Конспект «Микросетки. Продолжение». Раздел 1

Поперечная ось и свойство align-items (флекс)

Во флексах свойство `align-items` управляет расположением элементов на поперечной оси.

Поперечная ось идёт перпендикулярно главной оси и по умолчанию направлена сверху вниз:

Во флекс-контейнере свойство `align-items` может иметь следующие значения:

- `stretch` — значение по умолчанию; элементы растягиваются на всю высоту поперечной оси.
- `flex-start` — элементы сжимаются до содержимого и располагаются в начале поперечной оси (по умолчанию сверху);
- `flex-end` — элементы сжимаются до содержимого и располагаются в конце поперечной оси (по умолчанию снизу);
- `center` — элементы сжимаются до содержимого и располагаются по центру поперечной оси;

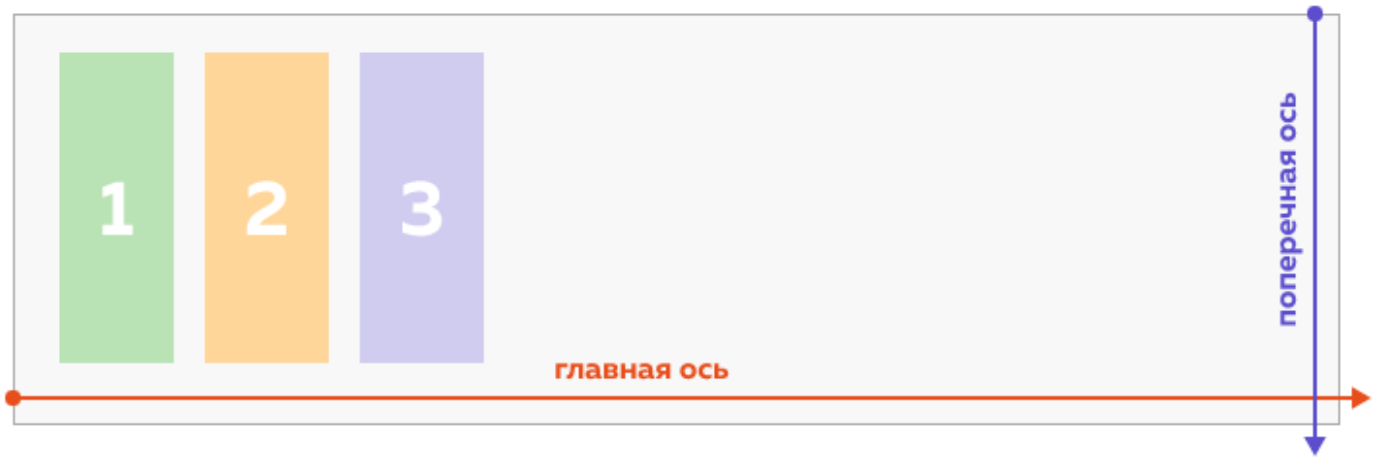
Поворот главной оси, свойство flex-direction

Главную ось поворачивают, чтобы сохранить внутри флекс-контейнера направление потока сверху вниз. За направление главной оси отвечает свойство `flex-direction`. По умолчанию у него значение `row` (ряд), но его можно изменить на `column` (колонка):

```
flex-container {

  display: flex;

  flex-direction: column;

}
```

В этом случае главная ось будет направлена сверху вниз, а поперечная — слева направо. В результате флекс-элементы выстроятся сверху вниз.

По умолчанию флекс-элементы сжимаются по главной оси и растягиваются по поперечной. Таким образом, если главная ось направлена слева направо, то элементы сжимаются по горизонтали и растягиваются по вертикали.



Если же главная ось направлена сверху вниз, то сжатие происходит по вертикали, а растяжение — по горизонтали.

Получается, чтобы при повёрнутых осях выровнять элемент по горизонтали, нужно задать ему выравнивание по поперечной оси.

