Суть

Родитель становится контейнером сеткой. Прямые потомки помещаются в ячейки.

Размер сетки описывается в родителе.

.grid-container {

display: grid;

grid-template-rows: 5em 200px 200px; /\* 3 строки \*/

grid-template-columns: 200px 5em 50%; /\* 3 столбца \*/

}

Потомки привязываются не к столбцам и строкам как в таблицах, а к номерам линии сетки. Левый край – правый край, верхний край – нижний карай.

.item-a {

grid-column-start: 2;

grid-column-end: 4;

grid-row-start: 1

grid-row-end: 3

}

Выравнивание

Очень важно, что потомки не целиком заполняют ячейки. Если потомок не 100% ширины, то он может быть меньше ячейки, а если больше, влезает из ячейки, поэтому у всех потомков, если они должны целиком заполнять ячейку должен стоять width: 100%; Это очень важно, когда необходимо разместить в потомке несколько элементов, а они упорно не выравнисаются.

Остальное по необходимости.

Разные способы привязки

Это сокращённая запись. Ещё можно указать сколько ячеек вместо конечной линии.

grid-column: 3 / span 2;

Можно по имени который описывается в grid-template-areas у родителя. Это красиво, но заморочено.

Выравнивание

Выравнивание со стороны родителя звучит как justify-items / align-items – выравнивать элементы. Оно назначает выравнивание сразу всем потомкам по горизонтали / вертикали.

Выравнивание со стороны потомка звучит как justify-self / align-self – выровнять самого себя по горизонтали / вертикали.

Разрывы

grid-gap: 10px 15px;

2. Контейнер-сетка

Для создания макета на основе сетки необходимо определить контейнер-сетку. Контейнер-сетка (grid container) — это блок, который устанавливает контекст форматирования по типу сетки, то есть создает область с сеткой, а дочерние элементы располагаются в соответствии с правилами компоновки сетки, а не блочной компоновки. Когда вы определяете контейнер сетки с помощью display: grid или display: inline-grid, вы создаете новый контекст форматирования для содержимого этого контейнера, который влияет только на дочерние элементы сетки.

Контейнер-сетка бывает двух видов: обычный display: grid и встроенный display: inline-grid. Первый генерирует grid-контейнер уровня блока, второй — grid-контейнер уровня строки. Контейнеры-сетки не являются блочными контейнерами, поэтому некоторые CSS-свойства не работают в контексте макета сетки:

* float и clear игнорируются элементами сетки (но не самим контейнером-сеткой).
* vertical-align не влияет на элементы сетки.
* Псевдоэлементы ::first-line и ::first-letter не применяются к контейнеру-сетке и его потомкам.
* Если контейнер-сетка является контейнером уровня строки display: inline-grid и для него заданы обтекание или абсолютное позиционирование, то вычисляемое значение свойства display будет grid.

Определение сетки

Когда вы создаете контейнер-сетку, сетка по умолчанию имеет один столбец и одну строку, которые занимают полный размер контейнера. Для разделения контейнера-сетки на столбцы и строки используются свойства grid-template-columns, grid-template-rows и grid-template-areas. С помощью этих свойств можно определить сетку явно. (template – это шаблон).

Окончательная сетка может оказаться больше из-за элементов сетки, размещенных вне явной сетки; в этом случае будут созданы неявные дорожки, размер этих неявных дорожек будет определяться свойствами grid-auto-rows и grid-auto-columns.

Свойства grid и grid-template — это сокращенные обозначения, которые можно использовать для одновременной установки всех трех явных свойств сетки grid-template-columns, grid-template-rows и grid-template-areas. grid сбрасывает свойства, управляющие неявной сеткой, тогда как свойство grid-template оставляет их без изменений.

Пример:

.grid-container {

display: grid;

grid-template-rows: 5em 200px 200px; /\* 3 строки \*/

grid-template-columns: 200px 5em 50%; /\* 3 столбца \*/

}

Строки и столбцы

Количество строк / столбцов определяется с помощью свойств grid-template-rows и grid-template-columns. Свойства не наследуются. Пример см. выше. Далее перечислены все варианты описания размера и количества.

