> Эта страница была переведена с английского языка силами сообщества. Вы тоже можете внести свой вклад, присоединившись к русскоязычному сообществу MDN Web Docs.

Основные понятия Grid Layout

Грид-раскладка (CSS Grid Layout) представляет собой двумерную систему сеток в CSS. Гриды подойдут и для верстки основных областей страницы, и небольших элементов пользовательского интерфейса. Эта статья даёт общее представление о грид-раскладке и новой терминологии, которая является частью спецификации CSS Grid Layout Level 1. Более подробно показанные возможности будут описаны описаны далее в руководстве.

Что такое грид?

Грид представляет собой пересекающийся набор горизонтальных и вертикальных линий, образующих колонки и строки. Элементы могут быть помещены в грид в пределах линий этих колонок и строк. Грид имеет следующие особенности:

Фиксированные и гибкие размеры полос

Вы можете создать грид с фиксированными размерами полос, например используя пиксели. Это установит грид на определенный пиксель, соответствующим желаемому макету. Вы также можете создать грид с гибкими размерами, используя проценты или новую единицу измерения — « `fr` », разработанную для этой цели.

Расположение элемента

Вы можете размещать элементы в заданном месте на гриде используя номера строк, имена или путём привязки к области грида. Грид также содержит алгоритм

управления размещением элементов, не имеющих явной позиции на гриде.

Создание дополнительных полос для хранения контента

Вы можете определить явную сетку с помощью грид-раскладки. Спецификация грид-раскладки достаточно гибкая, чтобы добавить при необходимости дополнительные строки и колонки. Также в нее включены такие возможности как, например, добавление «стольких колонок, сколько будет помещено в контейнер».

Управление выравниванием

Грид содержит механизм выравнивания, таким образом мы можем контролировать, как элементы выравниваются после размещения в области сетки и как выравнивается вся сетка.

Управление перекрывающимся контентом

В ячейку или область грида может быть помещено несколько элементов; эти элементы могут частично перекрывать друг друга. Такое наложение можно

Грид – это мощная спецификация, и в сочетании с другими частями CSS, такими как flexbox, поможет вам создать макеты, которые ранее невозможно было построить в CSS. Все начинается с создания сетки в вашем `грид-контейнере`.

Грид-контейнер

Мы создаём *`grid контейнер`*, объявляя на элементе `display: grid` или `display: inline-grid` . Как только мы это сделаем, *`все прямые потомки`* этого элемента станут *`элементами сетки`*.

В этом примере у меня есть контейнер div с классом-обёрткой и пятью дочерними элементами внутри.

