Тема 7. FlexBox. Сетка Flexbox

Модуль *Flexbox-layouta* (flexible box — «гибкий блок», на данный момент W3C Candidate Recommendation) ставит задачу предложить более эффективный способ вёрстки, выравнивания и распределения свободного места между элементами в контейнере, даже когда их размер неизвестен и/или динамический (отсюда слово «гибкий»).

Главная идея flex-вёрстки в наделении контейнера способностью изменять ширину/высоту (и порядок) своих элементов для наилучшего заполнения пространства (в большинстве случаев — для поддержки всех видов дисплеев и размеров экранов). Flex-контейнер растягивает элементы для заполнения свободного места или сжимает их, чтобы предотвратить выход за границы.

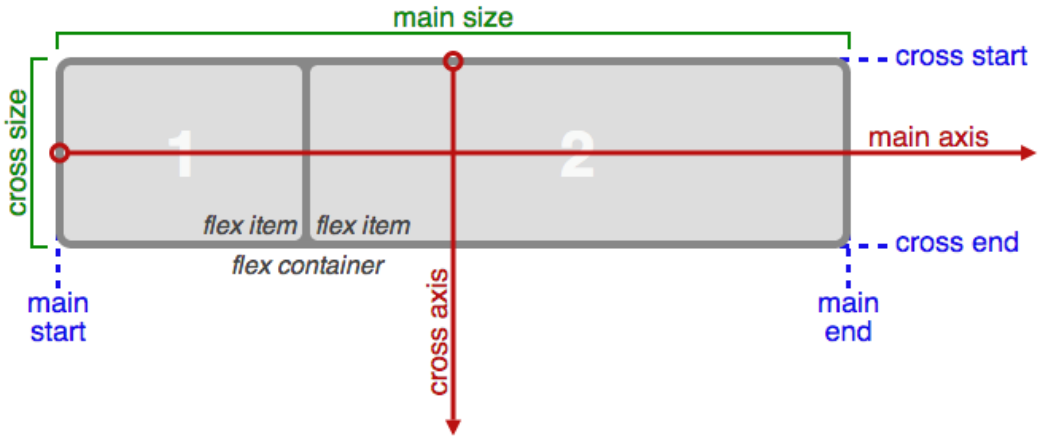
Самое важное, flexbox-layout не зависит от направления в отличие от обычных layoutов (блоки, располагающиеся вертикально, и инлайн-элементы, располагающиеся горизонтально). В то время, как обычный layout отлично подходит для веб-страниц, ему не хватает гибкости (никакого каламбура) для поддержки больших или сложных приложений (особенно когда дело доходит до смены ориентации экрана, изменения размера, растягивания, сжатия и т.п.).

Замечание: Flexbox-layout лучше всего подходит для составных частей приложения и мелкомасштабных layoutов, в то время как Grid-layout больше используется для layoutов большого масштаба.

Основы

Т.к. flexbox — это целый модуль, а не просто единичное свойство, он объединяет в себе множество свойств. Некоторые из них должны применяться к контейнеру (родительскому элементу, так называемому *flex-контейнеру*), в то время как другие свойства применяются к дочерним элементам, или *flex-элементам*.

Если обычный layout основывается на направлениях потоков блочных и инлайн-элементов, то flex-layout основывается на «направлениях flex-потока». Ознакомьтесь с этой схемой из спецификации, разъясняющей основную идею flex-layoutов.



В основном элементы будут распределяться либо вдоль *главной оси* (от *main-start* до *main-end*), либо вдоль *поперечной оси* (от *cross-start* до *cross-end*).

- **main-axis** - главная ось, вдоль которой располагаются flex-элементы. Обратите внимание, она необязательно должна быть горизонтальной, всё зависит от свойства flex-direction (см. ниже).
- **main-start** | **main-end** - flex-элементы размещаются в контейнере от позиции main-start до позиции main-end.
- **main size** - ширина или высота flex-элемента в зависимости от выбранной основной величины. Основная величина может быть либо шириной, либо высотой элемента.
- **cross axis** - поперечная ось, перпендикулярная к главной. Её направление зависит от направления главной оси.
- **cross-start** | **cross-end** - flex-строки заполняются элементами и размещаются в контейнере от позиции cross-start и до позиции cross-end.
- **cross size** - ширина или высота flex-элемента в зависимости от выбранной размерности равняется этой величине. Это свойство совпадает с width или height элемента в зависимости от выбранной размерности.

Свойства

display: flex | inline-flex;

Применяется к: родительскому элементу flex-контейнера.

