Grid.

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 контролировать с помощью свойства z-index.

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

Создадим 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 есть контейнер div с классом-обёрткой и пятью дочерними элементами внутри.

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

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

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. Чтобы он стал более похожим на сетку, нужно добавить полосы-колонки.

Грид-треки (грид-полосы)

Ряды и колонки в нашей сетке можно определить 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.

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

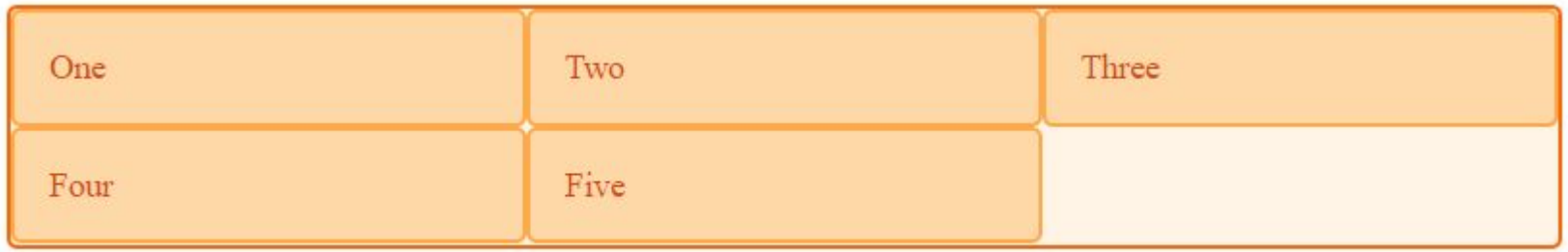
```css
.wrapper {
  display: grid;
  grid-template-columns: 200px 200px 200px;
}
```

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

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

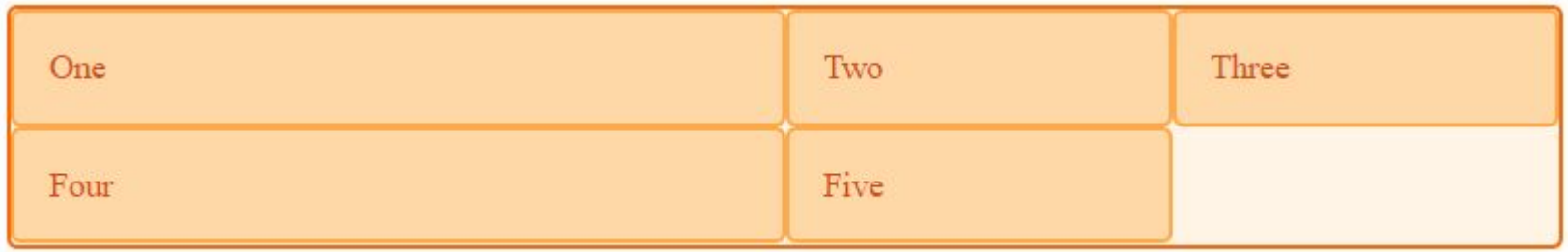


Разные размеры

В следующем примере создаlbv 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
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

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



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.

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

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

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:

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

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

Repeat-нотация может быть использована для части структуры треков. В следующем примере создан грид с начальным треком в 20 пикселей, затем повторяется секция с шестью треками по 1fr и завершается 20-пиксельным тр

```
.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 грид состоит из 10 треков: за треком в 1fr следует трек в 2fr. Этот шаблон будет повторен пять раз.

```
.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.

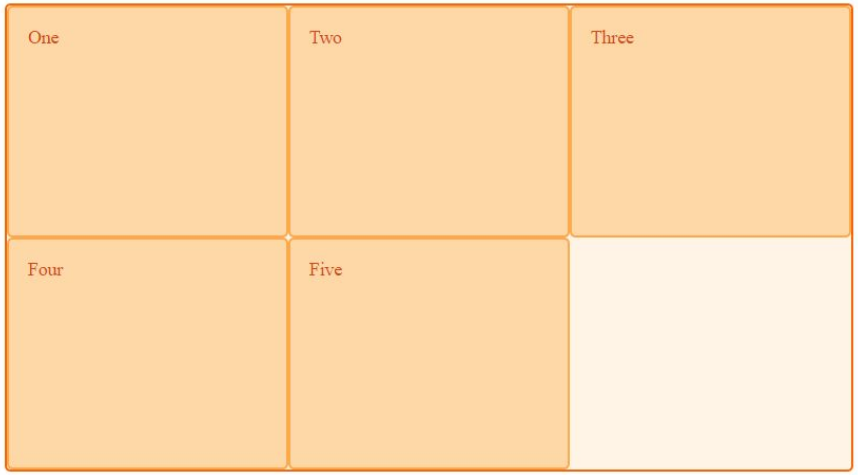
Если вы размещаете что-нибудь вне рамок определённого грида или из-за количества контента требуется большее количество грид-треков, грид создаёт строки и колонки в виде неявного грида. Размер этих треков по умолчанию задаётся автоматически в зависимости от находящегося в них контента.