HTML Play

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
```

```
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

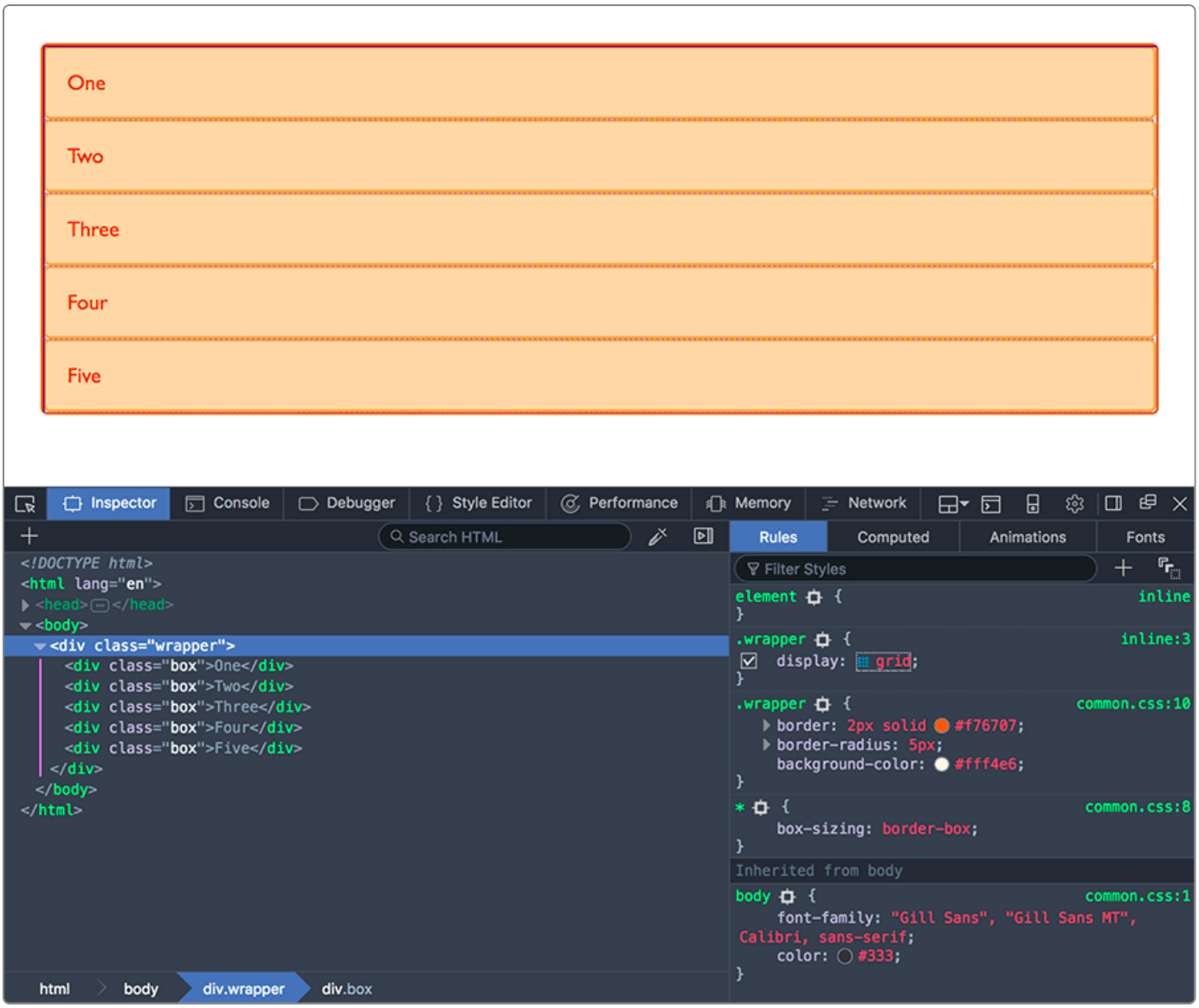
Я сделал `.wrapper` грид-контейнером.

CSS Play

```
.wrapper {
  display: grid;
}
```

Play

Все прямые потомки теперь являются грид-элементами. В браузере вы не увидите разницы с тем, как элементы отображались до помещения их в грид, поскольку грид сделан как одноколоночная сетка. На этой стадии вам, возможно, будет удобнее работать с инструментом «Грид-инспектор» (en-US) , который является частью «Инструментов веб-разработчика» Firefox (Firefox's Developer Tools). Если вы просмотрите этот пример в Firefox и проинспектируете грид, вы увидите маленький значок рядом с значением `grid` . Нажмите на него и сетка на этом элементе будет наложена в окне браузера.

