



Flexbox

Оглавление

Основные преимущества flexbox	3
Поддержка браузерами	5
Основы flexbox-верстки	7
Свойства flex-контейнера	9
flex-flow	9
justify-content	11
align-items	13
align-content	15
Свойства flex-элементов	18
order	18
align-self	18
flex	19
Учимся играя	23
Очень полезные ссылки	25

Спецификация CSS Flexible Box Layout Module (в народе - Flexbox) позволяет контролировать размер, порядок и выравнивание элементов по нескольким осям, распределение свободного места между элементами и многое другое.

Основные преимущества flexbox

Flexbox обеспечивает весьма полезный набор свойств, которые позволяют заливать элементы в строке макета без необходимости их выравнивания или использования свойства inline-block. Гибкость flexbox-верстки позволяет автоматически настраивать ширину элементов, находящихся внутри flex-контейнера, что очень похоже на выравнивание на странице обтекаемых элементов с применением процентных значений.

Все блоки очень легко делаются “резиновым”, что уже следует из названия “flex”. Элементы могут сжиматься и растягиваться по заданным правилам, занимая нужное пространство.

Выравнивание по вертикали и горизонтали, базовой линии текста работает превосходно.

Расположение элементов в html не имеет решающего значения. Его можно поменять в CSS. Это особенно важно для некоторых аспектов responsive верстки.

Элементы могут автоматически выстраиваться в несколько строк/столбцов, занимая все предоставленное место.

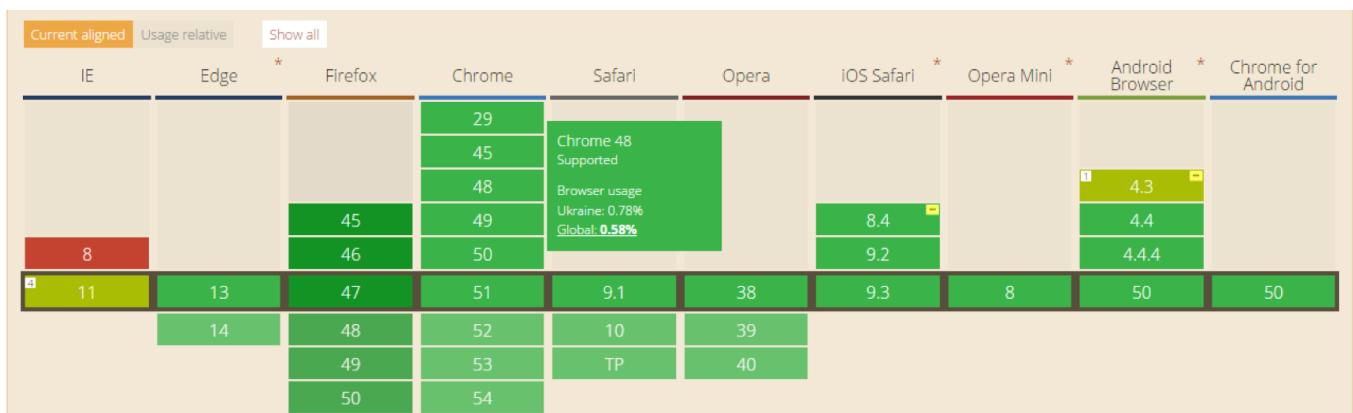
Множество языков в мире используют написание справа налево rtl (right-to-left), в отличии от привычного нам ltr (left-to-right). Flexbox адаптирован для этого. В нем есть понятие начала и конца, а не право и лево. Т.е. в браузерах с локалью rtl все элементы будут автоматически расположены в реверсном порядке.

Синтаксис CSS-правил очень прост и осваивается довольно быстро.