Вы также можете задать размер треков, создаваемых в виде неявного грида, с помощью свойств grid-auto-rows и grid-auto-columns

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

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

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



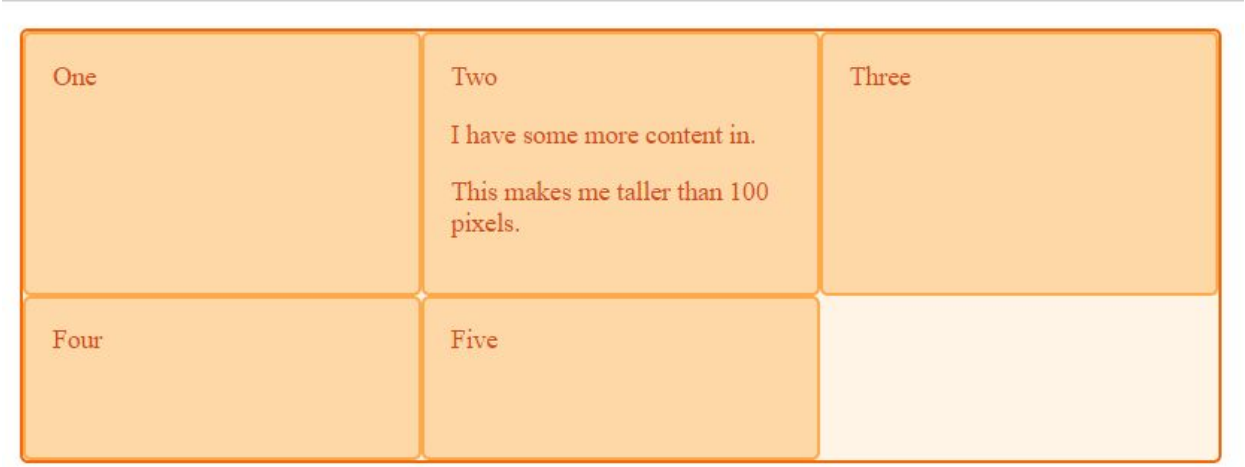
Масштабирование треков и minmax()

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 используется minmax() в качестве grid-auto-rows 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.

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

```
<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>
```



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.

Размещение элементов по линиям

Следующий пример демонстрирует простой способ. При размещении элемента мы задаем линию, а не на трек.

В следующем примере первые два элемента размещаются на трёхколоночном гриде с помощью свойств grid-column-start, grid-column-end, grid-row-start и grid-row-end. 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

```html
<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
.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;
}
```

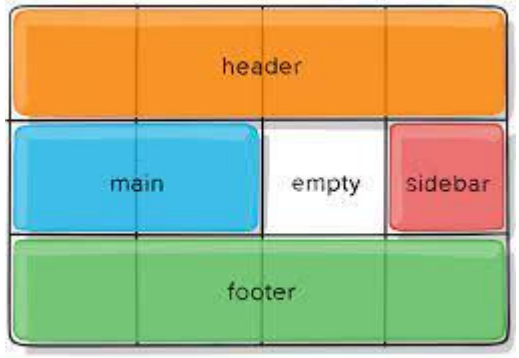
Сокращения при использовании размещения по линиям
Обычные значения, использованные выше, могут быть умещены в одну строку: для колонок – с использованием grid-column, для строк – с использованием grid-row. Следующий пример сделает такое же расположение, как и предыдущий, но с менее громоздким кодом CSS.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

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

```
.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;
}
```

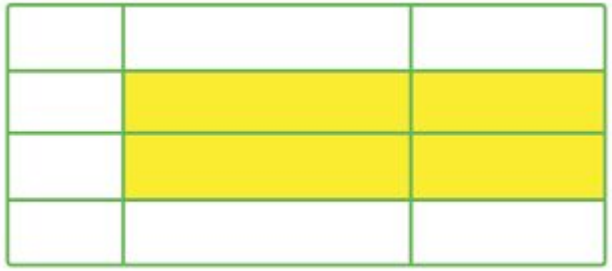
Грид-ячейки

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.



