**Преобразование во флекс-контейнер**

До сих пор мы верстали поток элементов. Те, что идут в HTML-коде первыми, всегда выше либо левее вторых. Если вы делаете новостной или продуктовый сайт, так не годится. Свежие новости и товары должны стоять выше. Вам также нужна свобода перестроения из ряда в колонку на узком экране мобильного телефона, свобода выравнивания всех элементов по краю или центру — вам вообще нужно больше свободы. Её даёт одна строчка CSS-кода:

Скопировать кодCSS

display: flex;

Это приказ браузеру: отображать элемент с таким свойством как гибкий блок. По английски — “flexible box“, или сокращённо — “flexbox“. Мы будем писать кириллицей: флексбокс или флекс.

Если вы забыли о свойстве display, перечитайте теорию в уроке по ссылке внизу. В вёрстке понимание этого свойства делает вас уверенным. Вы будете задавать свойство display постоянно, нужно делать это осмысленно.

Свойство display: flex; превращает элемент во флекс-контейнер (англ. flex container), а вложенные в него дочерние — во флекс-элементы (англ. flex items). Контейнер поступает со своими элементами как тренер с футболистами: размещает их по определённой схеме и прописывает поведение в меняющейся обстановке. В свою очередь игроки, то есть флекс-элементы, могут проявить инициативу, показав личные свойства: они меняются местами и сами выбирают себе длину маршрута. Всё это мы сейчас изучим.

Начнём с простого. В контейнере один дочерний элемент с единственным свойством margin: auto;

Флекс-контейнер управляет не элементом, а пустотой вокруг него. Если свойство содержит уточнение — например, margin-left: auto; — вся пустота оказывается слева, а элемент прижимается к правому краю контейнера.

А margin: auto; оказывает совершенно магическое действие. Пустота равномерно окружает элемент, и он оказывается точно посреди своего контейнера как по ширине, так и по высоте. Безо всякого свойства vertical-align. И это только начало!

**Флекс-контейнер с несколькими элементами**

Когда во флекс-контейнере несколько элементов, они сразу же выстраиваются в строку независимо от типа. Нет надобности переводить их значением inline-block в строчно-блочные. Поскольку возможности флексбокс-вёрстки на этом только начинаются, её охотнее используют для построения макетов, чем inline-block.

**Вложенные флекс-контейнеры**

Вложенный во флекс-контейнер элемент может сам стать флекс-контейнером. Так он сможет распоряжаться вложенными в него элементами. Заметим, что флекс-элементы подчиняются только своему «непосредственному начальнику» (родителю) — настройки флекс-контейнеров более высокого уровня до них не доходят.

# Направление внутри флекс-контейнера

По умолчанию флекс-элементы внутри контейнера отображаются в ряд слева направо в том порядке, в каком они прописаны в коде HTML-документа. Они следуют направлению потока. Флексбокс-вёрстка позволяет преобразовать ряд в колонку и обратить порядок следования, направив его против потока. Для этого контейнеру придают свойство flex-direction (англ. direction, «направление»). Его возможные значения:

* flex-direction: row; — обычный ряд (англ. row, «ряд»), элементы следуют в потоке, слева направо. Это значение по умолчанию, его можно и не прописывать.
* flex-direction: row-reverse; — ряд в обратном порядке (англ. reverse, «обратный ход»).
* flex-direction: column; — колонка (англ. column, «столбец») в обычном порядке. Очень пригождается при адаптации под мобильные устройства.
* flex-direction: column-reverse; — колонка в обратном порядке.

**Управление положением содержимого флекс-контейнера**

В предыдущем уроке мы рассказывали, как распределить флекс-элементы. Но сколько бы их ни было, остаётся ещё пустое пространство. Его тоже нужно распределять. В этом уроке расскажем, как это делать.

Это делается в двух направлениях — вдоль оси, по которой расставлены элементы (т.е. вдоль строки/колонки), и поперёк. Регулировка вдоль строки либо колонки задаётся свойством justify-content. Название пришло из механики, где юстировку начинают с выравнивания деталей вдоль главной оси конструкции (например, объектива).

