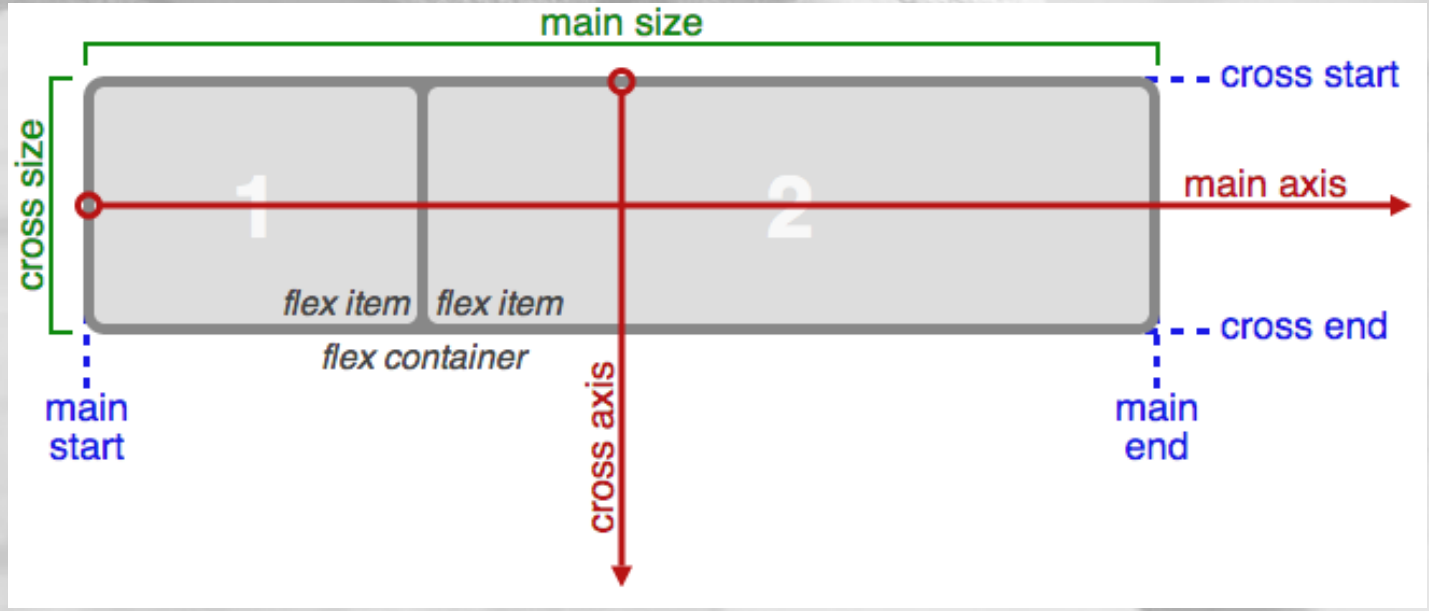

FLEXBOX

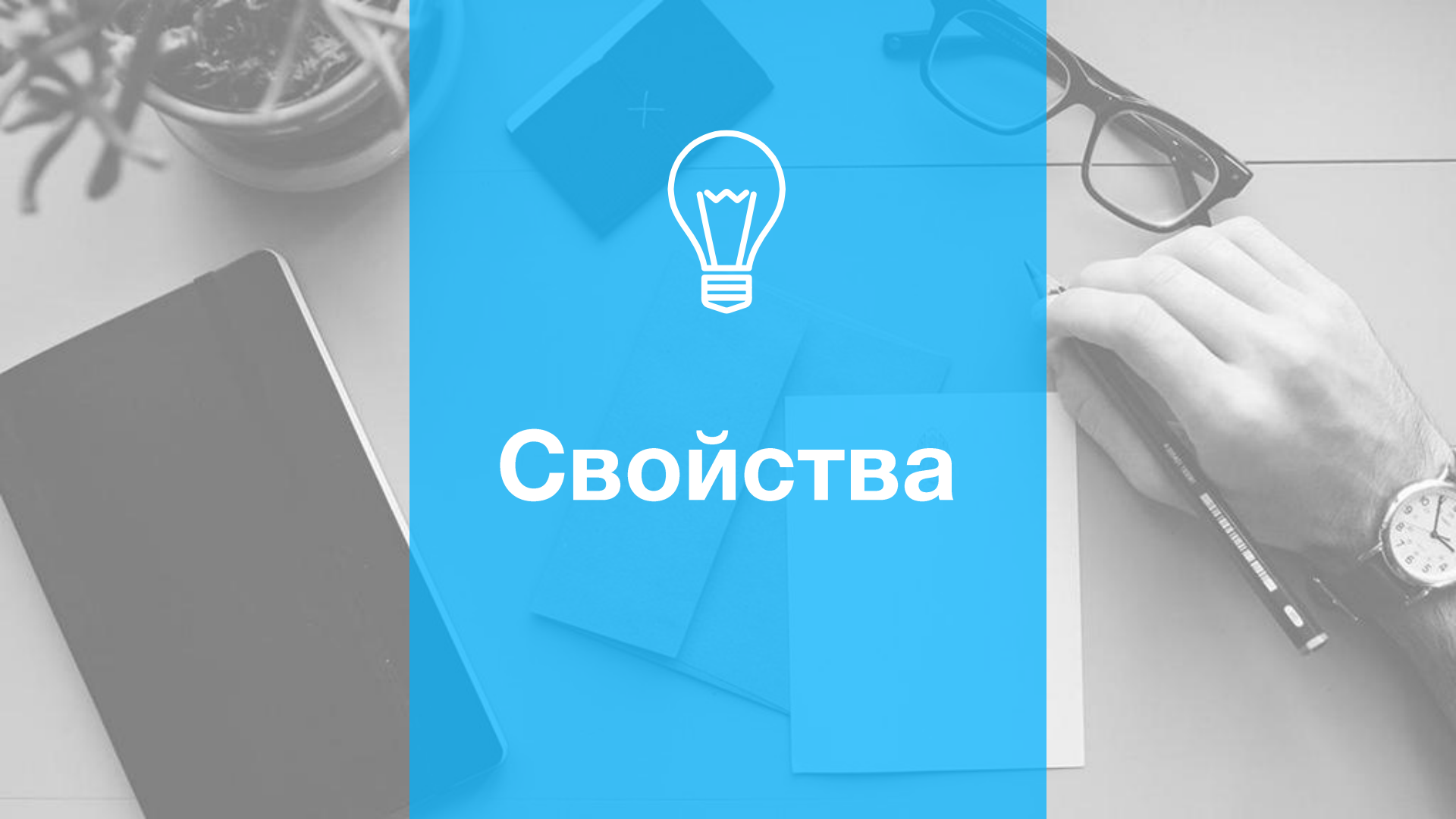
Основы

Во **FlexBox** есть два основных типа элементов:
Гибкий Контейнер (**Flex Container**) и
Гибкие Элементы (**Flex Item**) - его дочерние элементы
Для инициализации контейнера достаточно присвоить, через css, элементу **display: flex;** или **display: inline-flex;**.
Разница между flex и inline-flex заключается лишь в принципе взаимодействия с окружающими контейнер элементами, подобно display: block; и display: inline-block;, соответственно.

Внутри гибкого контейнера создаются две оси:
главная ось (**main-axis)** и
перпендикулярная или кросс ось (**cross axis**)
Преимущественно гибкие элементы выстраиваются именно по главной оси, а потом уже по кросс оси. По умолчанию главная ось горизонтальная и имеет направление слева направо, а кросс ось вертикальна и направлена сверху вниз.

ОСНОВНАЯ ИДЕЯ FLEX-ЛЕЙАУТОВ





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: flex | inline-flex;

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** (default): главная ось гибкого контейнера имеет ту же ориентацию, что и инлайн ось текущего режима направления строк. Начало (main-start) и конец (main-end) направления главной оси соответствуют началу (inline-start) и концу (inline-end) инлайн оси (inline-axis).
➜ **row-reverse**: все то же самое, что и в row только main-start и main-end меняются местами.
➜ **column**: так же, как и row, только теперь главная ось направлена сверху вниз.
➜ **column-reverse**: так же, как row-reverse, только главная ось направлена снизу вверх.

Пример

flex-wrap

Применяется к: родительскому элементу flex-контейнера.

По умолчанию все гибкие элементы в контейнере укладываются в одну строку, даже если не помещаются в контейнер, они выходят за его границы. Данное поведение переключается с помощью свойства **flex-wrap**.

flex-wrap: nowrap | wrap | wrap-reverse

У этого свойства есть три состояния:

➜ **nowrap** (default): гибкие элементы выстраиваются в одну строку слева направо.
➜ **wrap**: гибкие элементы строятся в многострочном режиме, перенос осуществляется по направлению кросс оси, сверху вниз.
➜ **wrap-reverse**: так же, как и wrap, но перенос происходит снизу вверх.

Пример

justify-content

Применяется к: родительскому элементу flex-контейнера.

Элементы в контейнере поддаются выравниванию при помощи свойства **justify-content** вдоль главной оси.