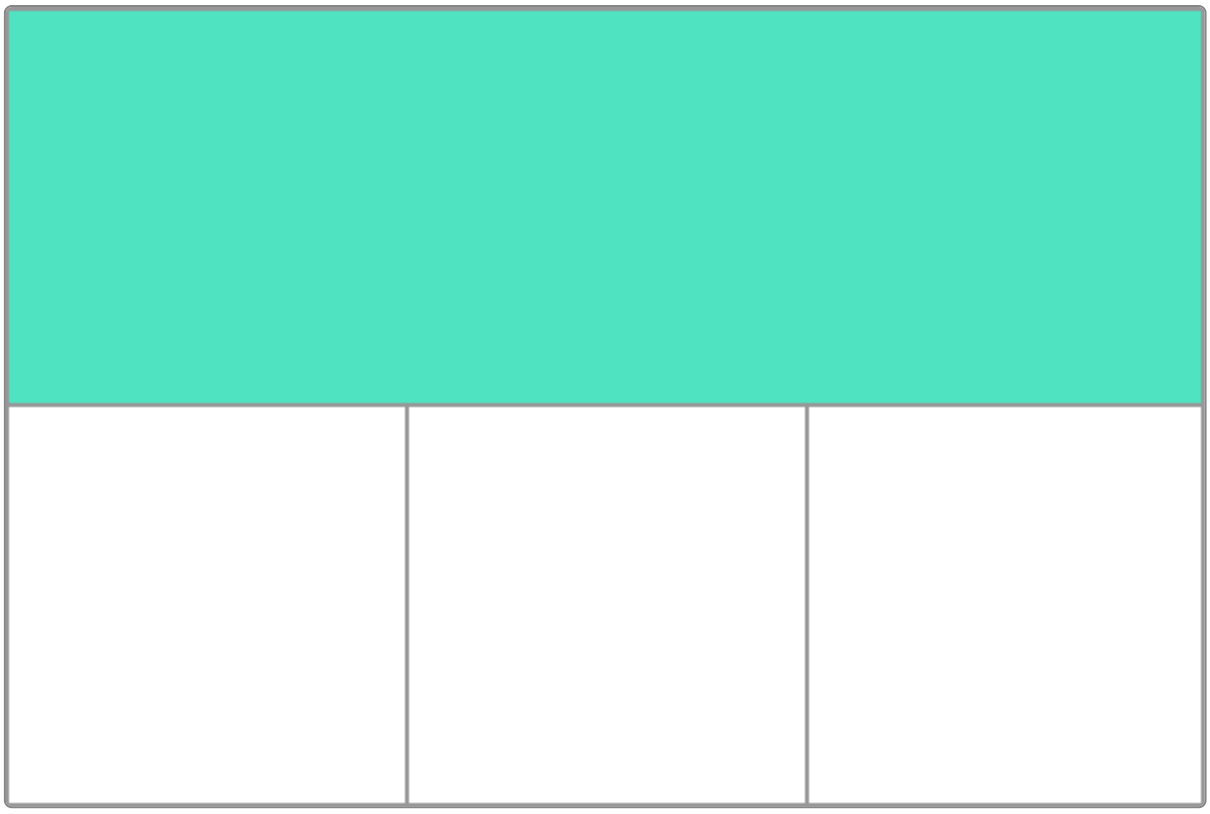


По мере вашего обучения и последующей работы с грид-раскладкой этот инструмент даст вам лучшее визуальное представление о том, что происходит с вашим гридом.

Если мы хотим, чтобы пример стал более похожим на сетку, нам нужно добавить полосы-колонки.

Грид-треки (грид-полосы)

Мы определяем ряды и колонки в нашей сетке при помощи свойств `grid-template-columns` и `grid-template-rows`. Это определения грид-треков (грид-полос). *Грид-трек* – это промежуток между любыми двумя линиями грида. На изображении ниже вы можете увидеть подсвеченный трек – это первый трек-строка в нашем гриде.



Базовый пример

Я могу дополнить пример выше, добавив свойство `grid-template-columns` и задав размеры треков-колонок.

Сейчас я создал грид с тремя колонками шириной по 200px. Каждый дочерний элемент будет располагаться в отдельной ячейке грида.

HTML Play

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

CSS Play

```
.wrapper {
  display: grid;
  grid-template-columns: 200px 200px 200px;
}
```

Play

Единица измерения «fr»

Размер треков может быть задан с помощью любой единицы длины. Спецификация также вводит дополнительную единицу длины, позволяющую создавать гибкие (flexible) грид-треки. Новая единица длины « `fr` » представляет собой долю (fraction) доступного пространства в грид-контейнере. Следующее определение грида создаст три одинаковых по ширине трека, расширяющихся и сужающихся в соответствии с доступным пространством.

HTML Play

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

CSS Play

```
.wrapper {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```

Play

Разные размеры

В следующем примере мы создаём грид с треком в `2fr` и двумя треками по `1fr`. Доступное пространство разбивается на четыре части. Две части занимает первый трек, и две части – два оставшихся.

HTML Play

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

CSS Play

```
.wrapper {
  display: grid;
  grid-template-columns: 2fr 1fr 1fr;
}
```

Play

Смешение гибких и фиксированных размеров

В последнем примере смешаем треки с абсолютными размерами и треки с размерами, определенными в `fr`. Первый трек – 500 пикселей, фиксированная ширина убирается из доступного пространства. Оставшееся пространство разбивается на три части и пропорционально разделяется между двумя гибкими треками.

HTML Play

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

CSS Play