Поддержка браузерами

- 2008 – CSS Working Group обсуждает предложение “Flexible Box Model” на основе XUL (XML User Interface Language – язык разметки в приложениях Mozilla) and XAML (Extensible Application Markup Language – язык разметки в приложениях Microsoft).
 - 2009 – опубликован черновик “Flexible Box Layout Module”. Chrome и Safari добавляет частичную поддержку, пока Mozilla начинает поддерживать XUL-подобный синтаксис, известный как “Flexbox 2009”.
 - 2011 – Tab Atkins берется за развитие Flexbox и публикует 2 черновика. В этих черновиках синтаксис изменен значительно. Chrome, Opera и IE 10 внедряют поддержку этого синтаксиса. Он известен под названием “flexbox 2011”
 - 2012 – Синтаксис снова немного изменен и уточнен. Спецификация переходит в статус Candidate Recommendation и известна под названием “flexbox 2012”. Opera внедряет беспрефиксную поддержку, Chrome поддерживает текущую спецификацию с префиксами, а Mozilla без них, IE10 добавляет поддержку устаревающего “flexbox 2011” синтаксиса.
 - 2014 – все новые браузеры поддерживают последнюю спецификацию (включая IE 11)

На данный момент (2016 год) поддержка браузерами flexbox выглядит следующим образом, согласно caniuse.com



Основы flexbox-верстки

Суть flexbox-верстки довольно проста. Для работы с ней необходимы лишь два компонента:

1. **Flex-контейнер.** Любой HTML-элемент может быть flex-контейнером, но обычно им становится div или какой-либо другой структурный HTML-элемент. Элемент, который вы станете использовать в качестве контейнера, будет содержать дочерний и другие элементы, которые составляют второй компонент модели flexbox.
2. **Flex-элементы.** Элементы, вложенные непосредственно в flex-контейнер, называют flex-элементами. Каждый прямой потомок элемента контейнера автоматически становится flex-элементом. Вы можете поместить любой HTML-элемент внутрь flex-контейнера, и они не должны быть одного типа. Имейте также в виду, что только дочерние элементы flex-контейнера превращаются в flex-элементы.

Чтобы преобразовать любой HTML-элемент во flex-контейнер, используйте следующий код: `display: flex;`

Браузер отобразит эту серию div в виде блочных элементов, заполняя ими всю ширину внешнего элемента div и размещая, их друг за другом, как это показано на *рис. 1*



В элементах, для наглядности, картинки прибиты к низу блока, а цифра, обозначающая номер блока по порядку, расположена в центре (она центрируется на flexbox).

Забегая немного вперед, вы можете согласовать ширину всех элементов div внутри контейнера и заполнить контейнер, добавив свойство flex элементу и присвоив ему значение 1.



Свойства flex-контейнера

flex-flow

Свойство flex-flow позволяет выбрать направление, в котором элементы отображаются, а также указать, будут ли они переноситься на следующую строку.

Для свойства flex-flow необходимо задать два значения, разделяя их пробелом.

