

FLEXBOX

НАШИ ПРАВИЛА



Включенная камера



Вопросы по поднятой руке



Не перебиваем друг друга



Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору



Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

ЦЕЛЬ

Изучить концепцию Flexbox. Узнать какие свойства отвечают за выравнивание элементов.

ПЛАН ЗАНЯТИЯ

- 1. Определение Flexbox
- 2. Главная и поперечная оси
- 3. Свойства контейнера
- 4. Свойства элемента

ПОИГРАЕМ ;)

■ Какие значения есть у свойства position

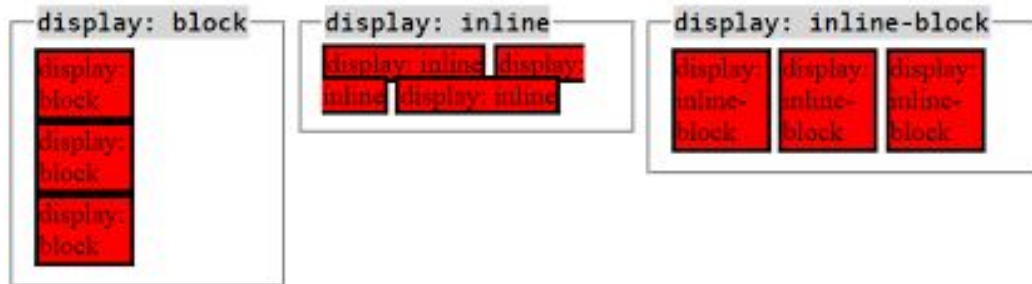
■ Какое значение свойства position отвечает за его фиксацию относительно экрана

■ Какое свойства помогает нам указать порядок позиционированных элементов по слоям

СВОЙСТВО **display**

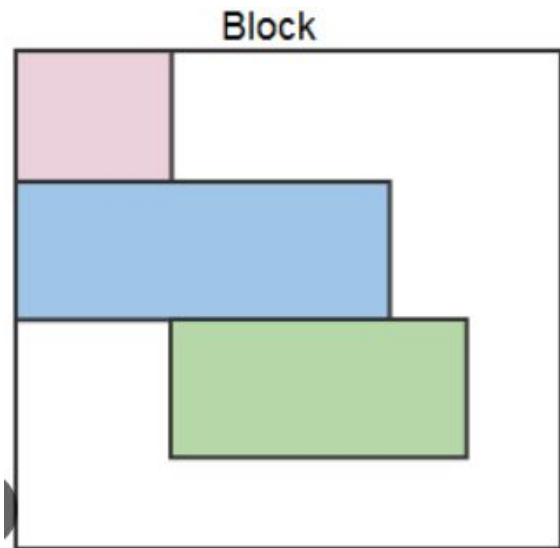
display

Свойство `display` в CSS определяет, как элемент будет отображаться на веб-странице, и контролирует его тип отображения. Это одно из наиболее важных свойств CSS, потому что оно позволяет определить, будет ли элемент блочным, строчным, таблицей, или другим типом элемента.



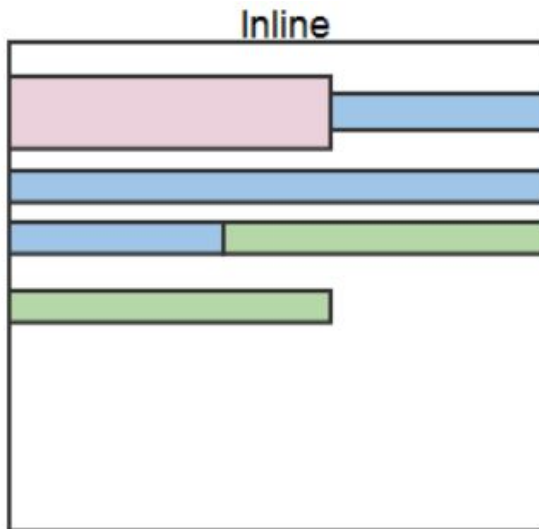
display

block: Элемент становится блочным элементом, занимающим всю доступную горизонтальную ширину и начиная новую строку после себя. Примеры блочных элементов включают `div`, `p`, `h1` и другие.



display

inline: Элемент становится строчным элементом и занимает только столько горизонтального пространства, сколько необходимо для его содержимого. Строчные элементы могут находиться на одной строке с другими строчными элементами. Примеры строчных элементов включают `span`, `a`, `strong` и другие.



display

inline-block: Элемент комбинирует свойства блочного и строчного элемента. Он занимает только столько горизонтального пространства, сколько необходимо для содержимого, но также позволяет устанавливать высоту и ширину элемента, как для блочных элементов.

