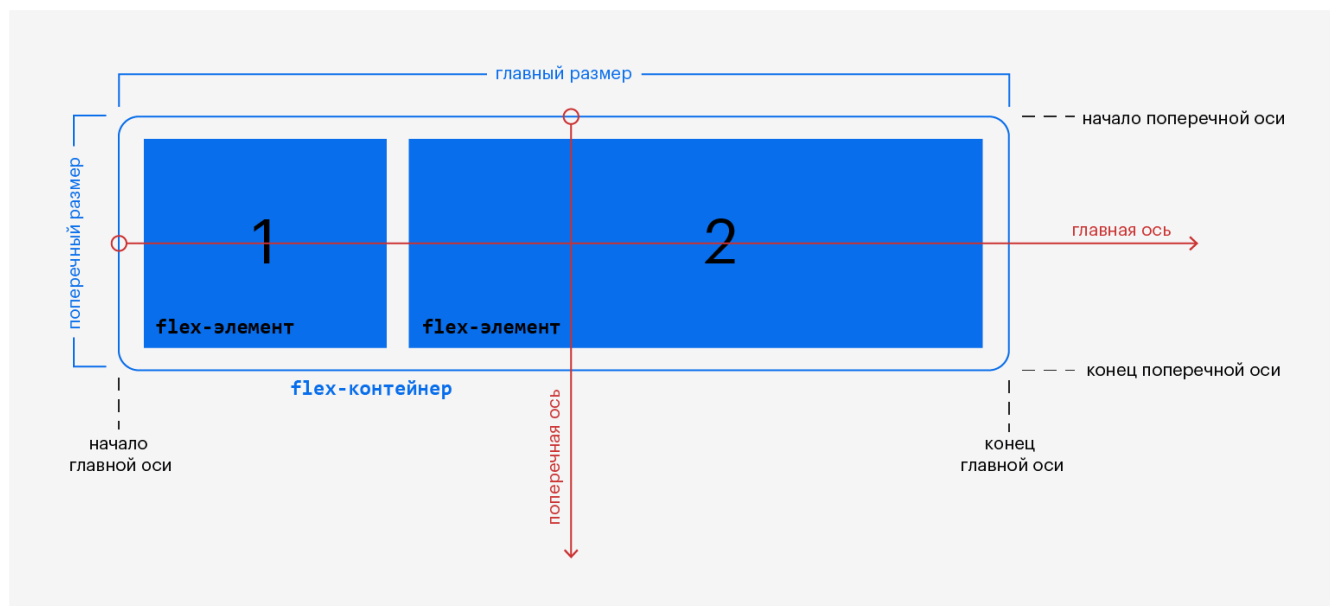
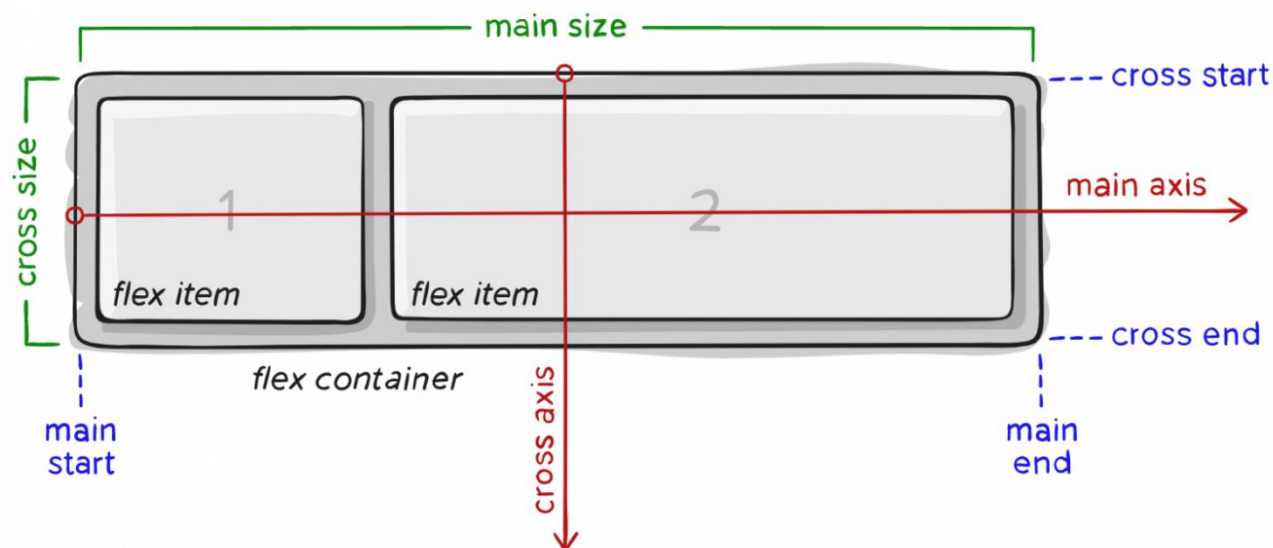
До появления Flexbox разработчики верстали веб-страницы с помощью таблиц, CSS-свойств [position](#), [float](#) и прочих инструментов, которые на самом деле для этого не предназначены. Например, [float](#) определяет, с какой стороны **текст**, а не группа блоков, будет обтекать элемент. Но так как до начала 2010-х других средств не было, разработчикам приходилось прибегать к подобным «костылям».