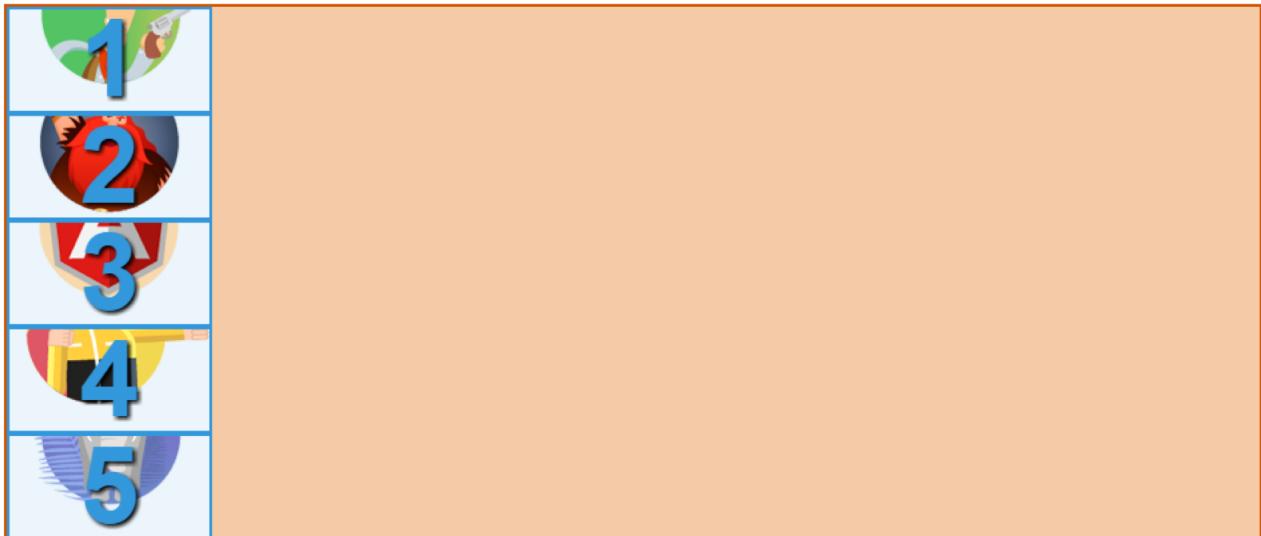
Первое значение свойства flex-flow определяет направление и может быть одним из следующих.

- **row** — значение по умолчанию. Оно отображает flex-элементы последовательно, рядом друг с другом.
- **row-reverse** также отображает flex-элементы рядом друг с другом, но обращает порядок их следования.
- **column** отображает flex-элементы в виде блоков, друг над другом. Это обычный режим отображения элементов div.
- **column-reverse** — это значение аналогично column, с той лишь разницей, что изменяет направление отображения элементов на противоположное.

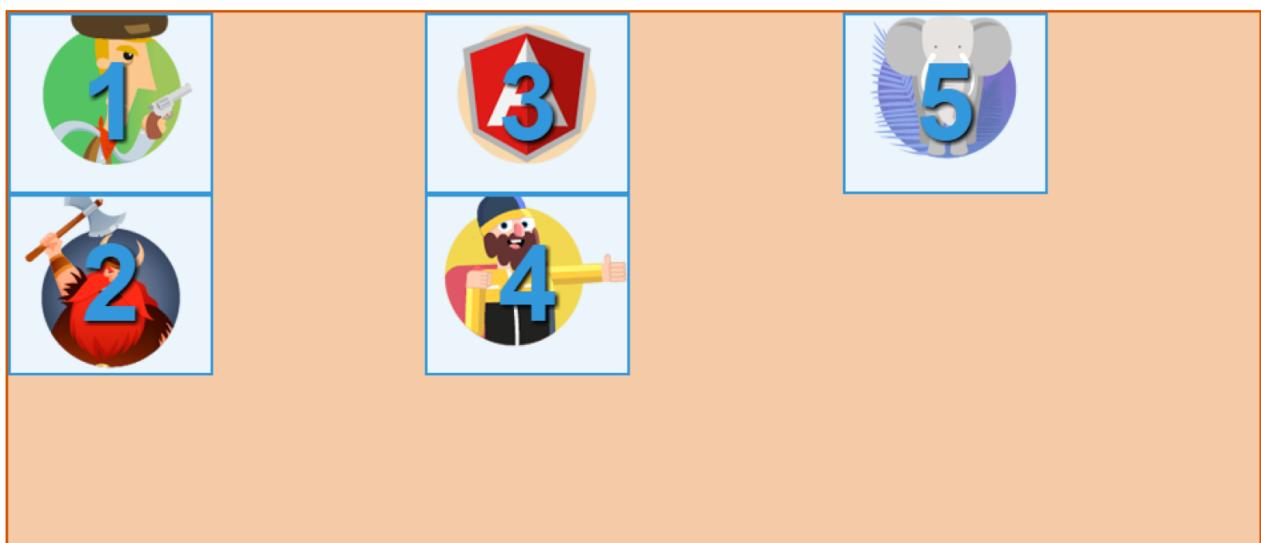
Второе свойство определяет, будут ли flex-элементы переноситься на новую строку (при значении row) или в новую колонку (при значении column). Свойство может принимать три возможных значения.