Этот блочный элемент занимает всю ширину.

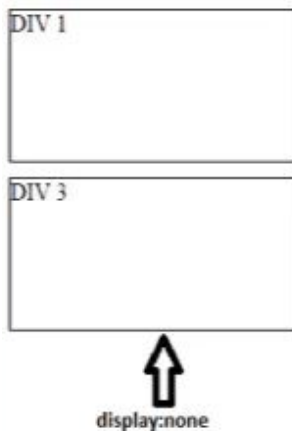
Этот строчный элемент занимает только столько места, сколько ему нужно.

Этот элемент имеет значение `inline-block`, поэтому его содержимое

может быть отформатировано как блок и иметь ширину и высоту.

display

none: Элемент полностью скрывается и не занимает место в макете документа. Это используется для скрывания элементов.



display

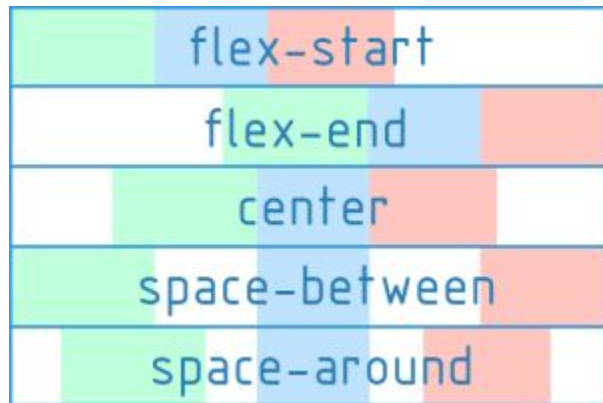
table, table-row, table-cell, и другие: Эти значения применяются к элементам для создания макета таблицы, аналогичного HTML-таблицам. Они используются, когда требуется более сложное расположение элементов, подобное таблице.

```
table    { display: table }
tr       { display: table-row }
thead    { display: table-header-group }
tbody    { display: table-row-group }
tfoot    { display: table-footer-group }
col       { display: table-column }
colgroup { display: table-column-group }
td, th   { display: table-cell }
caption  { display: table-caption }
```

<https://developer.mozilla.org/en-US/docs/Web/CSS/display>

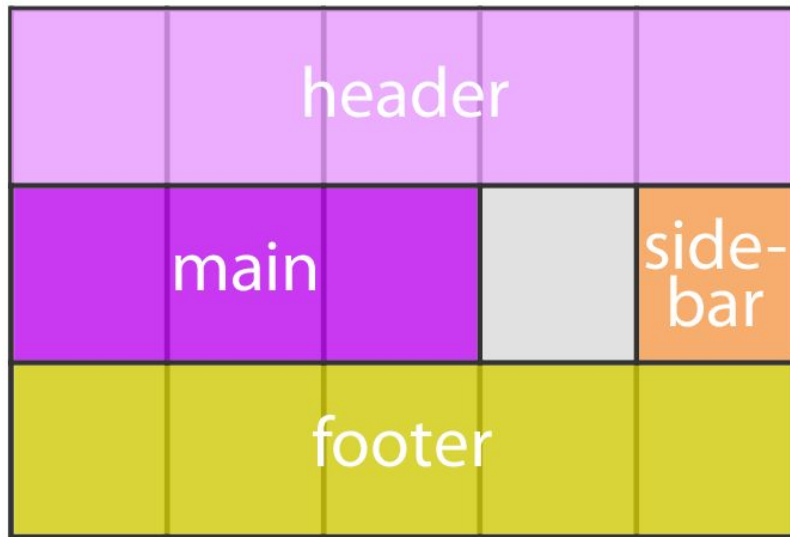
display

flex: Элемент становится контейнером для гибких (flex) элементов. Это позволяет управлять распределением пространства в контейнере и создавать адаптивные макеты.



display

grid: Элемент становится контейнером для элементов в сетке (grid). Это позволяет создавать сложные многоколоночные макеты.