К счастью, в 2009 году инициативные разработчики решили навсегда избавиться от вёрстки таблицами и позиционирования и создали Flexbox. Сегодня он, как и [CSS Grid](#), является частью стандарта CSS3 и его не нужно подключать отдельно.

Основы и терминология:

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»).

Если «обычная» компоновка основана как на блочном, так и на inline направлениях, flex layout основана на «направлениях flex-flow». Пожалуйста, посмотрите на этот рисунок из спецификации, объясняющий основную идею гибкого макета.



Во Flexbox есть два вида свойств: одни применяются к flex-контейнеру, другие — к элементам, которые в нём расположены.

Flex-контейнер — это «коробка», в которой хранятся flex-элементы (flex item). Чтобы превратить элемент во flex-контейнер, нужно установить ему свойство `display: flex` или `display: inline-flex`.

Разница между `flex` и `inline-flex` в том, что в первом случае контейнер будет занимать всю ширину экрана (как блочный элемент), а во втором — лишь пространство, занимаемое его содержимым.

Flex-элементы (flex items) — это дочерние элементы flex-контейнера. Мы можем управлять их расположением, размерами и расстоянием между ними.

Главная ось (main axis) — направление, в котором располагаются flex-элементы.

Поперечная ось (cross axis) — ось, перпендикулярная главной оси.

Обратите внимание: направление главной и поперечной осей можно изменить с помощью свойства `flex-direction` (см. ниже).

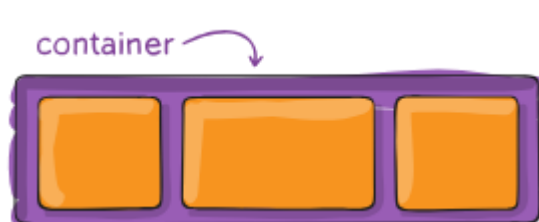
Главный размер (main size) — размер, соответствующий направлению главной оси.

Начало главной оси (main start) — точка, в которой начинается последовательность flex-элементов, расположенных по главной оси.

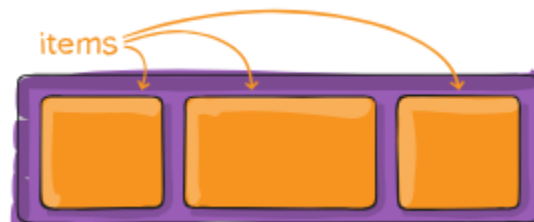
Конец главной оси (main end) — точка, в которой заканчивается последовательность flex-элементов, расположенных по главной оси.

Поперечный размер (cross size) — размер, соответствующий поперечной оси.