Определяет flex-контейнер (инлайновый или блочный в зависимости от выбранного значения), подключает flex-контекст для всех его непосредственных потомков.

```
display: other values | flex | inline-flex;
```

Имейте в виду:

- CSS-столбцы columns не работают с flex-контейнером
- float, clear и vertical-align не работают с flex-элементами

flex-direction

Применяется к: родительскому элементу flex-контейнера.

Устанавливает главную ось main-axis, определяя тем самым направление для flex-элементов, размещаемых в контейнере.

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

- row (по умолчанию): слева направо для ltr, справа налево для rtl;
- row-reverse: справа налево для ltr, слева направо для rtl;

- column: аналогично row, сверху вниз;
- column-reverse: аналогично row-reverse, снизу вверх.

flex-wrap

Применяется к: родительскому элементу flex-контейнера.

Определяет, будет ли контейнер однострочным или многострочным, а также направление поперечной оси, определяющей направление, в котором будут располагаться новые строки.

flex-wrap: nowrap | wrap | wrap-reverse

- nowrap (по умолчанию): однострочный / слева направо для ltr, справа налево для rtl;
- wrap: многострочный / слева направо для ltr, справа налево для rtl;
- wrap-reverse: многострочный / справа налево для ltr, слева направо для rtl.

flex-flow

Применяется к: родительскому элементу flex-контейнера.

Это сокращение для свойств flex-direction и flex-wrap, вместе определяющих главную и поперечную оси. По умолчанию принимает значение row nowrap.

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

justify-content

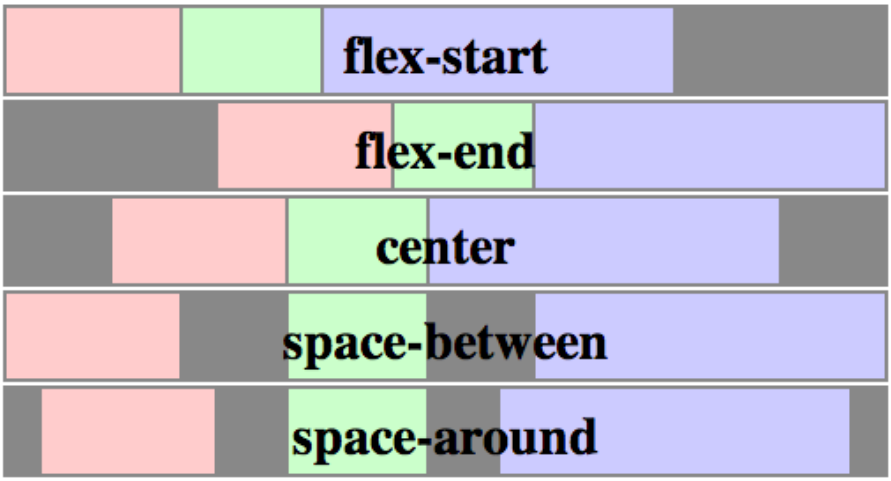
Применяется к: родительскому элементу flex-контейнера.

Определяет выравнивание относительно главной оси. Помогает распределить оставшееся свободное место в случае, когда элементы строки не «тянутся», либо тянутся, но уже достигли своего максимального размера. Также позволяет в некотором роде управлять выравниванием элементов при выходе за границы строки.

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

- flex-start (по умолчанию): элементы сдвигаются к началу строки;
- flex-end: элементы сдвигаются к концу строки;
- center: элементы выравниваются по центру строки;

- space-between: элементы распределяются равномерно (первый элемент в начале строки, последний — в конце);
- space-around: элементы распределяются равномерно с равным расстоянием между собой и границами строки.



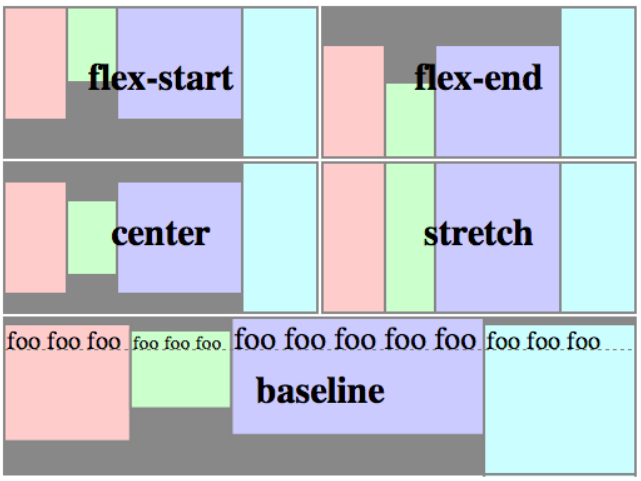
align-items

Применяется к: родительскому элементу flex-контейнера.