Flexbox

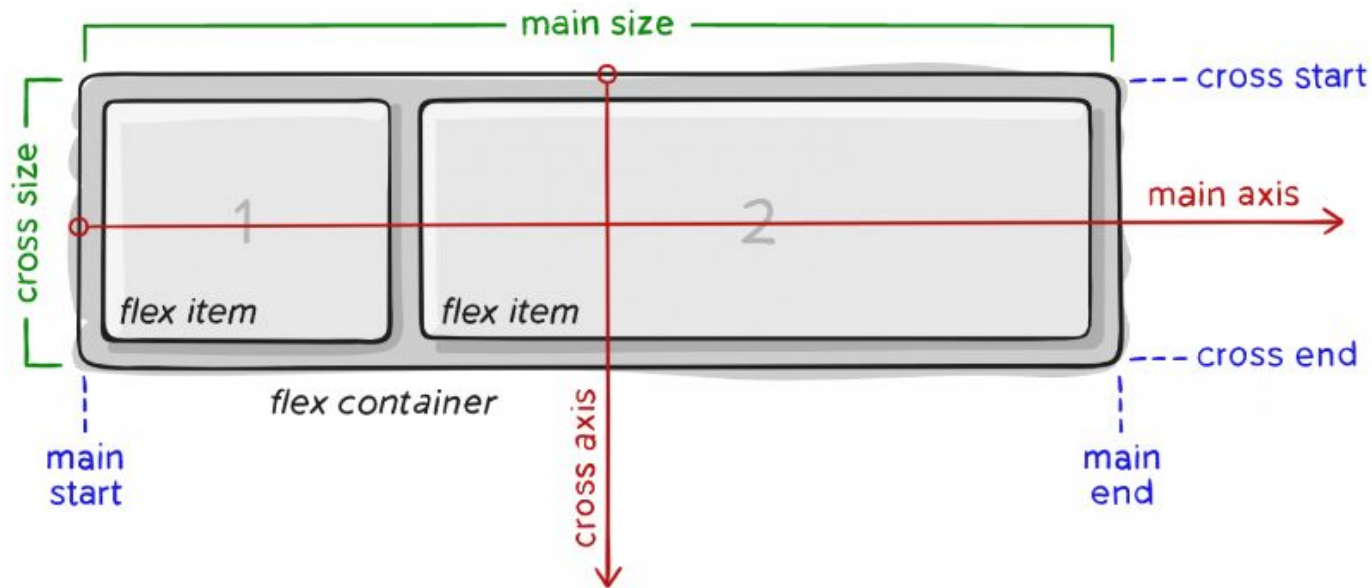


Flexbox

Flexbox - предоставляет инструменты для быстрого создания сложных, гибких макетов



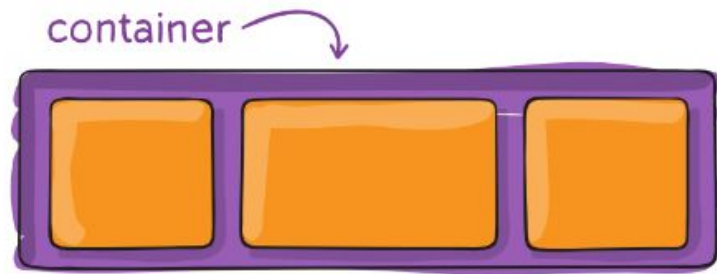
Если «обычная» компоновка основана как на блочном, так и на inline направлениях, то flex layout основана на «направлениях flex-flow»



Элементы будут расположены либо в направлении главной оси (main axis от main-start до main-end) или в направлении поперечной оси (cross axis от cross-start до cross-end).

- **main axis** — главная ось flex контейнера — это основная ось, вдоль которой располагаются flex элементы. Будьте внимательны, эта ось не обязательно горизонтальная; это зависит от flex-direction свойства (см. ниже).
- **main-start | main-end** — flex элементы помещаются в контейнер, начиная с main-start и заканчивая main-end.
- **main size** — ширина или высота flex элемента, в зависимости от того, что находится в основном измерении. Определяется основным размером flex элементов т.е. свойством 'width' или 'height', в зависимости от того, что находится в основном измерении.
- **cross axis** — ось перпендикулярная главной оси, называется поперечной осью. Её направление зависит от направления главной оси.
- **cross-start | cross-end** — flex строки заполняются элементами и помещаются в контейнер, начиная от cross-start flex контейнера по направлению к cross-end.
- **cross size** — ширина или высота flex элемента. В зависимости от css свойства flex-direction, это ширина или высота элемента. Это всегда поперечный размер flex элементов.

Свойства для Родителя (flex контейнер)



display

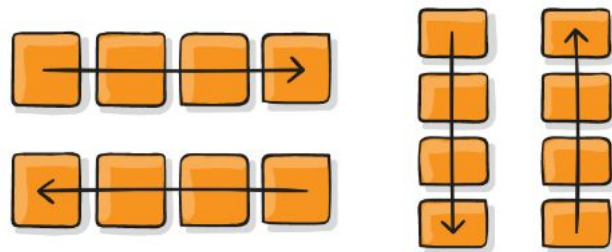
Определяет flex контейнер. Включает flex контекст для всех потомков первого уровня.

```
.container {  
  display: flex;  
}
```

Свойства для Родителя (flex контейнер)

flex-direction

Устанавливает основную ось, таким образом определяя направление flex элементов, помещаемых в flex контейнер.



