В [предыдущей части](https://htmlacademy.ru/courses/359) мы научилиcь управлять потоком документа и верстать крупные сеточные элементы. В этой части мы займёмся мелкими сетками, их ещё называют микросетками. Учиться верстать микросетки будем на примере *компонентов*, или блоков, которые часто встречаются на сайтах: навигации и списка карточек.

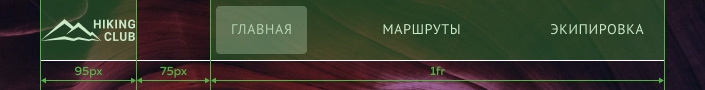
В отличие от крупных сеток, микросетки меньше зависят от макета и больше — от содержимого.

Для начала сверстаем блок навигации, состоящий из логотипа и списка ссылок. Ссылка на текущую страницу должна отличаться от остальных. Предположим, что мы верстаем навигацию для главной страницы сайта:



Содержимое навигации нужно расположить в две колонки и добавить отступ между ними. Левая колонка должна иметь фиксированный размер, а правая занять всё оставшееся пространство.

Для создания подобной сетки удобно использовать гриды. Эта технология позволяет легко создавать как фиксированные, так и тянущиеся колонки, а также управлять отступами между ними.



В прошлой части мы разбирали, как изменять [тип бокса](https://htmlacademy.ru/courses/359/run/4), описывать [шаблон грид-контейнера](https://htmlacademy.ru/courses/359/run/5), работать со [свободным пространством](https://htmlacademy.ru/courses/359/run/8) и задавать отступы между [колонками](https://htmlacademy.ru/courses/359/run/9). Эти навыки пригодятся нам, чтобы сверстать сетку навигации.

.grid-container {

display: grid;

grid-template-columns: 1fr 50px;

column-gap: 5px;

}

Первым делом для наглядности подключим стилевой файл с подсветкой боксов.

После сделаем элемент с классом navigation грид-контейнером и разделим его содержимое на две колонки с помощью свойства grid-template-columns. Затем используем свойство column-gap, чтобы добавить отступ между колонками.

**Флекс внутри грида**

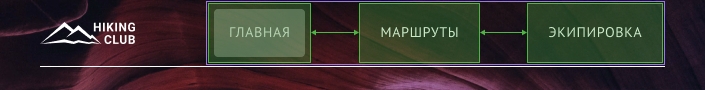
Разделив навигацию на две колонки, мы получили в одной из них логотип, а в другой — список ссылок. К логотипу мы вернёмся чуть позже, а пока займёмся списком. По умолчанию элементы списка имеют блочный тип бокса. Поэтому они растягиваются на всю ширину родителя и выстраиваются сверху вниз.

ГЛАВНАЯ

МАРШРУТЫ

ЭКИПИРОВКА

Нам нужно уменьшить ширину элементов списка, расположить их в ряд и распределить между ними свободное пространство.



Проще всего это сделать с помощью [флексов](https://htmlacademy.ru/courses/359/run/13), потому что флекс-элементы по умолчанию выстраиваются горизонтально и занимают столько места, сколько нужно их содержимому, включая отступы.

.flex-container {

display: flex;

}

Кроме того, флексы позволяют удобно управлять [выравниванием элементов](https://htmlacademy.ru/courses/359/run/15). По умолчанию флекс-элементы начинаются у левой границы контейнера и идут вплотную друг за другом. Но это можно изменить с помощью свойства justify-content. Оно задаётся флекс-контейнеру и отвечает за выравнивание флекс-элементов по главной оси.

Нам нужно, чтобы первый и последний флекс-элемент прижались к краям контейнера, а свободное пространство распределилось поровну между всеми элементами. Такое выравнивание задаёт свойство justify-content со значением space-between.

.flex-container {

display: flex;

justify-content: space-between;

}

Обратите внимание, мы сделали грид-контейнером сам блок навигации, а теперь используем флексбокс для вложенного в него списка ссылок. Необязательно использовать при вёрстке только какую-то одну технологию, их можно комбинировать.