- **nowrap** — обычное поведение flex-элемента в flex-контейнере. Браузер будет отображать элементы в одной строке, независимо от того, насколько узким станет окно браузера

- **wrap** — позволяет объектам, которые не помещаются в контейнер по ширине, переноситься на новую строку (или в колонку)
- **wrap-reverse** — работает аналогично значению wrap, но элементы переносятся в обратном порядке.



`flex-flow: column nowrap;`



`flex-flow: column wrap;` (у элемента отсутствует `flex: 1`, иначе была бы предыдущая ситуация)

На самом деле, свойство flex-flow - это более лаконичная запись двух свойств: flex-direction и flex-wrap. Например, код:

```
flex-flow: row wrap;
```

аналогичен коду:

```
flex-direction: row;  
flex-wrap: wrap;
```

justify-content

Свойство justify-content определяет способ выравнивания по ширине flex-элементов в строке. Это свойство применимо только в тех случаях, если для flex-элементов указана ширина и если суммарная ширина элементов меньше, чем ширина flex-контейнера. Если вы не ограничиваете ширину flex-элементов, то свойство justify-content не окажет никакого эффекта.

Существует пять возможных значений свойства:

- **flex-start** — выравнивает элементы по левому краю строки.
- **flex-end** — выравнивает элементы по правому краю.
- **center** — выравнивает flex-элементы по центру.
- **space-between** — равномерно распределяет flex-элементы, разделяя пространство между ними поровну, поместив первый элемент у левого края строки, а правый — у правого
- **space-around** — равномерно распределяет оставшееся пространство между всеми элементами, добавляя его также и по левому и правому краям крайних элементов.