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».

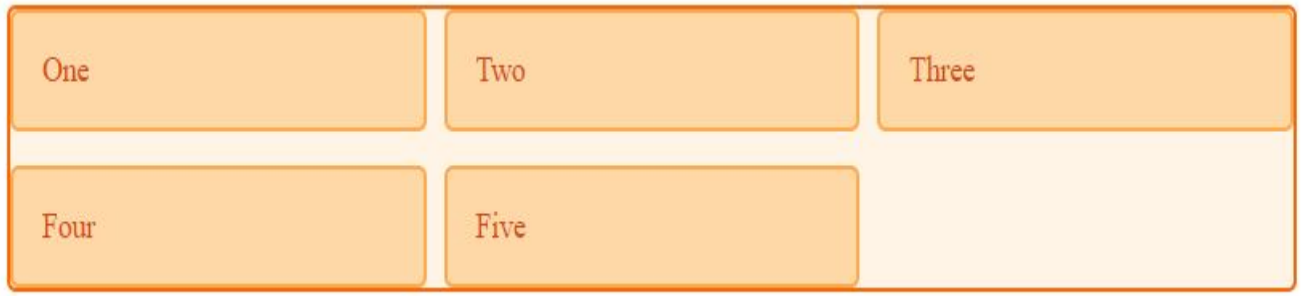


Промежутки

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

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

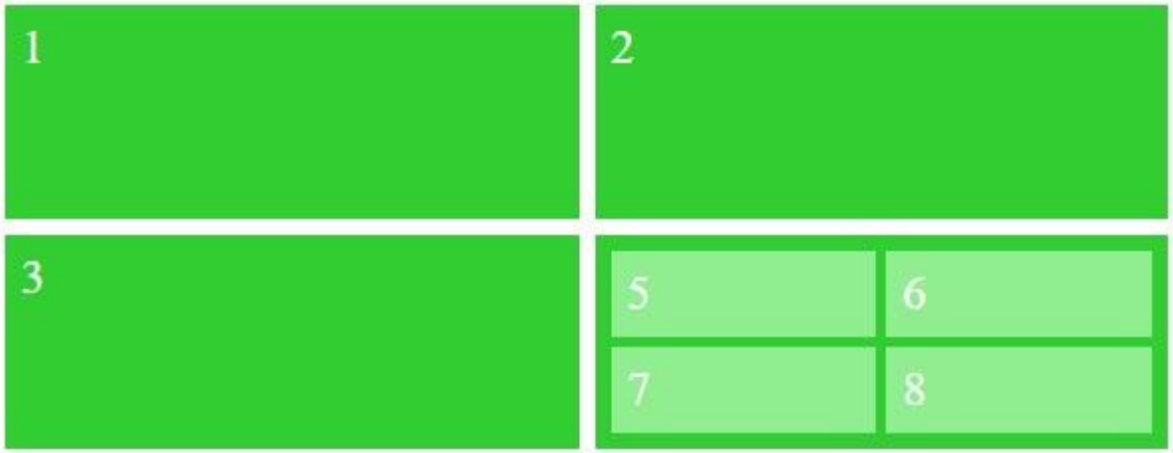
```css
.wrapper {
   display: grid;
   grid-template-columns: repeat(3, 1fr);
   grid-column-gap: 10px;
   grid-row-gap: 1em;
}
```



Вкладывание гридов

Грид-элемент может быть и грид-контейнером. В следующем примере есть созданный ранее трёхколоночный грид, с двумя нашими размещенными элементами. В данном случае у четвертого элемента есть несколько подэлементов.

Чтобы создать такой вложенный грид, всё, что вам нужно сделать — это применить display: grid (или display: inline-grid) к грид элементу и он сам станет гридом.



Grid - https://fls.guru/grid.html

Макет для верстки – https://www.figma.com/file/9B5e4qgZK8n6EGYaCAFyXo/Tours-%26-Adventures-Landing-Page-(Community)?node-id=2%3A73