```
.wrapper {
  display: grid;
  grid-template-columns: 500px 1fr 2fr;
}
```

Play

Задание треков с помощью нотации `repeat()`

В огромных гридах с большим количеством треков можно использовать нотацию `repeat()`, чтобы повторить всю структуру треков или её часть. Например, определение грида:

CSS

```
.wrapper {
  display: grid;
```

```
  grid-template-columns: 1fr 1fr 1fr;
}
```

Может быть также записано как:

CSS

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```

Repeat-нотация может быть использована для части структуры треков. В следующем примере я создал грид с начальным треком в 20 пикселей, затем повторил секцию с шестью треками по 1fr и завершил 20-пиксельным треком.

CSS

```
.wrapper {
  display: grid;
  grid-template-columns: 20px repeat(6, 1fr) 20px;
}
```

Repeat-нотация принимает список треков и использует его для создания повторяющегося шаблона треков. В следующем примере мой грид состоит из 10 треков: за треком в `1fr` следует трек в `2fr`. Этот шаблон будет повторен пять раз.

CSS

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(5, 1fr 2fr);
}
```

Явный и неявный грид

При создании грида в примере выше мы специально объявляли треки-колонки при помощи свойства `grid-template-columns`, но грид также самостоятельно создавал строки. Эти строки - часть «неявного» грида. В отличие от него, «явный» грид состоит из строк и колонок, заданных с помощью `grid-template-columns` или `grid-template-rows`.

Если вы размещаете что-нибудь вне рамок определённого грида или из-за количества контента требуется большее количество грид-треков, грид создаёт строки и колонки в виде неявного грида. Размер этих треков по умолчанию задаётся автоматически в зависимости от находящегося в них контента.

Вы также можете задать размер треков, создаваемых в виде неявного грида, с помощью свойств `grid-auto-rows` (en-US) и `grid-auto-columns` (en-US).

В примере ниже мы используем `grid-auto-rows`, чтобы убедиться, что треки, создаваемые в неявном гриде, были высотой 200 пикселей.

HTML Play

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

CSS Play

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 200px;
}
```

Play

Масштабирование треков и `minmax()`

При задании размеров явного грида или при определении размеров автоматически создаваемых колонок или строк нам может понадобиться задать трекам минимальный размер, но при этом быть уверенными, что они треки растянутся, чтобы вместить весь добавленный в них контент. Например, мне нужно, чтобы строки никогда не становились меньше 100 пикселей, но если контент занимает 300 пикселей в высоту, я бы хотел растянуть строку на эту высоту.

В гриде есть решение этой задачи – функция `minmax()`. В следующем примере я использую `minmax()` в качестве `grid-auto-rows` (en-US). То есть автоматически создаваемые строки будут как минимум 100 пикселей в высоту, а как максимум – примут значение `auto`. Использование `auto` означает следующее: высота строки «растягивается» до размера ячейки с самым высоким элементом контента.

CSS Play

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
```

```
  grid-auto-rows: minmax(100px, auto);
}
```