Относительные, абсолютные единицы и процентные значения

Размеры сетки можно задавать с помощью положительных значений, используя относительные единицы длины — например, em, vh, vw; абсолютные единицы длины — px; и проценты %. Размеры в % вычисляются от ширины или высоты контейнера-сетки. Пример:

.grid-container {

display: grid;

grid-template-rows: 5em 200px 200px; /\* 3 строки \*/

grid-template-columns: 200px 5em 50%; /\* 3 столбца \*/

}

Гибкие размеры: единица измерения fr

fr — единица длины, которая позволяет создавать гибкие дорожки. Не является единицей измерения в обычном ее понимании, поэтому не может быть представлена или объединена с другими типами единиц в выражениях calc(). Общий размер фиксированных строк или столбцов вычитается из доступного пространства контейнера-сетки. Оставшееся пространство делится между строками и столбцами с гибкими размерами пропорционально их коэффициенту.

Например:

.grid-container {

display: grid;

grid-template-columns: 1fr 1fr 1fr 1fr; /\* эквивалентно grid-template-columns: 25% 25% 25% 25%; \*/

}

.grid-container {

display: grid;

grid-template-columns: 1fr 2fr 1fr; /\* эквивалентно grid-template-columns: 25% 50% 25%; \*/

}

Если сумма всех гибких размеров дорожек меньше 1, они будут занимать только соответствующую долю оставшегося пространства, а не расширяться, чтобы заполнить его полностью.

Если доступное пространство бесконечно (то есть, ширина или высота контейнера-сетки не заданы), дорожки сетки гибкого размера масштабируются по своему содержимому, сохраняя при этом их соответствующие пропорции.

Пока достаточно. Далее <https://html5book.ru/css-grid#grid-container>

6. Выравнивание элементов сетки и промежутки между элементами

6.4. Промежутки между элементами сетки

Свойства row-gap и column-gap (и их сокращенная запись gap), если они указаны в контейнере сетки, определяют промежутки между строками и столбцами сетки. При определении размера дорожки каждый промежуток рассматривается как дополнительная пустая дорожка указанного размера. Дополнительный промежуток также может быть добавлен между дорожками за счет свойств justify-content и align-content.

Промежутки добавляются только между двумя дорожками сетки, то есть они не добавляются перед первой и после последней дорожки.

Значения:

normal —Вычисляется как 0px. Значение по умолчанию.

длина или % — Процентное значение вычисляется относительно размеров области сетки. Отрицательные значения не используются.

row-gap: 1.5em;

column-gap: 10px;

gap: 1%;

GRID

Элемент, к которому применяется display: grid. Это прямой родитель для всех элементов сетки. Каждый дочерний элемент — это ячейка таблицы. Родитель строит номера. Дети их заселяют.

Пример самого элементарного грида:

Свойства для родительского элемента (Контейнера сетки)

display определяет элемент как контейнер и устанавливает новый контекст форматирования сетки для его содержимого. Обратите внимание на то, что column, float, clear и vertical-align не дадут никакого эффекта на контейнере.

* grid - формирует сетку как блок;
* inline-grid - формирует сетку как инлайновый блок ;
* subgrid - если ваш контейнер это ещё и элемент (вложенная сетка), то вы можете использовать это свойство для обозначения того, чтобы размеры строк/колонок были взяты из родительского элемента, а не определяли собственный;

Отличие grid и inline-grid:

<style>

body {font: 12px Verdana, sans-serif;}

.grid, .inline-grid {

border: 3px dashed #333;

grid-gap: 3px 5px;

margin: 5px 0;

}

.grid {

display: grid;

}

.inline-grid {

display: inline-grid;

}

.grid-item {

border: 3px solid #F2E205;

border-radius: 5px;

padding: 10px;

background: rgba(242,226,5,.8);