Определяет поведение по умолчанию для того, как flex-элементы располагаются относительно поперечной оси на текущей строке. Считайте это версией justify-content для поперечной оси (перпендикулярной к основной).

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

- flex-start: граница cross-start для элементов располагается на позиции cross-start;
- flex-end: граница cross-end для элементов располагается на позиции cross-end;
- center: элементы выравниваются по центру поперечной оси;
- baseline: элементы выравниваются по своей базовой линии;
- stretch (по умолчанию): элементы растягиваются, заполняя контейнер (с учётом min-width/max-width).



align-content

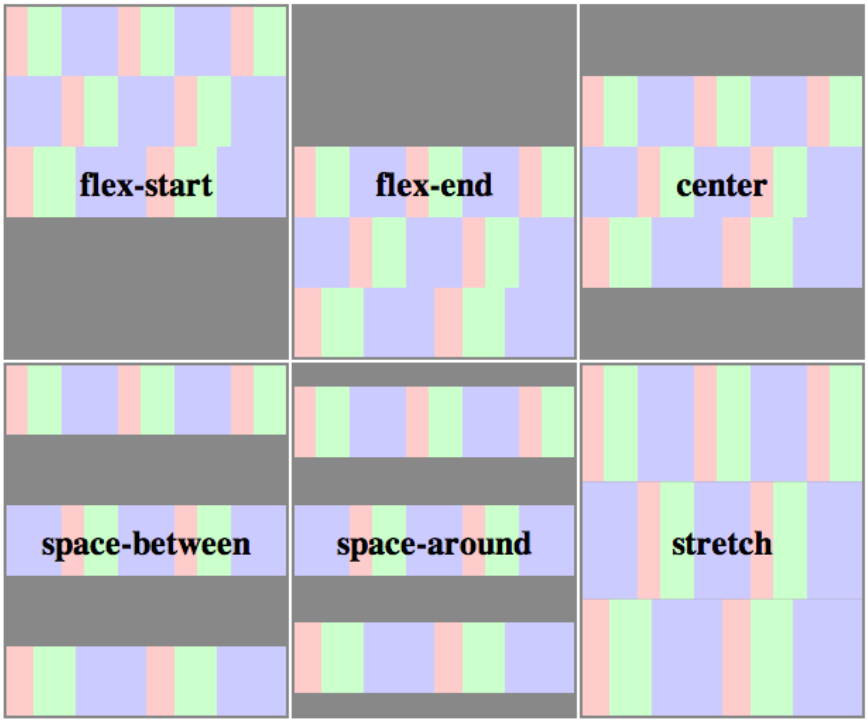
Применяется к: родительскому элементу flex-контейнера.

Выравнивает строки flex-контейнера при наличии свободного места на поперечной оси аналогично тому, как это делает justify-content на главной оси.

Замечание: это свойство не работает с однострочным flexbox.

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

- flex-start: строки выравниваются относительно начала контейнера;
- flex-end: строки выравниваются относительно конца контейнера;
- center: строки выравниваются по центру контейнера;
- space-between: строки распределяются равномерно (первая строка в начале строки, последняя — в конце);
- space-around: строки распределяются равномерно с равным расстоянием между собой;
- stretch (по умолчанию): строки растягиваются, заполняя свободное пространство.



order

Применяется к: дочернему элементу / flex-элементу.

По умолчанию flex-элементы располагаются в исходном порядке. Тем не менее, свойство order может управлять порядком их расположения в контейнере.

```
order: <integer>
```

flex-grow

Применяется к: дочернему элементу / flex-элементу.

Определяет для flex-элемента возможность «вырастать» при необходимости. Принимает безразмерное значение, служащее в качестве пропорции. Оно определяет, какую долю свободного места внутри контейнера элемент может занять.

Если у всех элементов свойство flex-grow задано как 1, то каждый потомок получит внутри контейнера одинаковый размер. Если вы задали одному из потомков значение 2, то он заберёт в два раза больше места, чем другие.

```
flex-grow: <number> (по умолчанию 0)
```

Отрицательные числа не принимаются.

flex-shrink

Применяется к: дочернему элементу / flex-элементу.

Определяет для flex-элемента возможность сжиматься при необходимости.

```
flex-shrink: <number> (default 1)
```

Отрицательные числа не принимаются.

flex-basis

Применяется к: дочернему элементу / flex-элементу.