Возможные значения свойства justify-content:

* flex-start: сначала идут флекс-элементы, за ними — оставшееся пустое пространство. Если значение flex-direction — row, все флекс-элементы прижмутся к левому краю контейнера. А справа окажется пустота.

Если flex-direction: column, элементы идут сверху вниз. И свойство justify-content: flex-start соберёт флекс-элементы сверху, а пустое пространство окажется снизу.

justify-content: flex-start — это значение по умолчанию.

* flex-end: сначала идёт пустое пространство, а к концу контейнера прижаты элементы. При flex-direction: row, элементы окажутся справа, а всё пустое пространство — слева. При flex-direction: row-reverse — наоборот.
* center: флекс-элементы собраны посередине, а пустое пространство распределено поровну с двух сторон от них.
* space-between: первый элемент прижат к началу контейнера, последний — к концу, а остальные расставлены между ними с одинаковыми промежутками. Подходит для меню, растянутого во всю ширину страницы.
* space-around: всё пустое пространство делится на равные промежутки. Каждый флекс-элемент получает отступы по обе стороны, равные этому промежутку. Так между флекс-элементами оказывается отступ в 2 промежутка, а перед первым и после последнего элементов — в 1 промежуток.
* space-evenly: все отступы равной величины. И между элементами, и между границами контейнера и первым и последним элементами.

**Выравнивание флекс-элементов в контейнере**

В предыдущих уроках мы разобрались с порядком элементов во флекс-контейнере. Также мы рассказали, как управлять пустым пространством вокруг них. Теперь возьмёмся за выравнивание.

Когда флекс-элементы разной высоты выстроены в ряд, или столбец составлен из элементов разной ширины, их также можно выровнять по определённой линии. Предназначенное для этого свойство так и называется align-items, в переводе с английского «выровнять элементы по линии». Эта линия проходит вдоль направления перечисления элементов. Так, если flex-direction задаёт расстановку в ряд, то линией может быть верхний край контейнера, нижний край, центральная ось и так называемая базовая линия — граница, где первый элемент разделяется на две части (например, на текст и картинку).

Возможные значения свойства align-items:

* stretch («растянуть») — установлено по умолчанию. Вместо выравнивания элементы растянуты от одного края контейнера до другого: в ряду сверху донизу, в колонке справа налево. Значение stretch стало большим достижением флексбокс-вёрстки. Оно позволяет придать всем элементам одинаковый размер, не прибегая к определению ширины и высоты.
* flex-start (по началу флекс-контейнера) — когда направление контейнера row или row-reverse, все элементы прижаты к верхнему краю контейнера, а при значениях column или column-reverse — к левому.
* flex-end (по концу флекс-контейнера) — когда направление контейнера row или row-reverse, все элементы прижаты к нижнему краю контейнера, а при значениях column или column-reverse — к правому.
* center («центрировать») — центральная ось контейнера делит каждый элемент пополам.
* baseline («по базовой линии») — базовые линии всех флекс-элементов становятся продолжением базовой линии первого по порядку.

# Перенос элементов при сжатии окна браузера

По умолчанию флекс-контейнер делает окно "резиновым": при сжатии или расширении окна просмотра браузера отдельные флекс-элементы автоматически настраивают свою ширину или высоту так, чтобы поместиться в контейнер.

Это поведение можно изменить так, чтобы при сжатии элементы переносились на новую строку или колонку.

Для этого свойству flex-wrap задают значение wrap (англ. wrap, «завернуться»). У флекс-контейнера с таким свойством последние элементы заворачиваются (переходят) на новую строчку, если не умещаются на текущей.

Значение wrap-reverse («перенос наоборот») позволяет устроить так, чтобы переносились не последние, а первые элементы строки/колонки.

Свойство flex-wrap есть у флекс-контейнера всегда, только по умолчанию ему задано значение nowrap, запрещающее перенос.

Два свойства **flex-direction** и **flex-wrap** используются так часто вместе, что было создано свойство **flex-flow** для их комбинирования. Это свойство принимает значения двух этих свойств, разделеные пробелом.

Например, ты можешь использовать **flex-flow: row wrap**, чтоб элементы располагались в ряд и автоматически переносились на новую строку.

# Свойства флекс-элементов. Порядок

Вот заданы главные настройки флекс-контейнера: порядок отображения, распределение пустот и правила переноса элементов. Это похоже на тренерские указания команде футболистов, которые получили схему игры на разные случаи.