Итак, превратим список с классом navigation-list во флекс-контейнер, чтобы элементы списка выстроились горизонтально. Затем выровняем флекс-элементы по главной оси.

**Управляем отступами**

С помощью флексов мы расположили элементы списка в ряд, а также выровняли их по главной оси. Чтобы наш компонент стал походить на макет, осталось поработать с отступами — обнулить отступы у списка и добавить их ссылкам.



Отступы у боксов бывают [внутренними](https://htmlacademy.ru/courses/359/run/6) (padding) и [внешними](https://htmlacademy.ru/courses/359/run/7) (margin). Если требуется задать одинаковые отступы со всех сторон, удобно использовать краткую запись:

.element {

margin: 0;

padding: 10px;

}

Для соответствия макету мы можем задать *внешние отступы* как элементам списка, так и вложенным в них ссылкам. Но *внутренние отступы* следует добавлять именно ссылкам. Внутренние отступы добавляют ссылкам для того, чтобы увеличить область, по которой можно кликнуть (её ещё называют активной областью). Чем проще попасть по ссылке, тем удобнее интерфейс.

А вот задавать ссылкам фиксированную [ширину](https://htmlacademy.ru/courses/359/run/14) не рекомендуется. В отличие от крупных сеток, где все размеры брались из макета, в микросетках ширина элементов чаще всего зависит от содержимого. Текст у ссылок разной длины, кроме того, он со временем может измениться. Если задать ссылкам точную ширину, то при изменении текста вёрстка сломается.

Итак, обнулим у списка navigation-list внешние и внутренние отступы по умолчанию. Затем добавим отступы ссылкам с классом navigation-link.

# Вертикальные отступы у строчных боксов

В прошлом задании мы добавили ссылкам внешние и внутренние отступы. Но внешние отступы по вертикали не появились, а внутренние вышли за пределы навигации — из-за этого фон у ссылки на текущую страницу налез на белую линию.

ГЛАВНАЯ

МАРШРУТЫ

ЭКИПИРОВКА

Так произошло потому, что по умолчанию ссылки имеют [строчный тип бокса](https://htmlacademy.ru/courses/359/run/1). Браузер игнорирует внешние отступы по вертикали у строчных боксов, а их внутренние отступы сверху и снизу не влияют на расположение других элементов и высоту строки.

Рассмотрим пример. Есть четыре ссылки на двух строках. Каждой ссылке заданы внутренние и внешние отступы со всех сторон и свой цвет фона:

Ссылка цвета бедра испуганной нимфыАпельсиновая ссылка  
Ляпис-лазурная ссылкаСсылка цвета молодой листвы

Обратите внимание, между ссылками появился горизонтальный отступ, потому что сработали внутренние и внешние отступы по горизонтали. Но при этом внешние отступы по вертикали у ссылок не появились, а внутренние отступы сверху и снизу наложились друг на друга и даже вышли за пределы контейнера. Это обычное поведение для строчных боксов.

Нам нужно, чтобы у ссылок в навигации заработали внешние отступы сверху и снизу, а внутренние отступы по вертикали не выходили за границы навигации. Самый простой способ добиться этого — изменить у ссылок тип бокса. Например, сделать ссылки блочными боксами:

.element {

display: block;

}

У блочных боксов работают все внешние отступы, а внутренние отступы по вертикали ведут себя так же, как и по горизонтали.

Мы уже превращали блочные боксы в грид- и флекс-контейнеры. Точно так же мы можем превращать строчные боксы в блочные и наоборот. Это не влияет на семантику тега.

Превратим ссылки в блочные боксы и убедимся, что вертикальные отступы заработали.

# Конспект «Микросетки». Раздел 1

## Микросетки

Микросетки — сетки мелких элементов веб-страницы.

В отличие от крупных сеток, микросетки меньше зависят от макета и больше — от содержимого. Содержимое страниц со временем может меняться. Если не учитывать этого, вёрстка сломается.

## Отступы у ссылок

Часто ссылкам добавляют внутренние отступы, чтобы увеличить область, по которой можно кликнуть (её ещё называют активной областью). Чем проще попасть по ссылке, тем удобнее интерфейс.