Определяет размер по умолчанию для элемента перед распределением пространства в контейнере.

```
flex-basis: <length> | auto (default auto)
```

flex

Применяется к: дочернему элементу / flex-элементу.
Это сокращение для flex-grow, flex-shrink и flex-basis. Второй и третий параметры (flex-shrink, flex-basis) необязательны. Значение по умолчанию — 0 1 auto.

```
flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
```

align-self

Применяется к: дочернему элементу / flex-элементу.

Позволяет переопределить выравнивание, заданное по умолчанию или в align-items, для отдельных flex-элементов.
Обратитесь к описанию свойства align-items для лучшего понимания доступных значений.

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

Примеры

Начнём с очень-очень простого примера, встречающегося практически каждый день: выравнивание точно по центру. Нет ничего проще, если использовать flexbox.

```
.parent {
  display: flex;
  height: 300px; /* Или что угодно */
```

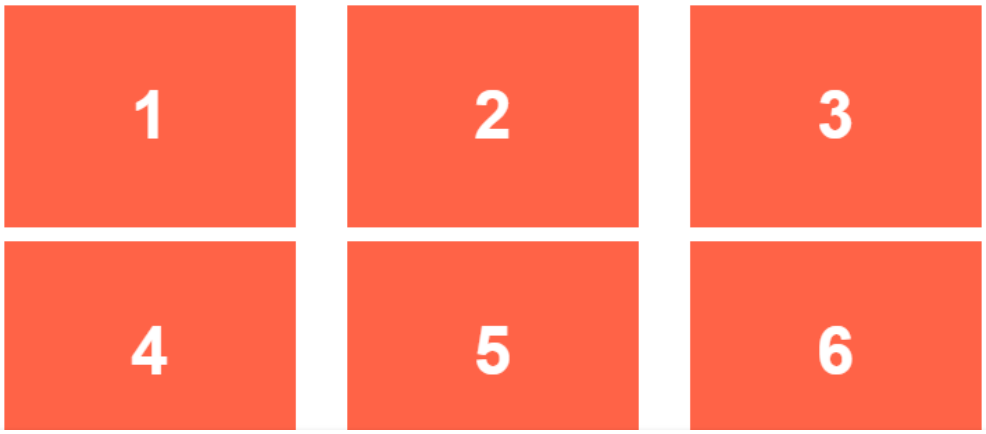
```
}

.child {
  width: 100px;  /* Или что угодно */
  height: 100px; /* Или что угодно */
  margin: auto;  /* Магия! */
}
```

Этот пример основывается на том, что margin во flex-контейнере, заданный как auto, поглощает лишнее пространство, поэтому задание отступа таким образом выровняет элемент ровно по центру по обеим осям.

Теперь давайте используем какие-нибудь свойства. Представьте набор из 6 элементов фиксированного размера (для красоты), но с возможностью изменения размера контейнера. Мы хотим равномерно распределить их по горизонтали, чтобы при изменении размера окна браузера всё выглядело хорошо (без @media-запросов!).

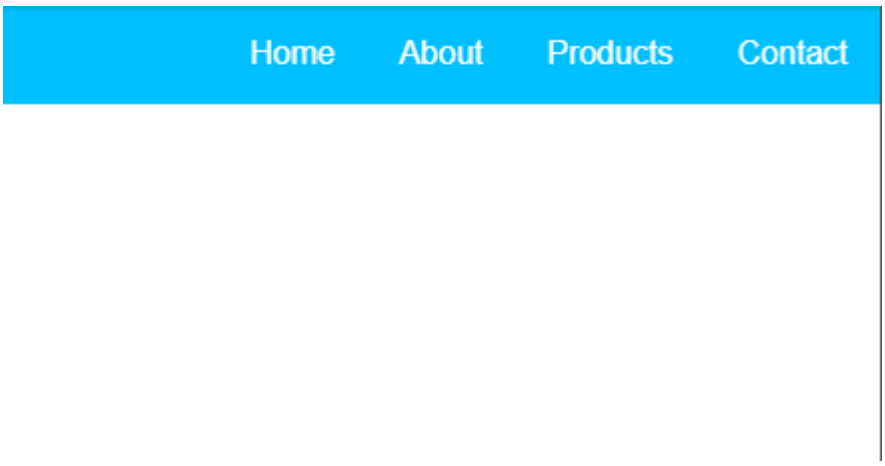
```
.flex-container {
  /* Сначала создадим flex-контекст */
  display: flex;

  /* Теперь определим направление потока и хотим ли мы, чтобы элементы
  переносились на новую строку
   * Помните, что это тоже самое, что и:
   * flex-direction: row;
   * flex-wrap: wrap;
   */
  flex-flow: row wrap;

  /* Теперь определим, как будет распределяться пространство */
  justify-content: space-around;
}
```

Готово. Всё остальное — уже дело оформления.



