## Построение сеток на гридах по макету Выполнен на 0% В Введение Теория 🖪 Базовая теория — Описание сетки — Явные и неявные координаты гридэлементов – Грид-области Именованные грид-линии — Грид-интервал — Размеры в гриде — Выравнивание в гриде В Углублённая теория Многослойность в гриде Повторы в гриде — Сокращённая запись Продвинутая теория Поведение автоматически размещаемых элементов Определение диапазона размеров Автоматическое заполнение грида 🖪 Дополнительный материал Инструменты для работы с гридами — Баги и ограничения 🕑 Методика построения сеток на гридах 🖸 Кейс 1, лёгкий уровень № Кейс 2, лёгкий уровень № Кейс 3, лёгкий уровень № Кейс 4, лёгкий уровень № Кейс 5, средний уровень

🖸 Кейс 6, средний уровень

🖸 Кейс 7, средний уровень

🖸 Кейс 8, средний уровень

🕠 Кейс 9, сложный уровень

Кейс 10, сложный уровень

Главная / Моё обучение / Построение сеток на гридах по макету / Теория / Продвинутая теория /

## Автоматическое заполнение грида

auto-fill и auto-fit — инструменты контроля над неопределённостью в гриде. Оба значения используются для того, чтобы сказать браузеру: «Умести как можно больше колонок с учётом указанной длины». Часто их используют для создания адаптивности в CSS Grid.

Разберём подробно где используются и как они работают.

## auto-fill и auto-fit

auto-fill и auto-fit используются в связке с repeat() (повторами). К примеру, мы используем grid-templatecolumns: repeat(5, 100px);, это создаёт фиксированное количество колонок фиксированного размера. Но если у нас динамический интерфейс, и количество элементов грида не определено наверняка (просто представьте, что это карточки!), то система становится неустойчивой.

Чтобы не фиксировать количество колонок, мы можем задать их размер и указать значение, которое повторит элемент столько раз, сколько поместится в грид-контейнер при учёте размера.

auto-fill добавит в строку столько колонок, сколько возможно при указанной ширине элемента и размере контейнера. Выглядеть запись будет так: grid-template-columns: repeat(auto-fill, 100px); . Если размер элемента указан в фиксированных единицах (в нашем примере — в пикселях), то рассматривать auto-fit нет смысла: он будет вести себя так же.

```
.grid {
 display: grid;
 grid-template-columns: repeat(auto-fill, 100px);
/* равнозначно */
.grid {
 display: grid;
 grid-template-columns: repeat(auto-fit, 100px);
```

Но в полной мере эти два значения раскрывают себя, если мы используем гибкую систему определения размеров блоков (функция minxmax () в CSS). В этом случае становится понятна разница между ними.

Например, мы задаём гриду, содержащему три грид-элемента, следующие параметры: grid-template-columns: repeat(auto-fill, minmax(100px, 1fr)); Это нужно читать так: мы определяем только размер колонок, они будут заполняться автоматически, размеры ячеек по горизонтали будут не меньше 100 пикселей, но не больше 1 фракции. При увеличении ширины грид-контейнера элементы будут растягиваться до тех пор, пока элементов размером 100 пикселей в контейнере не поместится на один больше. Если все грид-элементы поместились в одну строку, но контейнер растягивается дальше, то это свойство как бы будет добавлять «пустые элементы».

```
grid-template-columns: repeat(auto-fill, minmax(100px, 1fr));
                                       3
     auto-fill
```

(-2) auto-fill -2 -4 -5 -3 auto-fill

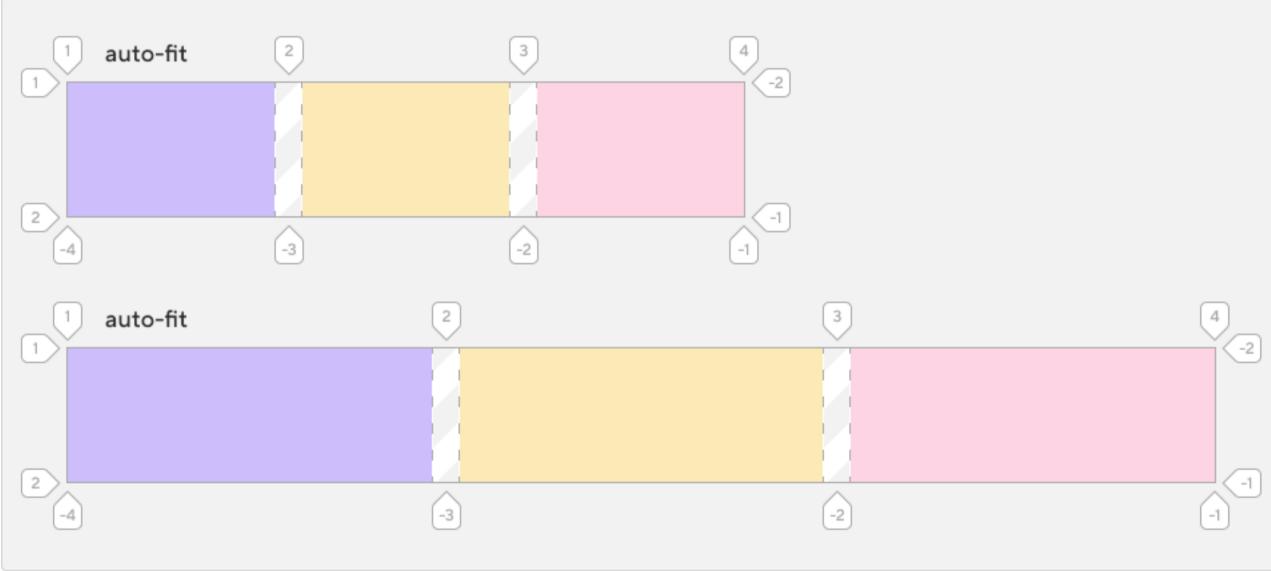
auto-fit ведёт себя немного иначе.