По умолчанию ссылки имеют строчный тип бокса. Браузер игнорирует внешние отступы по вертикали у строчных боксов, а их внутренние отступы сверху и снизу не влияют на расположение других элементов. Самый простой способ решить эту проблему — изменить у ссылок тип бокса. Например, сделать их блочными боксами:

.element {

display: block;

}

## Свойство align-items

По умолчанию грид-элементы занимают всё доступное пространство по высоте. Такое поведение можно изменить с помощью свойства align-items. Оно задаётся грид-контейнеру и управляет выравниванием грид-элементов по вертикали.

.grid-container {

display: grid;

align-items: start;

}

У align-items могут быть следующие значения:

* stretch — значение по умолчанию; элементы начинаются у верхней границы и растягиваются на всю высоту.
* start — элементы выстраиваются по верхней границе и, если содержимого немного, не тянутся на всю высоту;
* end — элементы выстраиваются по нижней границе;
* center — элементы располагаются по центру;

## Свойство flex-wrap

Если элементам не хватает места, они могут вылезти за границы контейнера. Такое поведение называют *выпадением элементов*. Также говорят, что произошло *переполнение*.

По умолчанию флекс-контейнер однострочный. Чтобы элементы не выпадали из контейнера, его делают многострочным. Для этого используют свойство flex-wrap со значением wrap.

.flex-container {

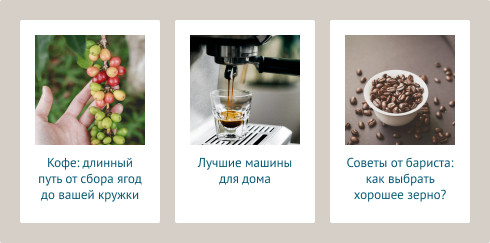
display: flex;

flex-wrap: wrap;

}

# Верстаем список с justify-content

Мы задали карточкам ширину и внутренние отступы. Пока карточки выстроены сверху вниз, а нам нужно расположить их в ряд. Используем для этого флексы.



Первая и последняя карточки в ряду должны прижиматься к границам контейнера, а свободное пространство нужно распределить равномерно между всеми карточками. Для этого подходит свойство justify-content со значением space-between.

Карточек в списке может быть много. Поэтому добавим ещё несколько штук в наш список и разрешим флекс-элементам переноситься на новую строку. Мы уже делали это, когда [верстали навигацию](https://htmlacademy.ru/courses/365/run/6).

Итак, превратим список карточек во флекс-контейнер и зададим ему выравнивание по главной оси. После этого добавим в список ещё пару карточек и, чтобы они не выпадали, сделаем флекс-контейнер многострочным.

# Верстаем список с gap

Мы посмотрели, как будет выглядеть список, если в ряду окажется две или четыре карточки. Так как мы убрали отступ у каждой третьей карточки, то третья и четвёртая карточки начали слипаться. Кроме того, справа у границы контейнера снова появился лишний отступ.



Кофе: длинный путь от сбора ягод до вашей кружки



Лучшие машины для дома



Советы от бариста: как выбрать хорошее зерно?



Рецепты на новогодние праздники

Мы не знаем, какой ширины окажется окно браузера у пользователя и какая карточка будет в ряду последней. Из-за этого мы не можем использовать :nth-child, чтобы убрать у неё отступ справа. Как быть?

К сожалению, простого решения этой проблемы для флексов нет. А вот свойство gap, которое есть у гридов, придётся как нельзя кстати. Это свойство позволяет задавать отступы между элементами и не влияет на расстояние до границ контейнера.