Давайте попробуем что-нибудь ещё. Представьте, что нам нужна выровненная по правому краю навигация в самом верху нашего сайта, но мы хотим, чтобы она выравнивалась по центру для экранов среднего размера и превращалась в один столбец на маленьких. Всё достаточно просто.

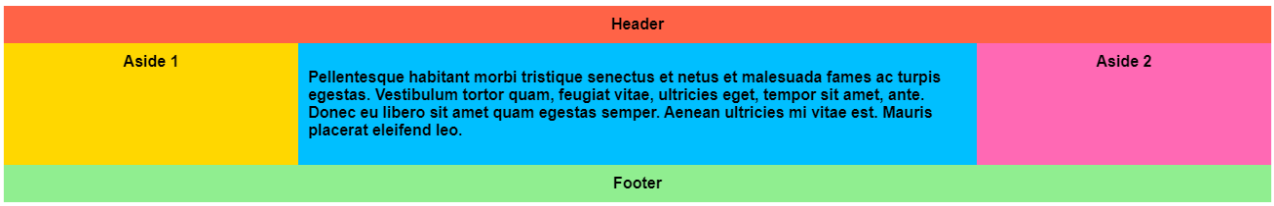
```
/* Большие экраны */
.navigation {
  display: flex;
  flex-flow: row wrap;
  /* Сдвигает элементы к концу строки по главной оси */
  justify-content: flex-end;
}

/* Экраны среднего размера */
@media all and (max-width: 800px) {
  .navigation {
    /* Для экранов среднего размера мы выравниваем навигацию по центру,
    равномерно распределяя свободное место между элементами */
    justify-content: space-around;
  }
}

/* Маленькие экраны */
@media all and (max-width: 500px) {
  .navigation {
    /* На маленьких экранах вместо строки мы располагаем элементы в
столбце */
    flex-direction: column;
  }
}
```



Давайте попробуем кое-что получше и поиграем с гибкостью flex-элементов! Как насчёт ориентированного на мобильные устройства трёхколоночного макета с полноширинной шапкой и подвалом? И другим порядком расположения.

```
.wrapper {
  display: flex;
  flex-flow: row wrap;
}

/* Задаём всем элементам ширину в 100% */
.header, .main, .nav, .aside, .footer {
  flex: 1 100%;
}

/* В этом случае мы полагаемся на исходный порядок для ориентации на
 * мобильные устройства:
 * 1. header
 * 2. nav
 * 3. main
 * 4. aside
 * 5. footer
 */

/* Экраны среднего размера */
@media all and (min-width: 600px) {
  /* Оба сайдбара располагаются в одной строке */
  .aside { flex: 1 auto; }
}

/* Большие экраны */
@media all and (min-width: 800px) {
  /* Мы меняем местами элементы .aside-1 и .main, а также сообщаем
```

```
    * элементу .main забирать в два раза больше места, чем сайдбары.
    */
  .main { flex: 2 0px; }

  .aside-1 { order: 1; }
  .main    { order: 2; }
  .aside-2 { order: 3; }
  .footer  { order: 4; }
}
```



Поддержка браузерами

- (modern) означает поддержку нового синтаксиса из спецификации (display: flex;)
- (hybrid) означает поддержку старого неофициального синтаксиса из 2011 (display: flexbox;)
- (old) означает поддержку старого синтаксиса из 2009 (display: box;)

Chrome	Safari	Firefox	Opera	IE	Android	iOS
21+ (modern) 20- (old)	3.1+ (old)	2-21 (old) 22+ (new)	12.1+ (modern)	10+ (hybrid)	2.1+ (old)	3.2+ (old)

Браузер Blackberry версии 10+ поддерживает новый синтаксис.

SASS-@mixin для облегчения боли:

```
@mixin flexbox() {
  display: -webkit-box;
  display: -moz-box;
  display: -ms-flexbox;
  display: -webkit-flex;
  display: flex;
}

@mixin flex($values) {
  -webkit-box-flex: $values;
  -moz-box-flex:  $values;
  -webkit-flex:  $values;
```

```
  -ms-flex:  $values;
  flex:  $values;
}

@mixin order($val) {
  -webkit-box-ordinal-group: $val;
  -moz-box-ordinal-group: $val;
  -ms-flex-order: $val;
  -webkit-order: $val;
  order: $val;
}

.wrapper {
  @include flexbox();
}

.item {
  @include flex(1 200px);
  @include order(2);
}
```