`justify-content: flex-start;`



`justify-content: center;`



`justify-content: space-between;`

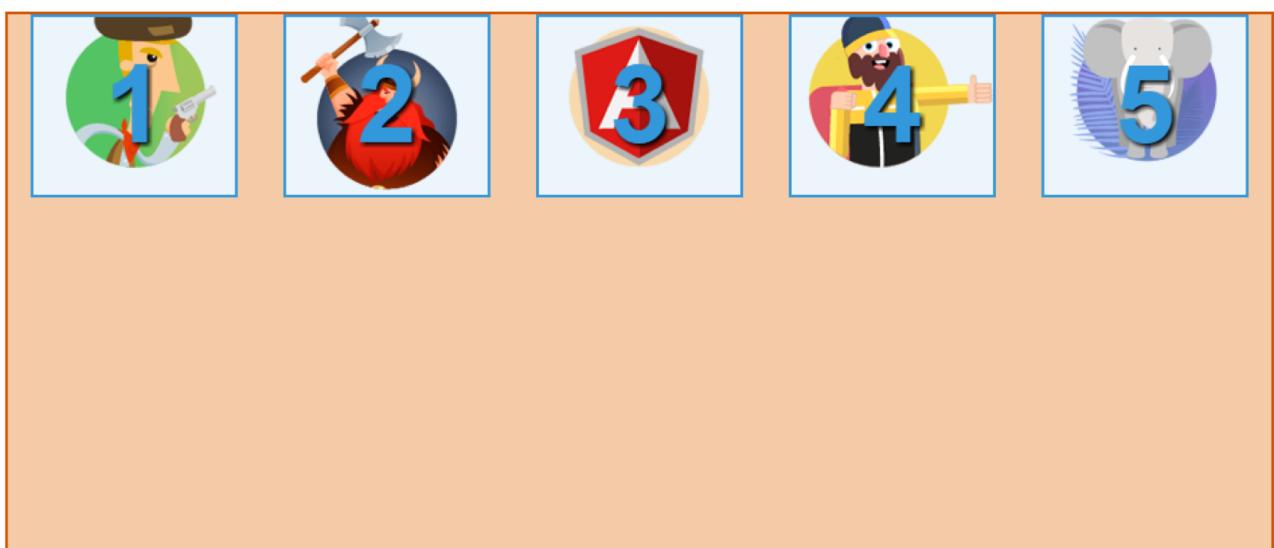


`justify-content: space-around;`

align-items

Свойство `align-items` определяет, как flex-элементы различной высоты будут выровнены по высоте строки в flex-контейнере. По умолчанию flex-элементы растягиваются до одинаковой высоты, чтобы заполнить контейнер.

- **flex-start** – выравнивает верхний край всех flex-элементов по верхнему краю.
- **flex-end** – выравнивает нижний край всех flex-элементов по нижнему краю контейнера.
- **center** – выравнивает вертикальный центр всех flex-элементов по вертикальному центру контейнера.
- **baseline** – выравнивает базовые линии всех flex-элементов по базовой линии первого элемента.
- **stretch** – обычное поведение flex-элементов. Растигивает каждый элемент по высоте контейнера, делая их высоты одинаковыми.



```
justify-content: space-around;  
align-items: flex-start;
```



```
justify-content: space-around;  
align-items: center;
```

align-content

Последнее свойство, которое можно применить к flex-контейнеру, – это align-content. Оно определяет, как браузер будет размещать flex-элементы, занимающие несколько строк. Это свойство работает только в случае соблюдения двух условий: во-первых, к flex-контейнеру должно быть применено значение wrap; во-вторых, flex-контейнер должен быть выше, чем строки flex-элементов.

Свойство align-content может принять одно из шести значений.

- **flex-start** — помещает строки flex-элементов у верхнего края flex-контейнера.
- **flex-end** — помещает строки flex-элементов у нижнего края flex-контейнера.
- **center** — выравнивает центры всех строк по вертикальному центру контейнера.

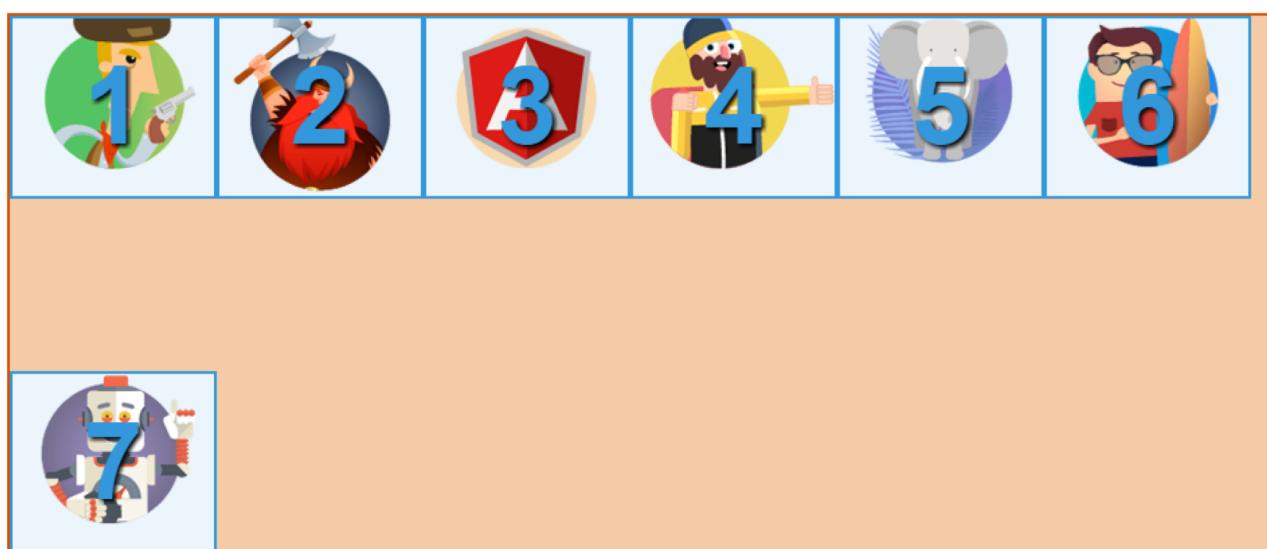
- **space-between** — равномерно распределяет дополнительное пространство по вертикали между строками, помещая верхнюю строку у верхнего края контейнера, а нижнюю — у нижнего.