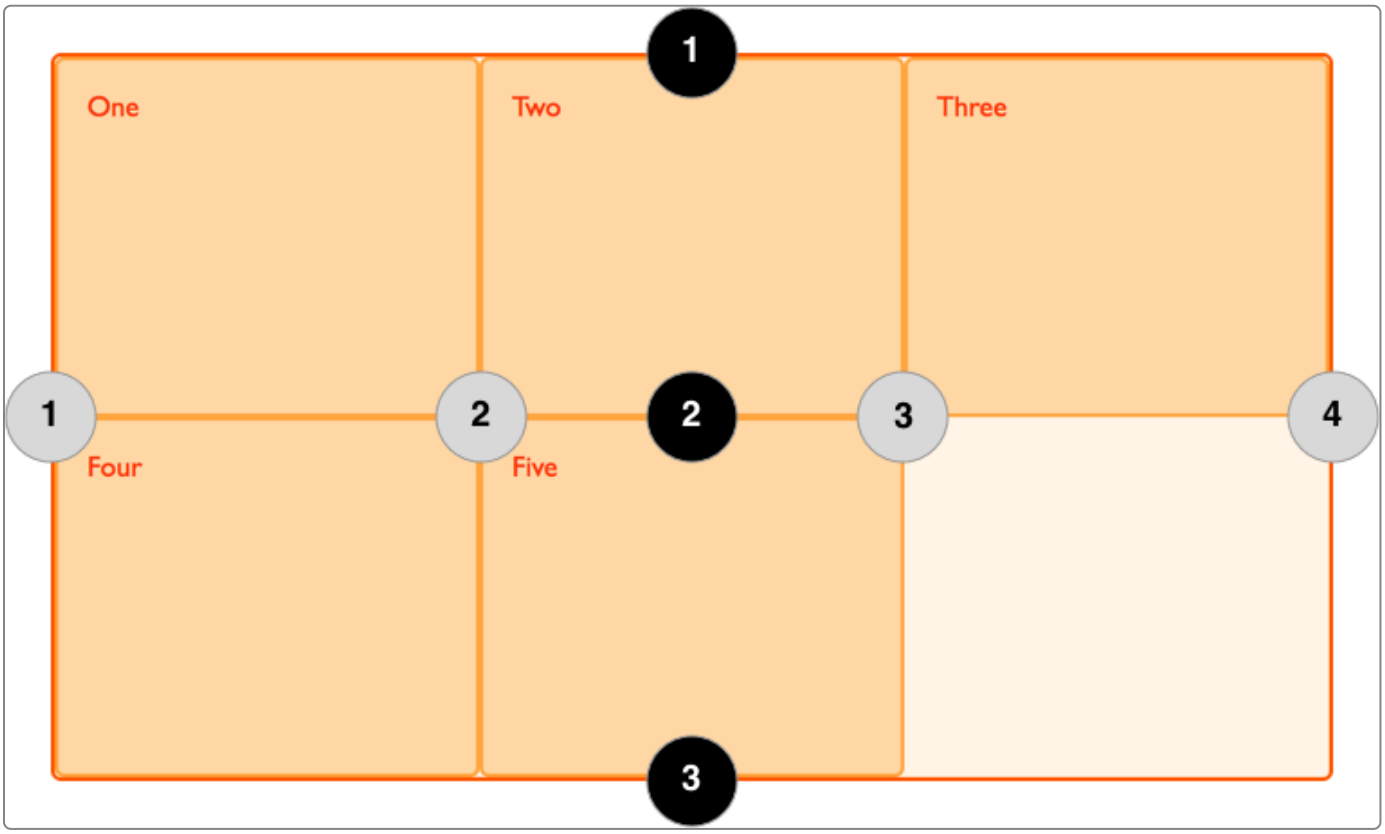
HTML Play

```
<div class="wrapper">
  <div>One</div>
  <div>
    Two
    <p>I have some more content in.</p>
    <p>This makes me taller than 100 pixels.</p>
  </div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

Play

Грид-линии

Нужно заметить, что когда мы определяем грид, мы определяем грид-треки, а не грид-линии. После этого грид обеспечивает нас пронумерованными линиями, для использования при размещении элементов. В нашем гриде с тремя колонками и двумя рядами у нас есть четыре линии колонок.



Линии пронумерованы в соответствии с режимом написания (writing mode) документа. В языках с написанием слева направо, линия 1 – самая левая линия в гриде. В языках с написанием справа налево линия 1 – самая правая линия в гриде. Линии также могут быть именованы, и мы рассмотрим, как это сделать, в одном из последующих руководств этой серии.

Размещение элементов по линиям

Детально размещение элементов с помощью линий мы рассмотрим в дальнейшем. Следующий пример демонстрирует простой способ. При размещении элемента мы задаем линию, а не на трек.

В следующем примере я разместил первые два элемента на нашем трёхколоночном гриде с помощью свойств `grid-column-start` (en-US), `grid-column-end` (en-US), `grid-row-start` и `grid-row-end` (en-US). Работая слева направо, первый элемент размещен начиная с колоночной линии 1 и занимает пространство до колоночной линии 4,

которая в нашем случае – самая правая линия грида. Наш элемент начинается со строчной линии 1 и заканчивается на строчной линии 3, таким образом занимая два строчных трека.

Второй элемент начинается с колоночной линии 1 и занимает один трек. Это поведение по умолчанию, поэтому нет необходимости задавать конечную линию. Элемент также занимает два строчных трека – со строчной линии 3 до строчной линии 5. Остальные элементы самостоятельно размещаются в свободном месте на гриде.

HTML Play

```
<div class="wrapper">
  <div class="box1">One</div>
  <div class="box2">Two</div>
  <div class="box3">Three</div>
  <div class="box4">Four</div>
  <div class="box5">Five</div>
</div>
```

CSS Play

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
}
.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
}
.box2 {
  grid-column-start: 1;
  grid-row-start: 3;
  grid-row-end: 5;
}
```

Play

> Примечание: Не забывайте, что вы можете использовать Grid Inspector (en-US) в Firefox Developer Tools, чтобы посмотреть, как элементы размещаются по линиям грида.