Свойство align-self (флекс)

Свойство `align-self` задаётся флекс-элементу и говорит, как ему расположиться на поперечной оси. Значения у этого свойства такие же, как у `align-items`: `stretch` (значение по умолчанию), `flex-start`, `flex-end` и `center`.

```
.element {

  align-self: flex-end;

}
```

Свойство order

Чтобы изменить визуальный порядок элементов, удобно использовать свойство `order`. В качестве значения свойство принимает число, причём оно может быть как положительным, так и отрицательным. По умолчанию у всех элементов свойство `order` равно нулю.

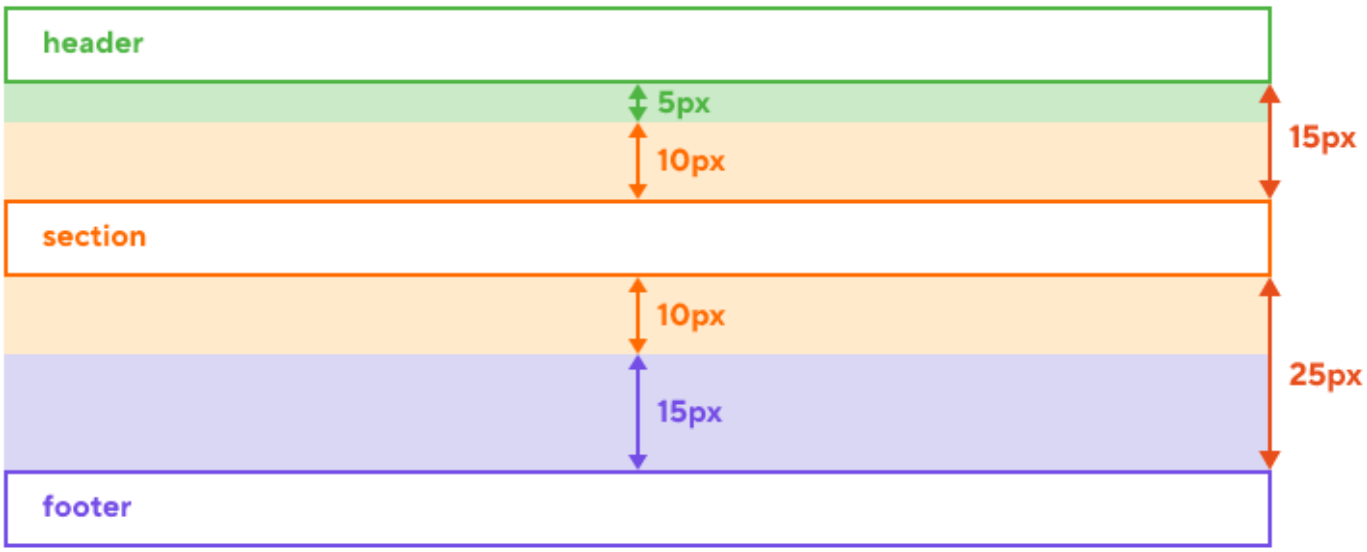
```
.element {

  order: 5;

}
```

Элементы выстраиваются от меньшего значения `order` к большему. Если у нескольких элементов одинаковое значение, используется их порядок в разметке.

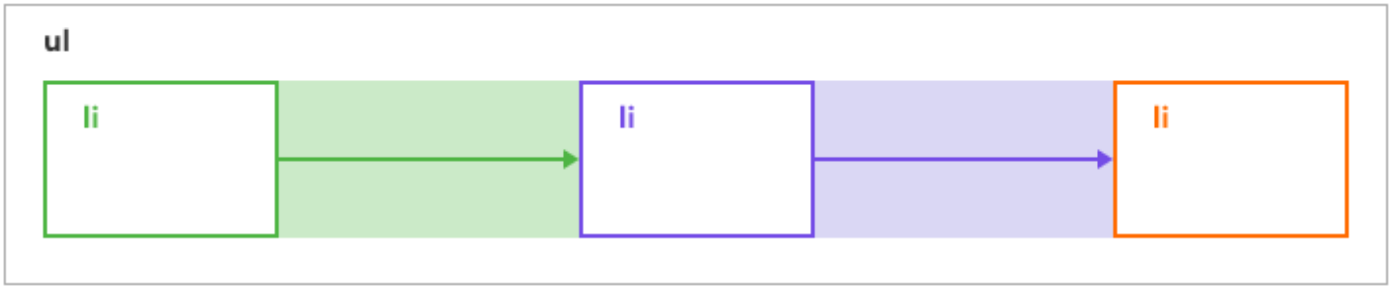
Свойство `order` работает только в грид- и флекс-контейнерах.