.grid {

display: grid;

Если есть только одна строка и есть место для дополнительных колонок, но нет элементов, которые можно вставить в эти колонки, значение auto-fit сделает ширину пустых колонок равную нулю. А если выставить максимальную ширину колонок на 1fr, то браузер разделит оставшееся место поровну между ними.

```
.grid {
 display: grid;
 grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));
```



auto-fit

Подытожим: разница между auto-fill и auto-fit заключается в том, как они обращаются с пространством, которое остаётся после того, как все элементы поместились в одну строку. auto-fill ведёт себя так, как будто элементов на самом деле бесконечное количество, он оставляет для этих несуществующих элементов место, a auto-fit делит это пространство поровну между элементами, которые есть.

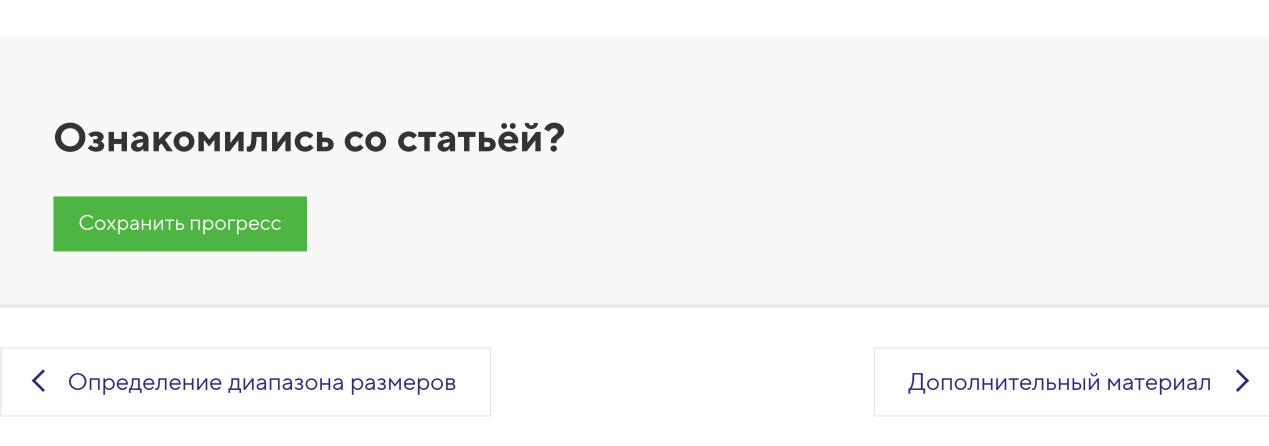
Эти особенности касаются расположения элементов, когда грид помещается в одну строку. Как только элементы занимают больше одной строки, различия между auto-fill и auto-fit снова исчезают.

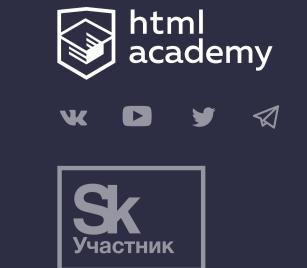
Важно: нельзя использовать значения [auto-fill]/[auto-fit] и указывать размеры колонок/строк вида [fr] или auto без min-max — это не будет работать. Невозможно подсчитать количество повторений для элементов, ширина которых занимает всё свободное пространство. Если всё же написать в свойстве grid-template-columns значение repeat(auto-fit, 1fr), то браузер покажет ошибку Invalid property value (Некорректное значение свойства).

```
.grid {
 display: grid;
 grid-template-columns: repeat(auto-fit, 1fr); /* Некорректное значение свойства */
```

Подробнее значение auto-fit мы разбираем в наших тренажёрах.

```
— Создание адаптивных сеток с помощью auto-fit
За счёт значения auto-fit легко создавать адаптивные сетки. При этом не нужно использовать
медиазапросы.
Например:
.grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));
Попробуйте уменьшить размеры грид-контейнера — и в какой-то момент все грид-элементы
выстраиваются в одну колонку.
        auto-fit
                                                                         auto-fit
                                                                   (2)
Адаптивная сетка с auto-fit
```





Практикум Тренажёры Подписка Для команд и компаний Учебник по РНР

Профессии

Услуги

Для учителей

Стать автором

React-разработчик

Работа наставником

HTML и CSS. Адаптивная вёрстка и автоматизация

HTML и CSS. Профессиональная вёрстка сайтов

JavaScript. Профессиональная разработка веб-интерфейсов JavaScript. Архитектура клиентских приложений React. Разработка сложных клиентских приложений

Фронтенд-разработчик РНР. Профессиональная веб-разработка PHP и Yii. Архитектура сложных веб-сервисов Фулстек-разработчик Бэкенд-разработчик

Шаблонизаторы HTML

Анатомия CSS-каскада

Алгоритмы и структуры данных

Курсы

Node.js. Разработка серверов приложений и API Анимация для фронтендеров Вёрстка email-рассылок Vue.js для опытных разработчиков

Регулярные выражения для фронтендеров

Блог С чего начать

Шпаргалки для разработчиков Отчеты о курсах

Об Академии О центре карьеры Остальное

Написать нам

Мероприятия

Форум

Информация

© ООО «Интерактивные обучающие технологии», 2013–2022