Свойства Flexbox

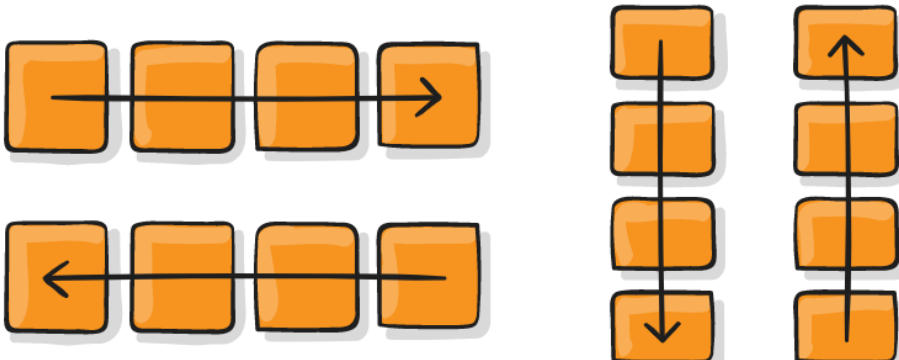


Родительский контейнер

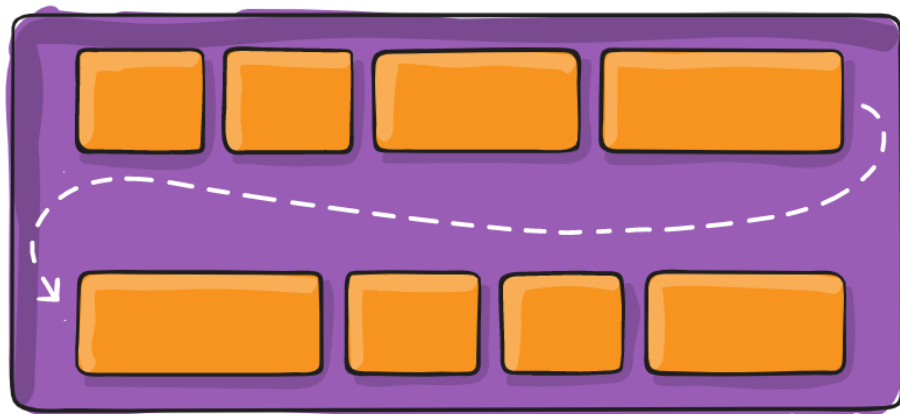


Дочерние элементы flex

Свойства для Родителя (flex контейнер)

Свойства	Описание
display	<p>Определяет flex контейнер; inline или block в зависимости от заданного значения. Включает flex контекст для всех потомков первого уровня.</p> <pre><code>.container { display: flex; /* or inline-flex */ }</code></pre> <p>Имейте в виду:</p> <p>Обратите внимание, что CSS-столбцы columns не влияют на flex контейнер.</p>
flex-direction	<p>Устанавливает основную ось, таким образом определяя направление flex элементов, помещаемых в flex контейнер. Flexbox — это (помимо дополнительной упаковки) концепция однонаправленного макета. Думайте о flex элементах, как о первичных раскладках в горизонтальных рядах или вертикальных столбцах.</p>  <pre><code>.container { flex-direction: row row-reverse column column-reverse; }</code></pre> <ul style="list-style-type: none">• row (по умолчанию): слева направо в ltr; справа налево в rtl• row-reverse справа налево ltr; слева направо в rtl• column: так же, как и row но сверху вниз• column-reverse: то же самое, row-reverse но снизу вверх

flex-wrap



По умолчанию гибкие элементы будут пытаться уместиться на одной строке. Вы можете изменить это и позволить элементам переходить на новую строку по мере необходимости с помощью этого свойства.

```
.container{  
  flex-wrap: nowrap | wrap | wrap-reverse;  
}
```

- **nowrap** (по умолчанию): все flex элементы будут в одной строке
- **wrap**: flex-элементы будут перенесены на несколько строк сверху вниз.
- **wrap-reverse**: flex-элементы будут перенесены на несколько строк снизу вверх.

flex-flow

Это сокращение для **flex-direction** и **flex-wrap** свойств, которые вместе определяют основные и поперечные оси flex контейнера. Значением по умолчанию является row nowrap.

```
flex-flow: <'flex-direction'> || <'flex-wrap'>
```

justify-content

flex-start



flex-end



center



space-between



space-around



space-evenly



Это свойство определяет выравнивание вдоль главной оси. Оно помогает распределить дополнительный остаток свободного пространства, когда-либо все flex элементы в строке негибкие, либо гибкие, но достигли своего максимального размера. Это также обеспечивает некоторый контроль над выравниванием элементов, когда они переполняют линию.