Сокращения при использовании размещения по линиям

Обычные значения, использованные выше, могут быть умещены в одну строку: для колонок – с использованием `grid-column` , для строк – с использованием `grid-row` (en-US). Следующий пример сделает такое же расположение, как и предыдущий, но с менее громоздким кодом CSS. значение до слэша («/») – это первая линия, значение после – последняя линия.

Можно опустить конечное значение, если область занимает только один трек.

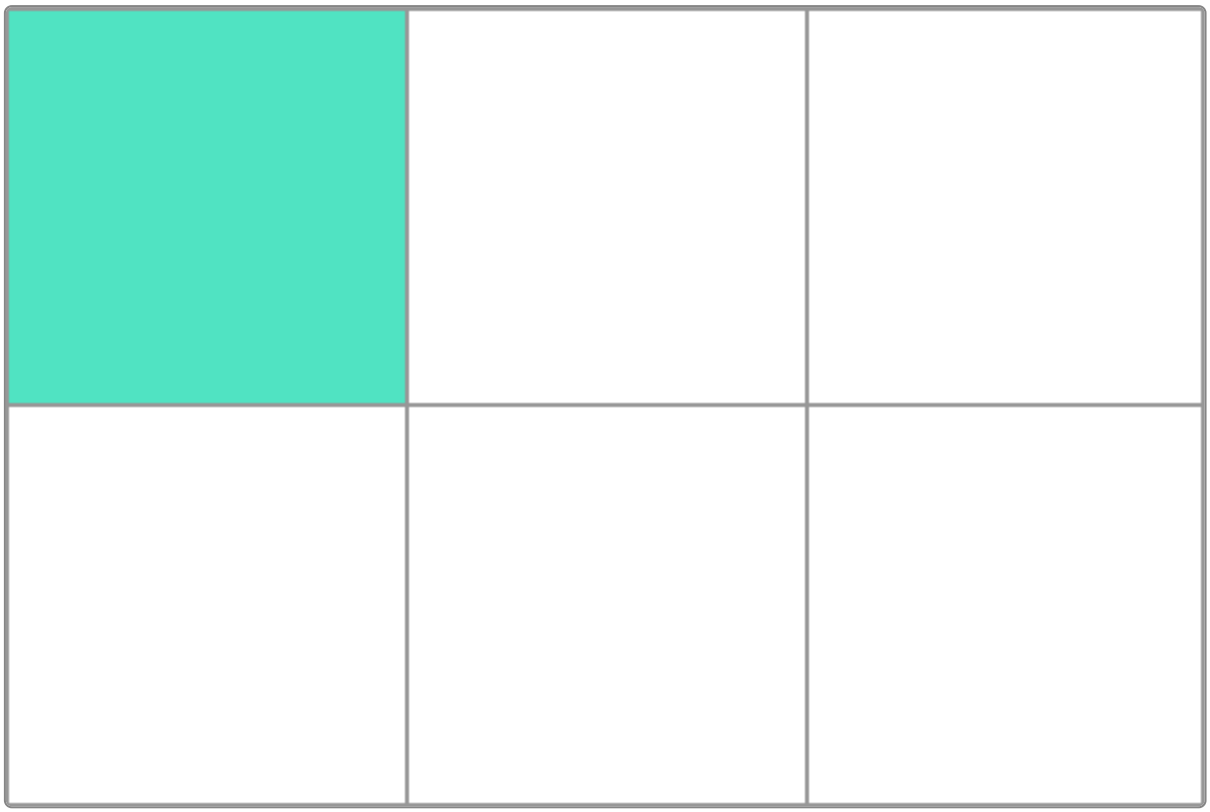
CSS

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
```

```
  grid-auto-rows: 100px;
}
.box1 {
  grid-column: 1 / 4;
  grid-row: 1 / 3;
}
.box2 {
  grid-column: 1;
  grid-row: 3 / 5;
}
```

Грид-ячейки

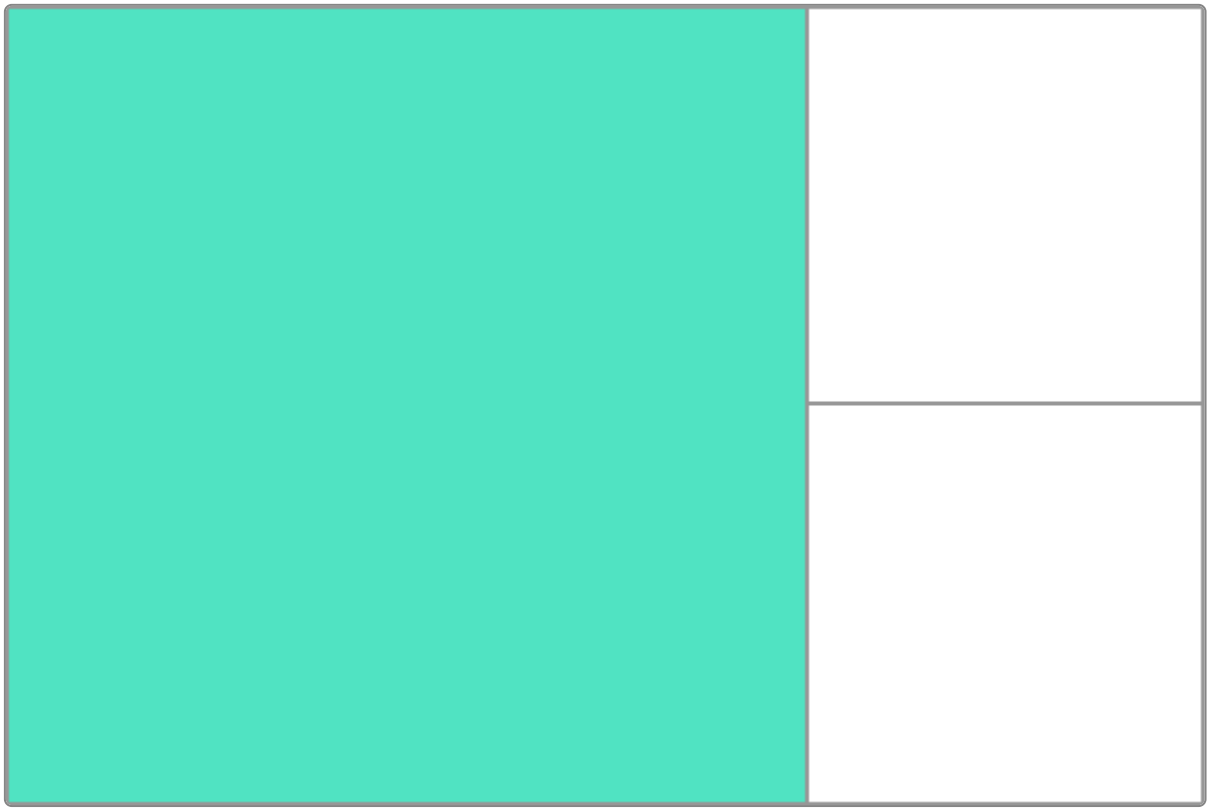
Грид-ячейка – наименьшая часть на гриде. Концептуально она похожа на ячейку таблицы. Как мы видели в предыдущих примерах, едва грид определён для родительского элемента, дочерние элементы автоматически размещаются в каждой ячейке заданного грида. На рисунке ниже я выделил первую ячейку грида.



Грид-области

Элементы могут занимать одну или несколько ячеек внутри строки или колонки, таким образом, создаётся *грид-область*. Грид-области должны быть перпендикулярными – невозможно создать область, например, в форме буквы «L».

Выделенная грид-область на рисунке ниже занимает два строчных трека и два колоночных.



Промежутки

Промежутки (gutters), или *дорожки (alleys)*, между грид-ячейками могут быть созданы с помощью свойств `grid-column-gap` (en-US) и `grid-row-gap` (en-US), или с помощью сокращённого свойства `grid-gap` (en-US). В примере ниже я создаю 10-пиксельный промежуток между колонками и промежуток в `1em` между строками.

CSS Play

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-column-gap: 10px;
  grid-row-gap: 1em;
}
```

> Примечание: Когда грид был добавлен в браузеры, у свойств `column-gap`, `row-gap` и `gap` был префикс «grid-»: «grid-column-gap», «grid-row-gap» «grid-gap» соответственно.Сейчас все браузеры поддерживают значения без префикса,

> однако версии с префиксом будут поддерживаться как псевдонимы, что делает их использование безопасным.