.grid-container {

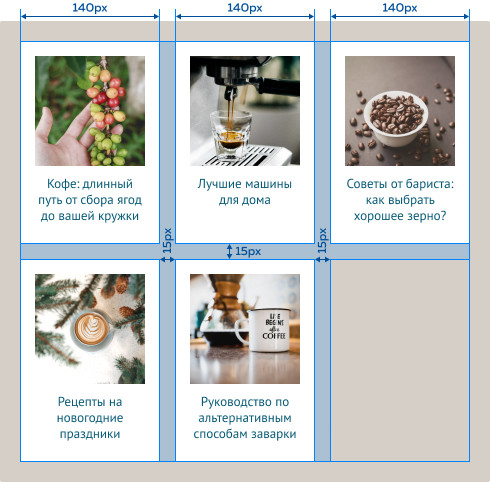
display: grid;

gap: 10px;

}

Напишем новые стили для списка карточек, на этот раз используя гриды.

Так как при описании шаблона грид-контейнера мы указываем ширину колонок, то задавать ширину самим карточкам больше не нужно.



Все колонки в списке должны быть одинаковой ширины. В этом случае удобно использовать значение-функцию repeat (от английского «повторить»). В скобках после repeat указывают количество колонок и их ширину. Значения разделяют запятой:

grid-template-columns: repeat(количество колонок, ширина колонки);

Этот код разобьёт грид-контейнер на четыре колонки шириной 100px:

.grid-container {

display: grid;

grid-template-columns: repeat(4, 100px);

}

Удалим весь код, который мы написали для реализации на флексах. После этого превратим список карточек в грид-контейнер и добавим отступы между карточками с помощью свойства gap. Используем repeat, чтобы разделить список на колонки, и попробуем задавать разные значения.

Некоторые браузеры позволяют использовать gap и внутри флекс-контейнера, но пока [далеко не все](https://caniuse.com/flexbox-gap).

# Убираем лишний отступ

Мы изменили значение justify-content и использовали внешние отступы, чтобы разделить карточки. Но в результате вместо трёх карточек в ряду оказалось всего две. Третья карточка не поместилась.

Разберём, почему так вышло. Мы добавили отступ справа всем карточкам, и у третьей карточки он тоже появится. В контейнере для него места не оказалось, и третья карточка перенеслась на новый ряд.



Кофе: длинный путь от сбора ягод до вашей кружки



Лучшие машины для дома



Советы от бариста: как выбрать хорошее зерно?



Рецепты на новогодние праздники

Чтобы вернуть третью карточку на место, отступ справа у неё придётся убрать. Для этого удобно использовать *псевдокласс*.

Псевдокласс — это дополнение к обычным селекторам, которое делает их точнее. Псевдокласс добавляется к селектору c помощью двоеточия. Подробнее о псевдоклассах рассказано в части «[Селекторы. Знакомство](https://htmlacademy.ru/courses/42)».

Нам нужно убрать отступ справа у каждой третьей карточки. Для этого подойдёт псевдокласс :nth-child. Он позволяет выбрать дочерний элемент по его порядковому номеру:

// Выберет второй элемент с классом item

.item:nth-child(2) { ... }

// Выберет каждый второй элемент с классом item

.item:nth-child(2n) { ... }

Поправим наш список. Используем селектор с псевдоклассом, чтобы убрать лишний отступ справа у каждой третьей карточки.

Подробнее об :nth-child рассказано в демонстрации [Использование псевдокласса: nth-child](https://htmlacademy.ru/demos/2#3).

**Значение auto-fit**

В прошлом задании мы использовали repeat, чтобы создать несколько колонок. Их количество было фиксированным: какое число указано в скобках, на столько колонок и разделялось содержимое контейнера. Но нам нужно сделать так, чтобы количество колонок зависело от ширины окна.

Для этого используем специальное значение auto-fit. Его пишут в скобках после repeat вместо числа колонок:

grid-template-columns: repeat(auto-fit, ширина колонки);

Значение auto-fit указывает, что колонок должно быть столько, сколько может поместиться в грид-контейнере.

.grid-container {

display: grid;

grid-template-columns: repeat(auto-fit, 100px);

}

Код в примере разделит контейнер на колонки шириной 100px.