- **space-around** — равномерно распределяет оставшееся пространство между всеми строками, добавляя его также и по верхнему и нижнему краям крайних строк.

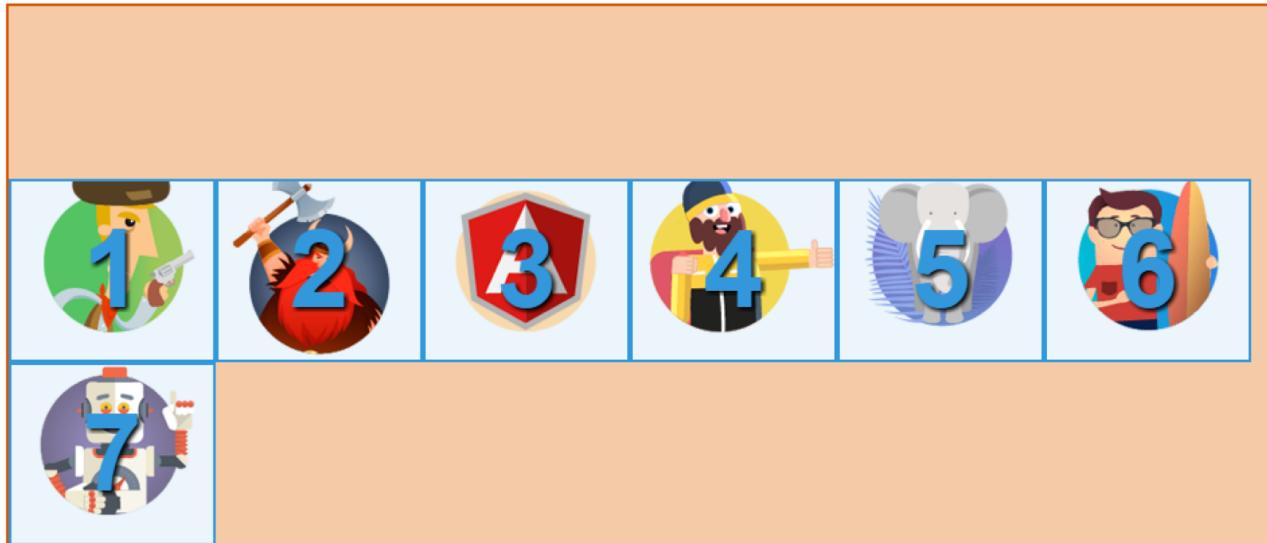
- **stretch** — обычное поведение строк flex-элементов.

Значение растягивает каждый элемент в строке таким образом, чтобы он соответствовал высоте других элементов в строке.

-



```
flex-flow: row wrap;  
align-content: space-between;
```



```
flex-flow: row wrap;  
align-content: flex-end;
```