Отступы у флекс-элементов

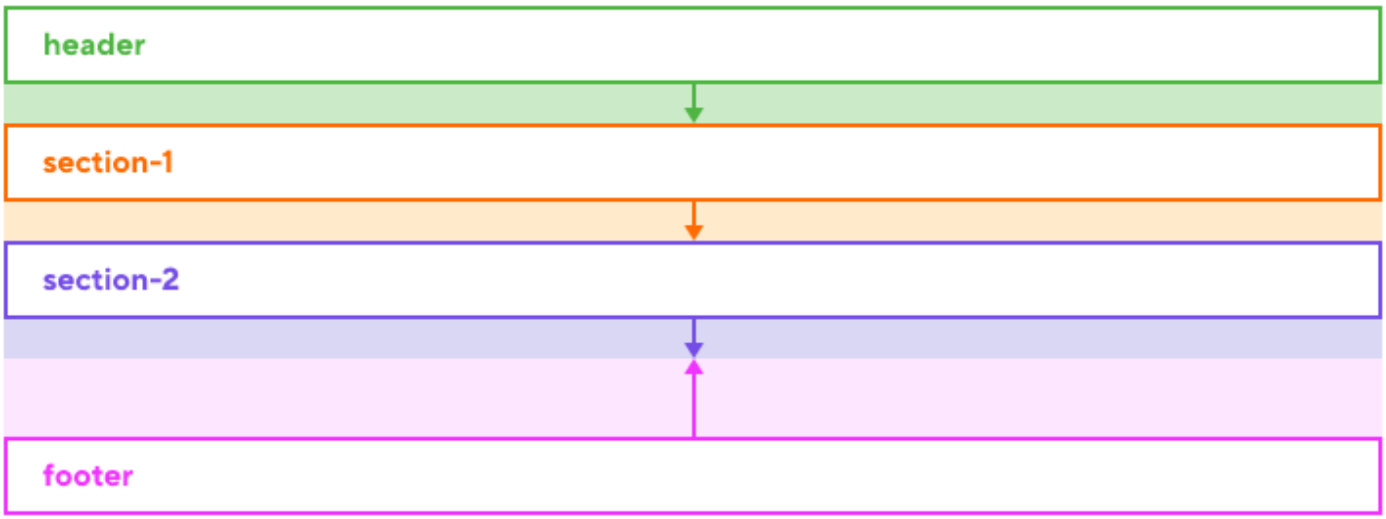
Внешние отступы у соседних флекс-элементов складываются.



Чтобы не запутаться и получить именно те размеры, которые указаны в макете, верстальщики добавляют элементам внешние отступы только с одной стороны. Часто внешние отступы задают в направлении потока. Если элементы выстроены горизонтально, то отступ задают справа, а у последнего элемента обнуляют:



Если элементы выстроены вертикально, то отступ добавляют снизу. Исключение — самый последний элемент (например, подвал страницы), ему при необходимости задают отступ сверху:



В этом случае, даже если изменить порядок секций, они не слипнутся, а между ними не появятся лишние отступы.