**repeat (auto-fit, 100px)**

width: 200px;

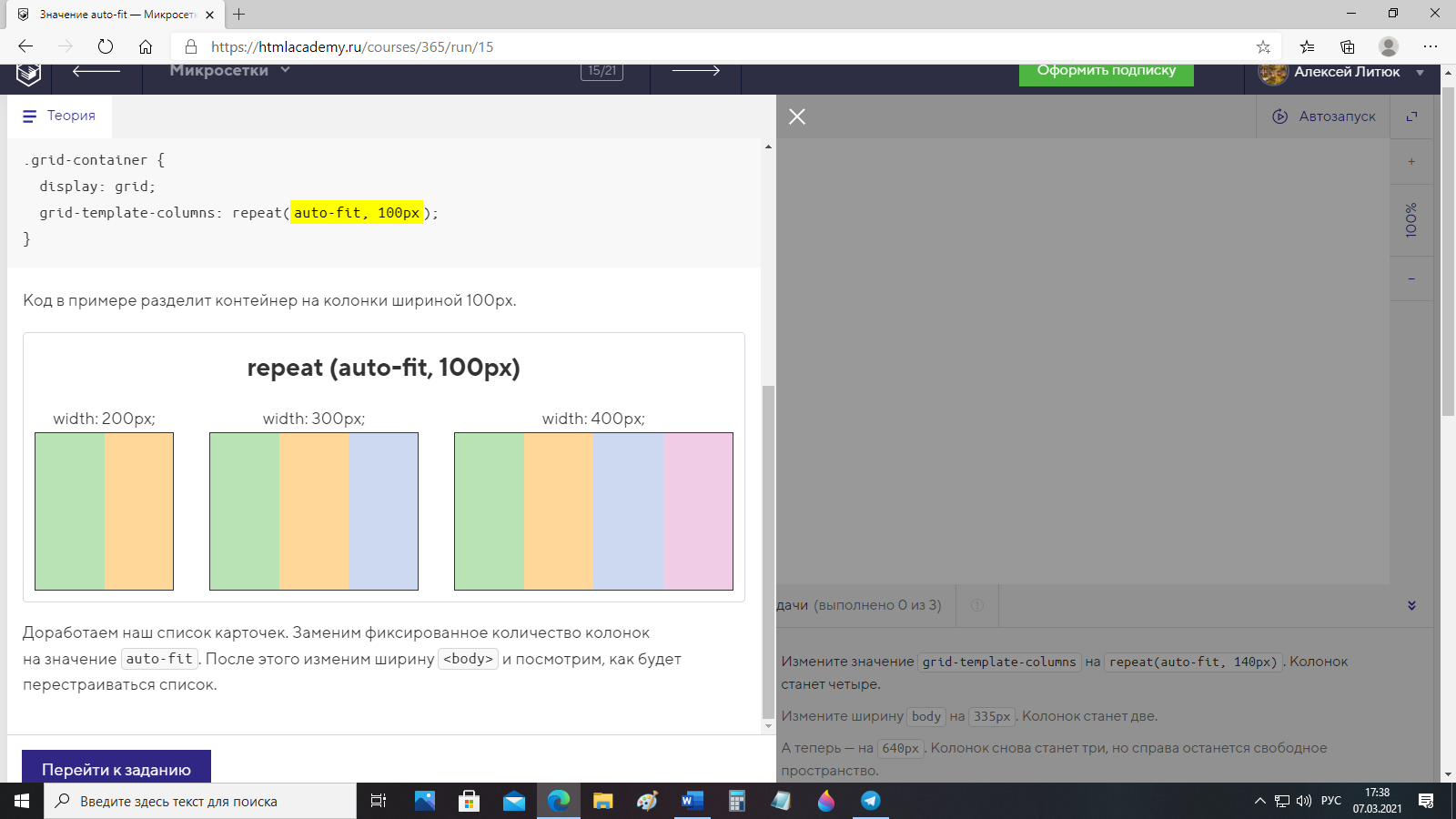
width: 300px;

width: 400px;

Доработаем наш список карточек. Заменим фиксированное количество колонок на значение auto-fit. После этого изменим ширину <body> и посмотрим, как будет перестраиваться список.