Свойства flex-элементов

order

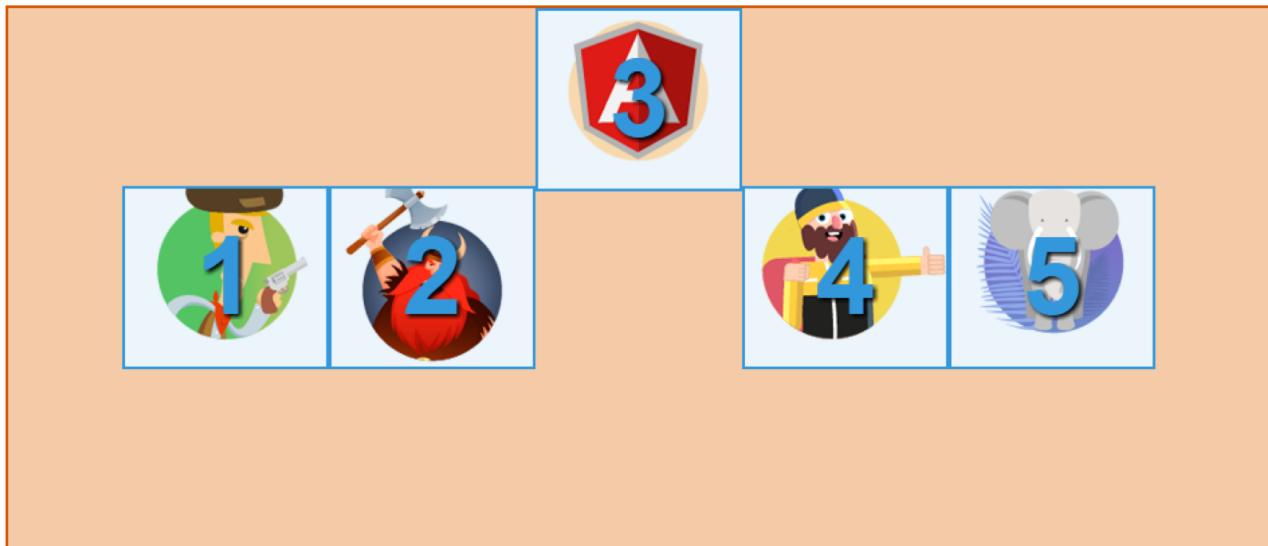
Позволяет назначать числовое значение приоритета flex-элемента, которое определяет его положение в строке (или колонке). Давайте для третьего (Angular) элемента изменим это свойство на 1. Числа, которые используются для указания значения свойства order, аналогичны значениям свойства z-index. Поэтому блок станет последним в списке.



align-self

Свойство align-self работает аналогично свойству align-items, которое используется для flex-контейнеров. Отличие в том, что align-item применяется ко всем flex-элементам в контейнере, а align-self — к отдельным. Свойство, примененное к элементу (неконтейнеру), замещает любое установленное значение свойства align-items. Давайте опять третий элемент выровняем не так как остальные.

```
.wrapper {  
  display: flex;  
  justify-content: center;  
  align-items: center;  
}  
  
.item:nth-child(3) {  
  align-self: flex-start;  
}
```



flex

Свойство **flex** обеспечивает гибкость flexbox-дизайнов. Оно является ключевым в управлении шириной flex-элементов; позволяет легко создавать «гибкие» колонки или изменять их ширину в соответствии с размером контейнера, даже если размер неизвестен или меняется динамически.

Свойство **flex** сочетает в себе сразу три flex-свойства.

Первое значение свойства **flex** – **число параметра flex-grow**, которое указывает на относительную ширину flex-элемента.

Например, если присвоить каждому из flex-контейнеров значение 1, мы сделаем их ширину одинаковой. Используемая нами величина – относительная, а не абсолютная. Если значение одного из элементов равно 2, то ширина элементов изменится. Ширина всех элементов равна 1, в то время как ширина третьего элемента равна 2.



Второе значение свойства `flex` — также **число, определяющее параметр `flex-shrink`**. Это свойство применимо, если ширина flex-контейнера меньше суммарной ширины flex-элементов внутри него. В этом случае свойство `flex-shrink` определяет, насколько узким может быть flex-элемент — то есть насколько он может сжаться. Это зависит от ширины flex-элементов, которая определяется последним значением свойства `flex`.