- **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

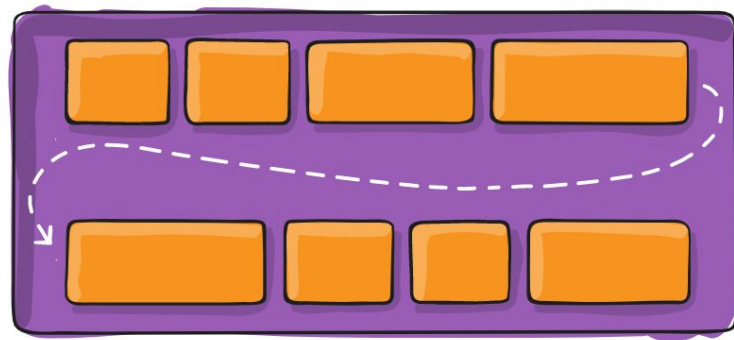
```
.container {  
  flex-direction: row | row-reverse | column | column-reverse;  
}
```

Свойства для Родителя (flex контейнер)

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.

- **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.



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

Свойства для Родителя (flex контейнер)

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'>
```

Свойства для Родителя

justify-content

Это свойство определяет выравнивание вдоль главной оси.

- **flex-start** (по умолчанию): элементы сдвинуты в начало flex-direction.
- **flex-end**: элементы сдвинуты ближе к концу flex направления.
- **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.
- **space-evenly**: элементы распределяются таким образом, чтобы расстояние между любыми двумя элементами (и расстояние до краев) было одинаковым.

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

flex-start



flex-end



center



space-between



space-around



space-evenly

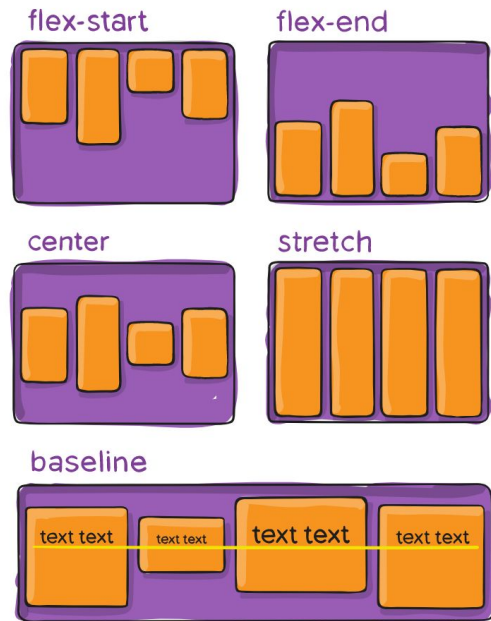


Свойства для Родителя

align-items

Это свойство определяет поведение по умолчанию того, как flex элементы располагаются вдоль поперечной оси на текущей линии.

- **stretch** (по умолчанию): растягивать, чтобы заполнить контейнер (все еще соблюдаются min-width / max-width)
- **flex-start**: элементы размещаются в начале поперечной оси.
- **flex-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



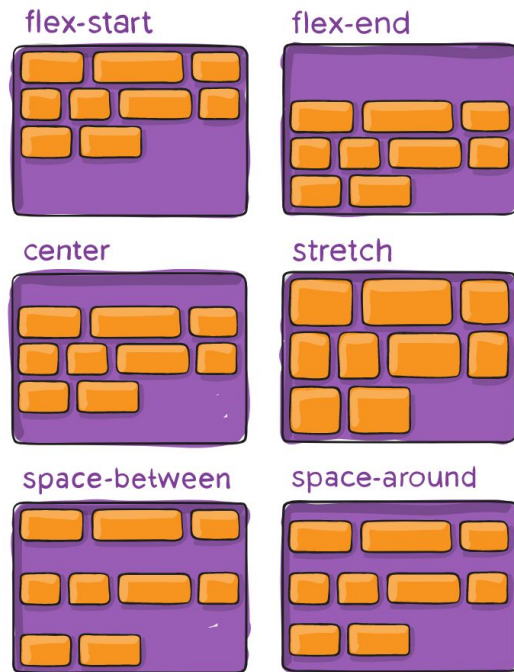
```
.container {  
  align-items: stretch | flex-start | flex-end | center | baseline  
}
```


Свойства для Родителя


align-content

Это свойство выравнивает линии в пределах flex контейнера, когда есть дополнительное пространство на поперечной оси

- **flex-start** : элементы, сдвинуты в начало контейнера.
- **flex-end** : элементы, сдвинуты в конец контейнера.
- **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



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



Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в
немецкой школе AIT TR GmbH