Перейти к заданию

* index.html
* **style.css**



Закрыть

* Теория

**Значение minmax**

Мы сделали так, чтобы количество колонок в списке зависело от ширины страницы. Но если пространство по ширине не получается разделить на количество колонок без остатка, то справа остаётся много свободного места.

Чтобы от него избавиться, можно увеличить ширину колонок. Но фиксированная ширина проблему не решит — если страница ещё увеличится, то свободное пространство появится снова. Поэтому нужно, чтобы ширина колонок изменялась динамически и зависела от наличия свободного пространства.

Это можно сделать с помощью ещё одного значения-функции minmax. Его указывают в repeat вместо фиксированной ширины колонок. В скобках после minmax задают минимальный и максимальный размеры колонок, они разделяются запятой:

repeat(auto-fit, minmax(минимальный размер, максимальный размер));

Код ниже позволит колонкам изменять ширину в зависимости от имеющегося в контейнере свободного пространства. При этом колонки не станут меньше 100px и не растянутся больше, чем на 150px:

.grid-container {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(100px, 150px));

}

В minmax в качестве максимального значения часто используют единицу измерения fr. Она позволяет колонкам увеличивать ширину до тех пор, пока свободного пространства в контейнере не хватит на ещё одну колонку.

**minmax (100px, 1fr)**

width: 200px;

width: 299px;

width: 300px;

В шаблоне списка укажем с помощью minmax минимальный и максимальный размеры колонок и понаблюдаем, как перестроится список, если изменить ширину страницы. После этого посмотрим, что случится, если добавить в список карточку, чьё содержимое шире колонки.

# Свойство grid-column

Мы добавили в список новую карточку, содержимое которой не помещается в одну колонку. Дизайнер настаивает, что уменьшать текст и картинку нельзя. Как быть? Разрешим карточке занять две колонки.

Сделать это можно с помощью свойства grid-column:

.element {

grid-column: span 2;

}

Элемент из примера растянется на две колонки:

1

2

3

4. Карточка, занимающая две колонки.

5

Чтобы растянуть грид-элемент, используют слово span, а после указывают число колонок, которые элемент должен занять:

// Этот элемент растянется на три колонки

.wide-element {

grid-column: span 3;

}

// А этот – на четыре

.very-wide-element {

grid-column: span 4;

}

Нам нужно, чтобы карточка в списке заняла две колонки. Добавим ей дополнительный класс и используем свойство grid-column, чтобы растянуть её. Поэкспериментируем с разными значениями span и разной шириной страницы.

# Свойство grid-row

Мы заставили карточку растянуться на несколько колонок. Дизайнеру идея понравилась, и он решил ни в чём себя не ограничивать. Так в списке появилась карточка, которой требуется две колонки *и два ряда*. Она уже добавлена в разметку, осталось только написать правильные стили.

Растянуть элемент на несколько рядов можно с помощью свойства grid-row. Ключевое слово span в нём работает так же, как в grid-column, только означает не количество колонок, которые элемент должен занять, а количество рядов:

.long-element {

grid-row: span 2;

}

Элемент из примера растянется на два ряда:

1

2

3  
Карточка, занимающая два ряда.

4

5

Свойства grid-column и grid-row можно использовать одновременно. Например, такой код заставит элемент занять три колонки и два ряда:

.element {

grid-column: span 3;

grid-row: span 2;

}

Карточка, которая в два раза шире и выше других, имеет класс huge. Скажем ей растянуться на два ряда и две колонки.

# Свойство grid-auto-flow

Мы сделали так, чтобы карточка с классом huge занимала два ряда и две колонки. Из-за этого в первом ряду образовалась пустая ячейка. Так произошло потому, что в ряду осталось место только для одной колонки, а следующая карточка занимает две. В результате карточка перенеслась на новый ряд, а предыдущий ряд остался незаполненным.

Такое поведение можно изменить с помощью свойства grid-auto-flow. Оно управляет автозаполнением грид-контейнера.