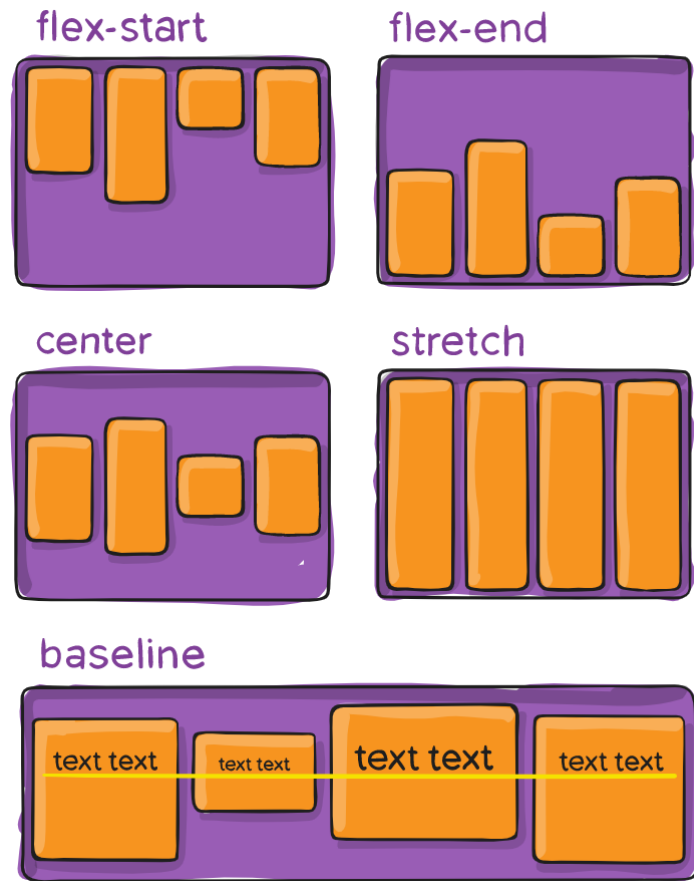
- **flex-start** (по умолчанию): элементы сдвинуты в начало flex-direction направления.
- **flex-end**: элементы сдвинуты ближе к концу flex направления.
- **start**: элементы сдвинуты к началу writing-mode направления.
- **end**: элементы сдвинуты в конце writing-mode направления.
- **left**: элементы сдвинуты по направлению к левому краю контейнера, если это не имеет смысла flex-direction, тогда он ведет себя как start.
- **right**: элементы сдвинуты по направлению к правому краю контейнера, если это не имеет смысла flex-direction, тогда он ведет себя как start.

- **center:** элементы центрированы вдоль линии
- **space-between:** элементы равномерно распределены по линии; первый элемент находится в начале строки, последний элемент в конце строки
- **space-around:** элементы равномерно распределены по линии с одинаковым пространством вокруг них. Обратите внимание, что визуально пространства не равны, так как все элементы имеют одинаковое пространство с обеих сторон. Первый элемент будет иметь одну единицу пространства напротив края контейнера, но две единицы пространства между следующим элементом, потому что у следующего элемента есть свой собственный интервал, который применяется.
- **space-evenly:** элементы распределяются таким образом, чтобы расстояние между любыми двумя элементами (и расстояние до краев) было одинаковым.

Обратите внимание, что поддержка браузером этих значений имеет свои нюансы. Например, `space-between` никогда не получал поддержку Edge, а `start / end / left / right` еще нет в Chrome. В MDN есть подробные графики. Самые безопасные значения это `flex-start`, `flex-end` и `center`.

Есть также два дополнительных ключевых слова, которые вы можете связать с этими значениями: `safe` и `unsafe`. Использование `safe` гарантирует, что как бы вы ни занимались этим типом позиционирования, вы не сможете расположить элемент таким образом, чтобы он отображался за пределами экрана (например, сверху) так, чтобы содержимое тоже не могло быть прокручено (это называется «потеря данных»).

align-items



Это свойство определяет поведение по умолчанию того, как flex элементы располагаются вдоль поперечной оси на текущей линии. Думайте об этом как о justify-content версии для поперечной оси (перпендикулярной главной оси).

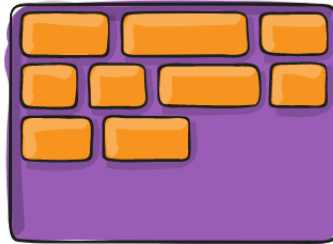
```
.container {  
  align-items: stretch | flex-start | flex-end | center |  
}
```

- **stretch** (по умолчанию): растягивать, чтобы заполнить контейнер (все еще соблюдаются min-width / max-width)
- **flex-start / start / self-start**: элементы размещаются в начале поперечной оси. Разница между ними невелика и заключается в соблюдении flex-direction правил или writing-mode правил.
- **flex-end / end / self-end**: элементы располагаются в конце поперечной оси. Разница опять-таки тонкая и заключается в соблюдении **flex-direction** или **writing-mode** правил.
- **center**: элементы центрированы по поперечной оси
- **baseline**: элементы выровнены, по их базовой линии