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

Это свойство принимает целых пять разных вариантов значений:

➜ **flex-start** (default): гибкие элементы выравниваются по началу главной оси.
➜ **flex-end**: элементы выравниваются по концу главной оси.
➜ **center**: элементы выравниваются по центру главной оси.
➜ **space-between**: элементы занимают всю доступную ширину в контейнере, крайние элементы вплотную прижимаются к краям контейнера, а свободное пространство равномерно распределяется между элементами.
➜ **space-around**: гибкие элементы выравниваются таким образом, что свободное пространство равномерно распределяется между элементами. Но стоит отметить, что пространство межу краем контейнера и крайними элементами будет в два раза меньше чем пространство между элементами в середине ряда.

Пример

align-items

Применяется к: родительскому элементу flex-контейнера.

Мы так же имеем возможность выравнивания элементов по кросс оси.
Это свойство позволяет выравнивать элементы в строке относительно друг друга.

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

Это свойство так же принимает пять разных вариантов значений:

- ➜ **flex-start**: все элементы прижимаются к началу строки.
- ➜ **flex-end**: элементы прижимаются к концу строки.
- ➜ **center**: элементы выравниваются по центру строки.
- ➜ **baseline**: элементы выравниваются по базовой линии текста.
- ➜ **stretch** (default): элементы растягиваются заполняя полностью строку.

Пример

align-content

Применяется к: родительскому элементу flex-контейнера.

Это свойство отвечает за выравнивание целых строк относительно гибкого контейнера. Оно не будет давать эффекта, если гибкие элементы занимают одну строку.

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

Это свойство принимает шесть разных вариантов значений:

➜ **flex-start**: все линии прижимаются к началу кросс-оси.
➜ **flex-end**: все линии прижимаются к концу кросс-оси.
➜ **center**: все линии паком выравниваются по центру кросс оси.
➜ **space-between**: линии распределяются от верхнего края до нижнего оставляя свободное пространство между строками, крайние же строки прижимаются к краям контейнера.
➜ **space-around**: линии равномерно распределяются по контейнеру.
➜ **stretch** (default): линии растягиваются занимая все доступное пространство.

Пример

order

Применяется к: дочернему элементу (flex-элементу)

Это свойство позволяет менять позицию в потоке конкретному элементу. По умолчанию все гибкие элементы имеют **order: 0;** и строятся в порядке естественного потока.

order: <integer>

Пример

flex-basis

Применяется к: дочернему элементу (flex-элементу)

С помощью этого свойства мы можем указывать базовую ширину гибкого элемента. По умолчанию имеет значение **auto**. Принимает значение ширины в px, %, em и остальных единицах. По сути это не строго ширина гибкого элемента, это своего рода отправная точка, относительно которой происходит растягивание или усадка элемента. В режиме auto элемент получает базовую ширину относительно контента внутри него.

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

flex-grow

Применяется к: дочернему элементу (flex-элементу)

Это свойство задает фактор увеличения элемента при наличии свободного места в контейнере. По умолчанию это свойство имеет значение 0.

Давайте представим, что у нас есть гибкий контейнер, который имеет ширину 500px, внутри него есть два гибких элемента, каждый из которых имеет базовую ширину 100px. Тем самым в контейнере остается еще 300px свободного места.
Если первому элементу укажем flex-grow: 2; а второму элементу укажем flex-grow: 1; эти блоки займут всю доступную ширину контейнера, только ширина первого блока будет 300px, а второго только 200px. Что же произошло? А произошло вот что: доступные 300px свободного места в контейнере распределились между элементами в соотношении 2:1, +200px первому и +100px второму.

flex-grow: <number> (default 0)

flex-shrink

Применяется к: дочернему элементу (flex-элементу)

Свойство так же задает фактор на изменение ширины элементов, только в обратную сторону. Если контейнер имеет ширину **меньше** чем сумма базовой ширины элементов, то начинает действовать это свойство.

Например контейнер имеет ширину 600px, а flex-basis элементов по 300px. Первому элементу укажем flex-shrink: 2; а второму flex-shrink: 1;
Теперь сожмем контейнер на 300px. Следовательно сумма ширины элементов на 300px больше чем контейнер. Эта разница распределяется в соотношении 2:1, получается от первого блока отнимаем 200px, а от второго 100px. Новый размер элементов получается 100px и 200px, у первого и второго элемента, соответственно. Если мы устанавливаем flex-shrink в значение 0, то мы запрещаем сжиматься элементу до размеров меньше чем его базовая ширина.

flex-shrink: <number> (default 1)

align-self

Применяется к: дочернему элементу (flex-элементу)

Позволяет переопределить выравнивание, заданное по умолчанию или в align-items, для отдельных flex-элементов.

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

Ссылки

Полное руководство по Flexbox:
http://frontender.info/a-guide-to-flexbox/

Примеры:
http://codepen.io/HugoGiraudel/pen/LklCv
http://codepen.io/HugoGiraudel/pen/pkwqH
http://codepen.io/HugoGiraudel/pen/qIAwr