Пора перейти к собственным возможностям «игроков», т.е. персональным свойствам отдельных элементов.

Так, футболисты, не нарушая схему своего наставника, могут меняться местами и ролями: центральный полузащитник может стать правым.

Во флексбокс-вёрстке такое можно проделывать, не трогая ни HTML-код, ни CSS-правило для контейнера. В правила тех флекс-элементов, чей порядок следования нужно переопределить, вводится свойство order.

Элемент, которому задано CSS-свойство order: 1; будет показан раньше элемента со свойством order: 2; (здесь могут быть любые целые числа, лишь бы одно из них было меньше другого).

Элемент с отрицательным значением порядка (order: -1;) будет первым в контейнере, если у других элементов значения положительны или даже не заданы («капитан команды»). Весьма удобно для главной новости на сайте или меню.

И наоборот, если порядок с положительным значением задан одному-единственному элементу, тот закрепляется на последнем месте («вратарь»).

**Свойства флекс-элементов. Выравнивание отдельного элемента**

Заданное флекс-контейнеру свойство align-items применяется ко всем элементам сразу. Иногда нужно, чтобы отдельный элемент не подчинялся общему правилу. На помощь приходит свойство align-self (по-английски примерный смысл этого словосочетания — «равняться на себя»). Оно применяется непосредственно для уникального флекс-элемента и принимает те же значения, что и align-items:

* flex-start,
* flex-end,
* center,
* baseline,
* stretch.

Так футболист может выбиться из общей схемы, взяв на себя дополнительную роль. Например, если команда играет в меньшинстве. Это указание не всей сборной, а только ему лично.

# Свойства флекс-элементов. Базовая ширина элемента

Иногда необходимо, чтобы один флекс-элемент имел определённый размер, пока другие свойства флекс-контейнера не начнут его сжимать или расширять. Для этого есть свойство flex-basis (от англ. «базис», базовый размер флекс-элемента). Значение задаётся в пикселях.

Это ширина, если флекс-элементы отображаются в ряд, и высота — если в колонку.

Найдите в вёрстке элемент с классом menu. Сразу после него создайте div с классом main. Поместите в этот новый div все остальные элементы, кроме элементов с классами header, menu и footer.

Так все блоки с классом flex-row со своим содержимым станут дочерними для блока с классом main.

А сейчас вернём макет в менее адаптивный, но более привлекательный вид. Определите стили для классов main, header, menu и footer. Правило стилей для селектора .main нужно создать, а в .header, .menu и .footer добавить новые свойства:

max-width: 1330px;

min-width: 1280px;

margin: auto;

Элементы с классом flex-row выглядят неаккуратно: между обложками и рамкой видны узкие белые щели. Чтобы избавиться от них, прижмите элементы флекс-строки к краям родителя. Для этого найдите правило стилей для класса flex-row. Измените значение свойства justify-content на space-between.

Уберём красные границы. Они были нужны нам, чтобы видеть, как работают флекс-свойства. Но поскольку менять эти свойства мы больше не будем, красные границы можно убирать.

Найдите правило для всех элементов div и удалите его.

Сделаем шрифт помоднее.

Установите странице шрифт 'Cormorant Garamond'. Для этого задайте свойство font-family для элемента body. Напишите это правило в самом начале CSS-файла.

Оформим ссылки. Селектор ссылки в шапке — с текстом «Яндекс Практикум» — уже вами описан в CSS. Найдите селектор .header-link. Определите для него размер шрифта 28 пикселей.

Затем создайте селектор ссылок в меню .menu-link. Укажите в нём размер шрифта 18 пикселей и внешний правый отступ 20 пикселей.

Пропишите и ссылкам в меню menu-link, и ссылке в шапке CSS-правила:

уберите подчёркивание;

задайте цвет текста #10191E.

Добавьте внешние отступы сверху и снизу селекторам:

* .menu — 40 пикселей;
* .flex-row\_reversed — 60 пикселей.

Присваивайте значение свойству в одну строку, если помните, как это делается. А если не помните — загляните в подсказку:

Селекторам .menu и .flex-row\_reversed задайте свойство margin со значениями 40px auto и 60px auto соответственно.