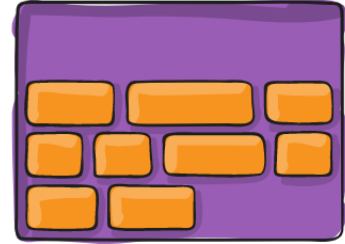
safe и **unsafe** ключевые слова модификаторов могут быть использованы в сочетании со всеми из этих ключевых слов (хотя это поддерживается не всеми браузерами), это помогает предотвратить выравнивание элементов таким образом, что содержание становится недоступным.

align-content

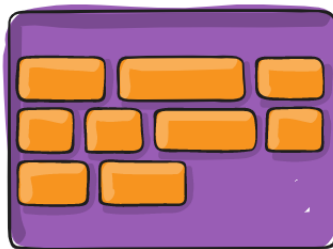
flex-start



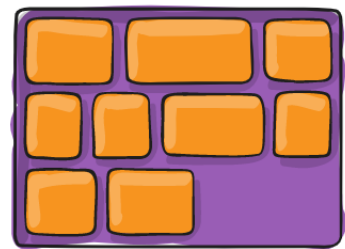
flex-end



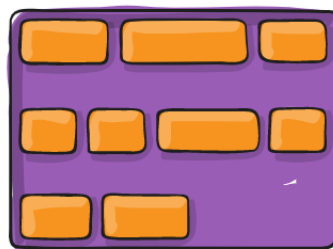
center



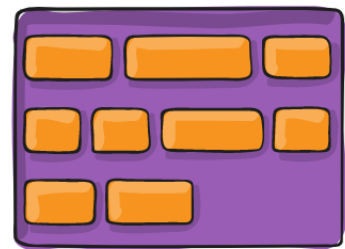
stretch



space-between



space-around



Это свойство выравнивает линии в пределах **flex** контейнера, когда есть дополнительное пространство на поперечной оси, подобно тому, как **justify-content** выравнивает отдельные элементы в пределах главной оси.

Примечание: это свойство не действует, когда есть только одна строка flex элементов.

```
.container {  
  align-content: flex-start | flex-end | center | space-between |  
}
```

- **flex-start / start:** элементы, сдвинуты в начало контейнера. Более поддерживаемый **flex-start** использует, **flex-direction** в то время как **start** использует **writing-mode** направление.

- **flex-end / end:** элементы, сдвинуты в конец контейнера. Более поддерживаемый **flex-end** использует **flex-direction** в то время как **end** использует **writing-mode** направление.
- **center:** элементы выровнены по центру в контейнере
- **space-between:** элементы равномерно распределены; первая строка находится в начале контейнера, а последняя — в конце
- **space-around:** элементы равномерно распределены с равным пространством вокруг каждой строки
- **space-evenly:** элементы распределены равномерно, вокруг них одинаковое пространство
- **stretch** (по умолчанию): линии растягиваются, чтобы занять оставшееся пространство

safe и **unsafe** ключевые слова модификаторов могут быть использованы в сочетании со всеми из этих ключевых слов (хотя это поддерживается не всеми браузерами), это помогает предотвратить выравнивание элементов таким образом, что содержание становится недоступным.

gap — это разрыв между отдельными элементами, строками или колонками внутри flex-контейнера. Значение по умолчанию — **none**; задаётся пикселями (px) или процентами.

**gap, row-gap,
column-gap**



Существует три версии свойства:

- **row-gap** — устанавливает разрыв между строками;
- **column-gap** — устанавливает разрыв между колонками;

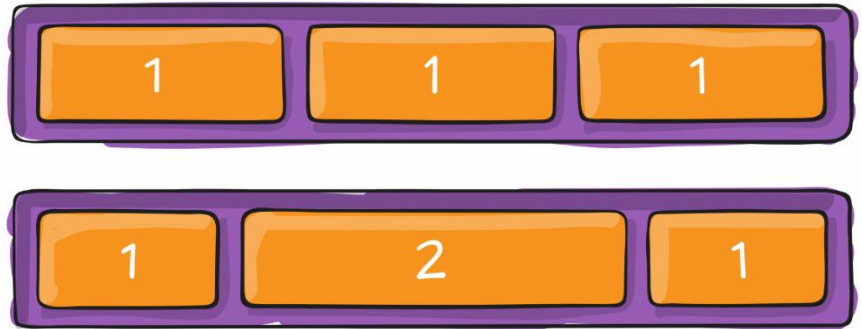
- **gap** — устанавливает разрыв между строками и колонками. При этом, если задать только значение, оно применяется и к строкам, и к колонкам.

```
.container {
  display: flex;
  ...
  gap: 10px;
  gap: 10px 20px; /* row-gap column gap */
  row-gap: 10px;
  column-gap: 20px;
}
```