.grid-container {

display: grid;

grid-auto-flow: row;

}

Значение по умолчанию row говорит располагать элементы в том порядке, в котором они идут в разметке, и при необходимости создавать новые ряды:

1

2

3. Карточка, занимающая две колонки.

4

5

Но если указать значение dense, то контейнер будет заполняться так, чтобы не было пропусков:

1

2

3. Карточка, занимающая две колонки.

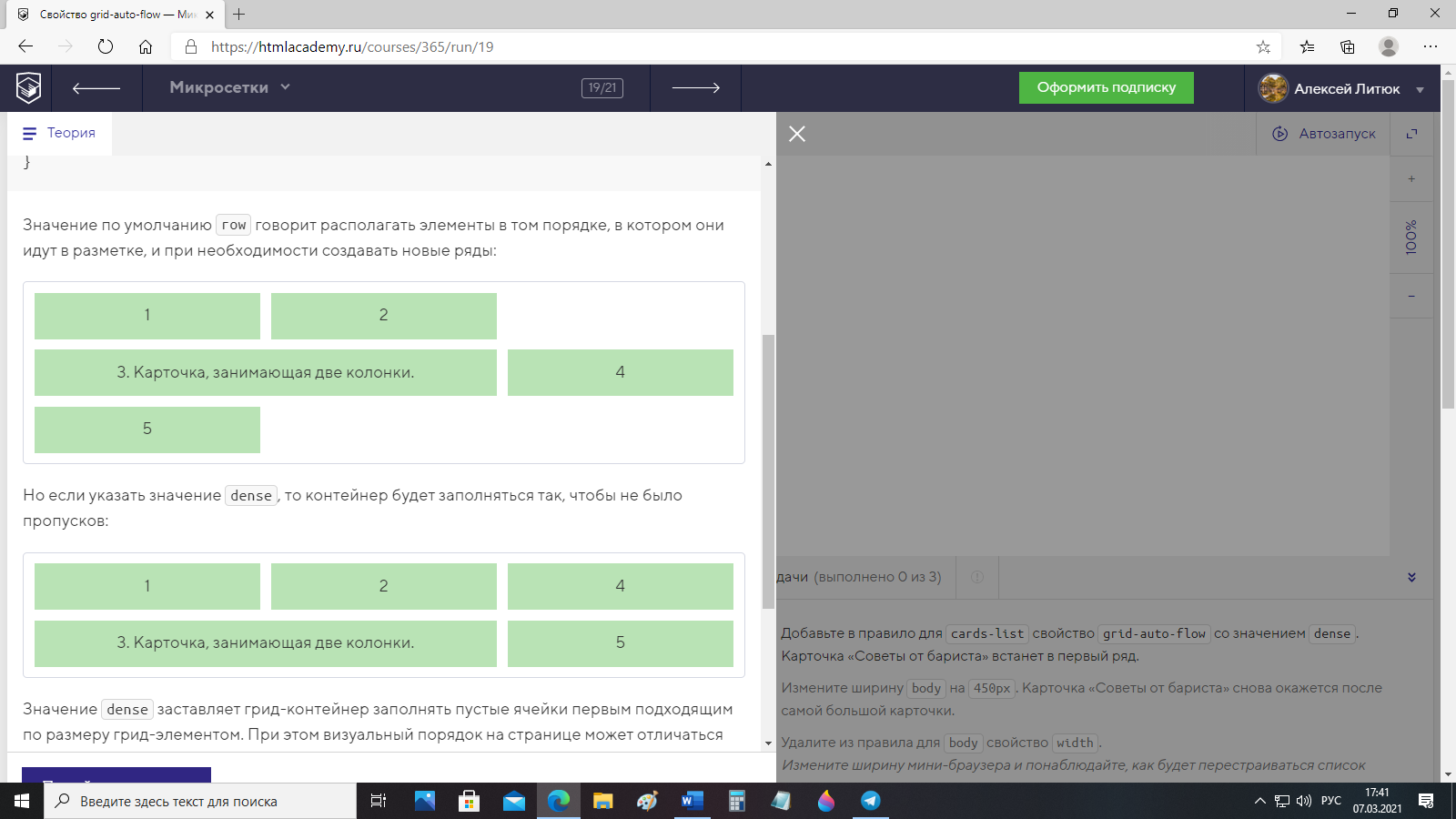
4

5

Значение dense заставляет грид-контейнер заполнять пустые ячейки первым подходящим по размеру грид-элементом. При этом визуальный порядок на странице может отличаться от порядка элементов в разметке. Если порядок элементов важен, лучше это значение не использовать.