Последнее значение — свойство **`flex-basis`**, которое определяет базовую ширину flex-элемента. Вы можете использовать абсолютное значение или значение в процентах. Свойство `flex-basis` можно трактовать как минимальную ширину flex-элемента. Если вы укажете значение `flex-basis`, оно установит ширину конкретного элемента, но, в зависимости от других flex-параметров, flex-элемент может стать шире (или уже), чем значение `flex-basis`.

Примеры:

```
.wrapper {  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  flex-flow: row wrap;  
}  
  
.item {  
  flex: 1 1 250px;  
}
```



Верхние элементы выстроились три в ряд из-за минимальной ширины flex-basis : 250px, а на второй строке оказались те элементы, которые не поместились. Ширина контейнера – 940px, три элемента занимают минимум – 750px, а четыре уже – 1000px, поэтому и произошел перенос на следующую строчку, начиная с элемента четыре.

Если третьему блоку дать ширину flex-basis: 500px, то ситуация станет следующей:



Третий блок из-за `flex: 1 1 500px;` не вмещается и переносится на вторую строку. А элемент пять, на третьей строке, из-за недостаточного размера контейнера по высоте, даже вылез из него.

Учимся играя

Чтобы хорошо усвоить все свойства flexbox, рекомендуем вам пройти две игры.

Первая это [FLEXBOX FROGGY](#). В ней вам предстоит помочь лягушатам преодолеть все препятствия.

The screenshot shows the 'FLEXBOX FROGGY' game interface. At the top, it says 'Уровень 7 из 24'. Below that is a text box containing CSS code:

```
1 #pond {  
2   display: flex;  
3   justify-content: space-around;  
4   align-items: flex-end  
5 }  
6  
7  
8  
9  
10
```

At the bottom of the text box is a red button labeled 'Следующий' (Next). Below the text box, there's a note: 'Лягушатам снова нужно пересечь пруд. В этот раз к лилиям, с достаточно большим пространством вокруг них. Используй комбинацию `justify-content` и `align-items`'. At the very bottom, it says 'Игра Flexbox Froggy создана Thomas Park - GitHub - Twitter' and lists various language links.

On the right side of the screenshot, there are three circular icons representing frogs in different colors: green, yellow, and red.

Вторая [Flexbox Defense](#) - это классическая стратегия, игра по типу [Tower Defense](#).



Желаем вам с пользой и приятно провести время :)

Очень полезные ссылки

1. <http://flexbox.help/> – очень хороший сервис, помогающий четко уяснить работу свойств flexbox
 2. [Схема работы flexbox](#) – еще один красиво оформленный пример-песочница, чтобы разобраться в работе flexbox
 3. [A Complete Guide to Flexbox](#) – гайд по основным свойствам flexbox на английском языке
 4. [Шпаргалка по шпаргалке по Flexbox](#) – краткий визуальный гайд в картинках по поведению flexbox
 5. [Что такое Flexbox?](#) – описание всех css свойств, основные принципы, преимущества и недостатки.
 6. [Практическое применение FlexBox](#) – верстка лейаута на FlexBox
 7. [Flexbox Patterns](#) – основные паттерны при верстке на flexbox. Очень наглядно
 8. [FlexBox Cheatsheet](#) – шпаргалка
 9. [Визуальное руководство по свойствам Flexbox из CSS3](#) – перевод статьи «A Visual Guide to CSS3 Flexbox Properties» с сайта scotch.io
 10. [Flexbox: Practical Examples](#) – красивые практические примеры на flexbox
 11. [Полное руководство по Flexbox](#)
 12. [Как работают в флексбоксе z-index и автоотступы](#)
 13. [Flexbox для интерфейсов во всей красе](#) – рассмотрены шаблоны раскладки, которые подходят для flexbox, используя интерфейс приложения Tracks, в котором также используются принципы атомного дизайна
 14. [Почти полное руководство по flexbox \(без самих flexbox\)](#)
- К сожалению, не у всех есть браузер/устройство, поддерживающие

flexbox. Это руководство в виде шпаргалки, в котором предлагаются альтернативы обратной совместимости для свойств flexbox.