Уменьшение изображений с сохранением пропорций

Если использовать картинку большего размера, чем родительский элемент, она выпадет из контейнера. Чтобы не допускать подобного, верстальщики добавляют картинкам такие стили:

```
img {

  max-width: 100%;

  height: auto;

}
```

Свойство `max-width` задаёт максимальную ширину, а значение `100%` говорит, что элемент не должен становиться больше ширины родителя.

Свойство `height` задаёт высоту элемента. Значение `auto` используют, чтобы изображение не деформировалось и сохраняло свои пропорции. Если его не указать, то будет использовано значение атрибута `height` из разметки, и картинка, скорее всего, исказится.

Этот способ работает, потому что у CSS-свойств `max-width` и `height` приоритет выше, чем у атрибутов `width` и `height` в разметке.

Конспект «Микросетки. Продолжение». Раздел 2

Придумать подходящую структуру грид-контейнера бывает непросто. Если элементы расположены асимметрично, часто требуется больше колонок, чем кажется на первый взгляд.

Грид-области

Грид-областью называют часть сетки грид-контейнера, у которой есть имя. Имя области придумывает сам разработчик. Оно должно начинаться с буквы и может включать цифры, дефис и знак подчёркивания. Например: `header`, `section-2`, `user_avatar`. Следует выбирать такие имена, которые описывают содержимое области.

Чтобы описать структуру грида с помощью областей, используют свойство `grid-template-areas`. В нём указывают имена грид-областей. При этом каждый ряд оборачивают в кавычки, а колонки разделяют пробелом.

```
.card {

  display: grid;

  grid-template-areas: "title price";

}
```

В `grid-template-areas` можно указать сколько угодно рядов. При этом в кавычки оборачивают **каждый** ряд, но точка с запятой ставится только после последнего ряда! Для большей наглядности ряды записывают друг под другом:

```
grid-template-areas: "title price"

                     "options description";
```

Однако просто описать шаблон недостаточно, ведь браузер не знает, какие элементы мы имеем в виду. Чтобы связать имя области с соответствующим грид-элементом, используют свойство `grid-area`. Обратите внимание, в `grid-area` кавычки не нужны!

```
.title {

  grid-area: title;

}
```

Как растянуть грид-область

Если требуется растянуть область на несколько колонок, её имя повторяют нужное число раз:

```
grid-template-areas: "title title price";
```

Чтобы растянуть область на несколько рядов, достаточно повторить её имя в разных рядах:

```
grid-template-areas: "options description"
                     "options disclaimer";
```

Ограничения при использовании грид-областей

Грид-области должны быть прямоугольными и непрерывными.

```
// Такой код не сработает
grid-template-areas: "logo main"
                     "main main";


// Такой – тоже
grid-template-areas: "nav logo nav";
```

При описании шаблона количество колонок в каждом ряду должно быть одинаковым:

```
// Такой код не сработает
grid-template-areas: "logo nav"
                     "aside main banner"
                     "footer";
```

Если задать неправильное значение `grid-template-areas`, браузер его проигнорирует, и вёрстка может сломаться.

Имена областей *в одном контейнере* должны быть уникальными. Если задать одинаковое имя нескольким элементам, они наложатся друг на друга. По этой причине грид-области не подойдут, например, для списка карточек, особенно если количество этих карточек неизвестно заранее.

Свойство align-self (грид)

Свойство `align-self` задаёт выравнивание по вертикали *одному элементу*. Значения `align-self` принимает те же, что и `align-items`: `stretch` (значение по умолчанию), `start`, `end` и `center`.

```
.element {

  align-self: start;

}
```

Свойство justify-self

Свойство `justify-self` отвечает за выравнивание отдельного элемента по горизонтали:

```
.element {

  justify-self: start;

}
```

Это свойство принимает следующие значения:

- `stretch` — значение по умолчанию; элемент занимает всё пространство по ширине.
- `start` — элемент сжимается до содержимого и прижимается к левой границе.
- `end` — элемент сжимается до содержимого и прижимается к правой границе.
- `center` — элемент сжимается до содержимого и располагается по центру.