В нашем списке порядок карточек не имеет значения, но желательно заполнять ряды полностью. Чтобы этого добиться, добавим списку карточек свойство grid-auto-flow со значением dense. Затем изменим ширину страницы и понаблюдаем, как элементы меняются местами.

Наконец, уберём у <body> фиксированную ширину. Будем изменять ширину окна мини-браузера и посмотрим, как меняются размеры колонок и перестраивается список карточек.



# Конспект «Микросетки». Раздел 2

## Список на флексах

Свойство justify-content со значением space-between заставляет первый и последний элемент прижиматься к границам контейнера. Но если в ряду всего два элемента, то свободного пространства между ними может оказаться слишком много. В этом случае лучше использовать margin.

Чтобы убрать лишний отступ у последнего элемента в ряду, используют псевдокласс :nth-child. Он позволяет выбрать дочерний элемент по его порядковому номеру:

// Выберет второй элемент с классом item

.item:nth-child(2) { ... }

// Выберет каждый второй элемент с классом item

.item:nth-child(2n) { ... }

Если не известно, какой элемент окажется в ряду последним, этот способ не сработает.

## repeat

Если все колонки в грид-контейнере должны быть одинаковой ширины, то удобно использовать значение-функцию repeat. В скобках после repeat указывают количество колонок и их ширину. Значения разделяют запятой:

grid-template-columns: repeat(количество колонок, ширина колонки);

## auto-fit

Если количество колонок зависит от ширины контейнера, используют специальное значение auto-fit. Его указывают в скобках после repeat вместо числа колонок:

grid-template-columns: repeat(auto-fit, ширина колонки);

**repeat (auto-fit, 100px)**

width: 200px;

width: 300px;

width: 400px;

## minmax

Чтобы ширина колонок изменялась пропорционально свободному пространству в контейнере, используют значение-функцию minmax.

Его указывают в repeat вместо фиксированной ширины колонок. В скобках после minmax задают минимальный и максимальный размеры колонок, они разделяются запятой:

repeat(auto-fit, minmax(минимальный размер, максимальный размер));

В minmax в качестве максимального значения часто используют единицу измерения fr. Она позволяет колонкам увеличивать ширину до тех пор, пока свободного пространства в контейнере не хватит на ещё одну колонку.

**minmax (100px, 1fr)**

width: 200px;

width: 299px;

width: 300px;

## Свойства grid-column и grid-row

Чтобы растянуть элемент на несколько колонок используют свойство grid-column. Число после ключевого слова span указывает число колонок, которые элемент должен занять:

.element {

grid-column: span 2;

}

Растянуть элемент на несколько рядов можно с помощью свойства grid-row. Ключевое слово span в нём означает не количество рядов, которые элемент должен занять:

.long-element {

grid-row: span 2;

}

Свойства grid-column и grid-row можно использовать одновременно.

## Свойство grid-auto-flow

Свойство grid-auto-flow управляет автозаполнением грид-контейнера.

.grid-container {

display: grid;

grid-auto-flow: row;

}

Значение по умолчанию row говорит располагать элементы в том порядке, в котором они идут в разметке, и при необходимости создавать новые ряды:

1

2

3. Карточка, занимающая две колонки.

4

5

Но если указать значение dense, то контейнер будет заполняться так, чтобы не было пропусков:

1

2

3. Карточка, занимающая две колонки.

4

5

Значение dense заставляет грид-контейнер заполнять пустые ячейки первым подходящим по размеру грид-элементом. При этом визуальный порядок на странице может отличаться от порядка элементов в разметке. Если порядок элементов важен, лучше это значение не использовать.