Свойства для первых дочерних элементов(flex элементы)

Свойства	Описание
order	 <p>По умолчанию flex элементы располагаются в исходном порядке. Однако свойство order управляет порядком их появления в контейнере flex.</p>

```
.item {  
  order: <integer>; /* default is 0 */  
}
```



flex-grow

Это свойство определяет способность flex элемента растягиваться в случае необходимости. Оно принимает значение от нуля, которое служит пропорцией. Это свойство, какое количество доступного пространства внутри гибкого контейнера должен занимать элемент.

Если для всех элементов **flex-grow** установлено значение 1, оставшееся пространство в контейнере будет равномерно распределено между всеми дочерними элементами. Если один из дочерних элементов имеет значение 2, этот элемент займет в два раза больше места, чем остальные (или попытается, по крайней мере).

```
.item {  
  flex-grow: <number>; /* default 0 */  
}
```

Отрицательные числа не поддерживаются.

flex-shrink

Это свойство определяет способность гибкого элемента сжиматься при необходимости.

```
.item {  
  flex-shrink: <number>; /* default 1 */  
}
```

Отрицательные числа не поддерживаются.

flex-basis

Это свойство определяет размер элемента по умолчанию перед распределением оставшегося пространства. Это может быть длина (например, 20%, 5rem и т.д.) Или ключевое слово. Ключевое слово auto означает «смотри на мое width или height свойство». Ключевое слово content означает «размер на основе содержимого элемента» — это ключевое слово все еще не очень хорошо поддерживается, так что трудно проверить что для него используется max-content, min-content или fit-content.

```
.item {  
  flex-basis: <length> | auto; /* default auto */  
}
```

Если установлено значение 0, дополнительное пространство вокруг содержимого не учитывается. Если установлено значение auto, дополнительное пространство распределяется в зависимости от его flex-grow значения.

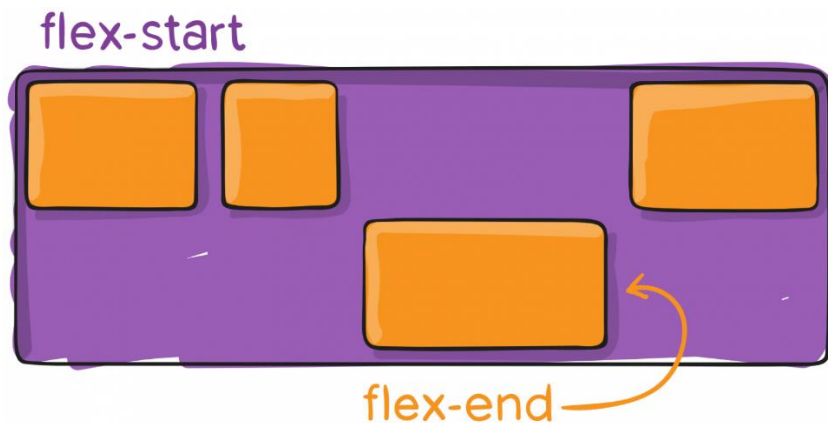
flex

Это сокращение для использования flex-grow, flex-shrink и flex-basis вместе. Второй и третий параметры (flex-shrink и flex-basis) являются необязательными. По умолчанию это 0 1 auto.

```
.item {  
  flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]  
}
```

Рекомендуется использовать это сокращенное свойство, а не устанавливать отдельные свойства. Это сокращение разумно устанавливает другие значения.

align-self



Это свойство позволяет переопределить выравнивание по умолчанию (или указанное с помощью align-items) для отдельных элементов flex.

Пожалуйста, смотрите align-items свойство, чтобы понять доступные значения.

```
.item {  
  align-self: auto | flex-start | flex-end | center |  
}
```

Обратите внимание что свойства float, clear и vertical-align не влияют на flex элементы.