HTML Play

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

Play

Пространство, занимаемое промежутками, будет учитываться как пространство, расположенное перед гибкими треками, заданными с помощью `fr`, и приписанное к длине этих треков, и промежутки служат для задания размеров как обычные грид-треки, однако в промежутках нельзя ничего размещать. С точки зрения размещения по линиям, промежутки действуют в роли толстой линии.

Вкладывание гридов

Грид-элемент может быть и грид-контейнером. В следующем примере у меня есть созданный ранее трёхколоночный грид, с двумя нашими размещенными элементами. В данном случае у первого элемента есть несколько подэлементов. Поскольку эти подэлементы не являются прямыми потомками грида, они не участвуют в структуре грида и отображаются в нормальном потоке документа.

Вкладывание без подгридов

Если я задам для `box1` значение `display: grid`, я могу установить для него структуру треков, и он тоже станет гридом. Элементы будут размещены на этом новом гриде.

CSS Play

```
.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```

HTML Play

```
<div class="wrapper">
  <div class="box box1">
    <div class="nested">a</div>
    <div class="nested">b</div>
    <div class="nested">c</div>
  </div>
  <div class="box box2">Two</div>
  <div class="box box3">Three</div>
```

```
    <div class="box box4">Four</div>
    <div class="box box5">Five</div>
  </div>
```

Play

В данном случае вложенный грид не связан с родительским. Как вы можете видеть, он не наследует значение свойства `gap` родителя и линии вложенного грида не выравниваются по линиям родительского грида.

Подгрид (Subgrid)

В рабочих черновиках спецификации гридов второго уровня есть функциональность, называемая *подгридом*, которая позволит нам создавать вложенные гриды, использующие структуру треков родительского грида.

> **Примечание:** Эта функция поддерживается в браузере Firefox 71, который в настоящее время является единственным браузером, обеспечивающим выполнение подгрида.

Согласно этой спецификации, мы можем отредактировать приведенный выше пример вложенного грида, изменив `grid-template-columns: repeat(3, 1fr)` на `grid-template-`

`columns: subgrid`. Вложенный грид будет использовать родительскую структуру треков для размещения элементов.

CSS

```
.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
  display: grid;
  grid-template-columns: subgrid;
}
```

Размещение элементов с помощью z-index

Несколько грид-элементов могут занимать одну и ту же ячейку и поэтому мы можем использовать свойство `z-index`, чтобы контролировать порядок наложения перекрывающихся элементов.

Наложение без z-index

Если мы вернемся к нашему примеру с элементами, размещёнными по номерам линий, мы можем изменить его, чтобы два элемента перекрывались:

HTML Play

```
<div class="wrapper">
  <div class="box box1">One</div>
  <div class="box box2">Two</div>
  <div class="box box3">Three</div>
  <div class="box box4">Four</div>
  <div class="box box5">Five</div>
</div>
```

CSS Play

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
}
.box1 {
```

```
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
}
.box2 {
  grid-column-start: 1;
  grid-row-start: 2;
  grid-row-end: 4;
}
```

Play

Элемент `box2` теперь перекрывает `box1`, он отображается сверху, поскольку в исходном коде находится ниже.

Управление порядком отображения

Мы можем управлять порядком наложения элементов с помощью свойства `z-index` – точно так же, как позиционированными элементами. Если мы зададим для `box2` значение `z-index`, меньшее, чем у `box1`, он отобразится под элементом `box1`.

CSS Play

```css
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
}
.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
  z-index: 2;
}
.box2 {
  grid-column-start: 1;
  grid-row-start: 2;
  grid-row-end: 4;
  z-index: 1;
}
```

Play

Что дальше?

В данной статье мы очень кратко рассмотрели спецификацию Grid Layout. Поиграйте с примерами кода и переходите к следующей части нашего руководства, где мы начнём углубляться в детали CSS Grid Layout.

This page was last modified on 7 авг. 2023 г. by MDN contributors.