# Ещё о флекс-контейнере

Есть свойство align-content. Оно отвечает за то, как поведут себя элементы при переносе на новую строку или колонку при включённом flex-wrap. Когда одна строка заполнена, где должна оказаться следующая: у другого края контейнера? По центру? Прямо рядом c предыдущей строкой? За это и отвечает свойство align-content.

Свойство align-content похоже на justify-content. Но justify-content отвечает за расположение элементов вдоль оси флекс-контейнера, а align-content — поперёк этой оси при переносе. То есть вдоль «поперечной» оси флекс-контейнера.

Без включённого flex-wrap: wrap; свойство align-content работать не будет вовсе.

Посмотрите на гифках, как работает свойство align-content. В примерах продольная ось флекс-контейнера везде направлена слева направо →, а поперечная — сверху вниз ↓.

При значении align-content: flex-start; блоки выстраиваются от начала поперечной оси флекс-контейнера. Сначала заполняется самая верхняя строчка контейнера — та, что находится у начала поперечной оси. Когда на первой строке места не остаётся, элементы переносятся на следующую строку. При этом элементы стараются располагаться как можно ближе к началу поперечной оси флекс-контейнера — то есть к верху.

В случае, когда align-content: flex-end;, блоки ведут себя похожим образом, но прижимаются к концу поперечной оси — к низу блока. Поэтому сначала заполняется нижняя строчка. Когда места не остаётся, заполняется вторая строчка снизу. И так далее.

Значение align-content: center; заставляет элементы выравниваться относительно центра поперечной оси.

Если установить align-content: space-around;, всё свободное пространство поделится на равные промежутки. Эти промежутки добавятся флекс-элементам сверху и снизу.

Значение align-content: space-between; старается разместить элементы так, чтобы между ними были равные отступы. Такие же отступы соблюдаются между элементами и границами флекс-контейнера.

Последнее значение свойства align-content — это stretch. Оно растягивает элементы так, чтобы они занимали всю поперечную ось флекс-контейнера.

Обратите внимание: мы использовали в описании понятия «верх» и «низ». Эти понятия удобны, потому что интуитивно понятны. Но помните: поведение элементов зависит от направления осей флекс-контейнера.

## **Ещё про флекс-элементы**

У элементов есть достаточно экзотические свойства flex-grow и flex-shrink. Эти свойства отвечают за распределение пространства, отведённого элементу в случаях, когда пространство родителя больше, чем нужно элементам (это flex-grow) или меньше (это flex-shrink).

Сначала расскажем про flex-grow и вспомним немного школьной математики. Когда элементы занимают не весь контейнер, остаётся свободное пространство. Это пространство можно распределить — «отдать» элементам в разных пропорциях. Например, 5/10 прибавить к первому элементу, 3/10 ко второму и 2/10 к третьему. В этом случае каждому элементу указывают flex-grow, равный числителю этих дробей: 5-ти, 3-м или 2-м.

Обратите внимание, что пропорционально распределяется именно оставшееся пространство, а не пространство всего контейнера.

Если вы любите формулы — [вот перевод хорошей статьи](https://medium.com/@stasonmars/%D0%BA%D0%B0%D0%BA-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%D0%B5%D1%82-flex-grow-%D0%B2-css-%D0%BF%D0%BE%D0%B4%D1%80%D0%BE%D0%B1%D0%BD%D0%BE%D0%B5-%D1%80%D1%83%D0%BA%D0%BE%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE-557d406be844) про flex-grow.

Flex-shrink работает по обратному принципу. Если элементы не умещаются в контейнер, браузер считает, сколько места не хватает. Далее он высчитывает, насколько нужно уменьшить элемент, ориентируясь на недостаток места, размер и общий вес элементов. Понять математику и увидеть формулы можно [в статье-продолжении](https://medium.com/@stasonmars/%D0%BA%D0%B0%D0%BA-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%D0%B5%D1%82-flex-shrink-%D0%B2-css-%D0%BF%D0%BE%D0%B4%D1%80%D0%BE%D0%B1%D0%BD%D0%BE%D0%B5-%D1%80%D1%83%D0%BA%D0%BE%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE-c41e40767194).