}

</style>

<div class="grid">

<div class="grid-item">Это display: grid</div>

</div>

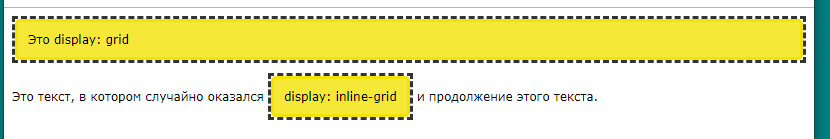
Это текст, в котором случайно оказался

<div class="inline-grid">

<div class="grid-item">display: inline-grid</div>

</div>

и продолжение этого текста.



subgrid используется если нужно привязаться к линиям сетки родителя. Примеры можно посмотреть в <https://codepen.io/rachelandrew/pen/mgpqBv> и <https://codepen.io/rachelandrew/pen/YMYEVO?editors=1100> и <https://codepen.io/rachelandrew/pen/axLzYv?editors=1100> разные способы привязки.

grid-template-columns grid-tempate-rows

Определяет колонки и строки сетки с помощью списка значений. Когда вы оставляете пустое пространство между значениями треков, линиям сетки автоматически присваиваются числовые имена. При этом дочерние элементы описываются с помощью номеров столбцов и строк. Или по именам, если определены. Всякие заморочки с двойными именами и т.д. Разберёмся по ходу дела.

Фреймы вычисляются после выделении всех фиксированных значений.

grid-template-areas

Определяет шаблон сетки ссылаясь на имена областей, которые заданы с помощью свойства grid-area. Повторение названия области приводит к тому, что содержимое охватывает эти ячейки. Точка означает пустую ячейку.

Пример:

.item-a {grid-area: header;}

.item-b {grid-area: main;}

.item-c {grid-area: sidebar;}

.item-d {grid-area: footer;}

.container {

grid-template-columns: 50px 50px 50px 50px;

grid-template-rows: auto;

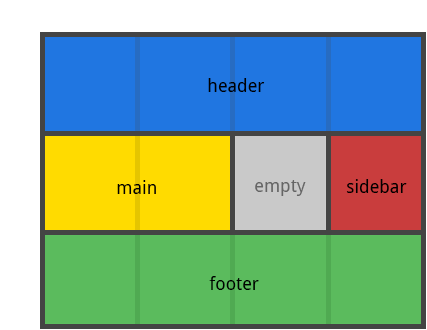
grid-template-areas:

"header header header header"

"main main . sidebar"

"footer footer footer footer";

}



grid-template

Сокращение для grid-template-rows, grid-template-columns, и grid-template-areas.

Вот пример:

.container {

grid-template:

[row1-start] 25px "header header header" [row1-end]

[row2-start] "footer footer footer" 25px [row2-end]

/ auto 50px auto;

}

Что эквивалентно следующему:

.container {

grid-template-rows: [row1-start] 25px [row1-end row2-start] 25px [row2-end];

grid-template-columns: auto 50px auto;

grid-template-areas:

"header header header"

"footer footer footer";

}

grid-column-gap grid-row-gap

Определяет размер ширины линий. Отступы создаются только между колонками и строками, но не для внешних краев сетки.

grid-gap

Сокращение для grid-row-gap и grid-column-gap.

justify-items

Выравнивает содержимое внутри сетки вдоль оси строки (в отличии от align-items который выравнивает элементы вдоль оси столбца). Это значение применяется ко всем элементам сетки внутри контейнера.

* start - выравнивает содержимое по левой стороне области;
* end - выравнивает содержимое по правой стороне области;
* center - выравнивает содержимое по центру области;
* stretch - заполняет всю ширину области (по умолчанию);

align-items

Выравнивает содержимое внутри сетки вдоль оси столбца. Смотреть выше.

justify-content

Выравнивает саму сетку внутри контейнера

* start - выравнивает сетку по левой стороне контейнера;
* end - выравнивает сетку по правой стороне контейнера;
* center - выравнивает сетку по центру контейнера;
* stretch - масштабирует элементы чтобы сетка могла заполнить всю ширину контейнера;
* space-around - одинаковое пространство между элементами, и полуразмерные отступы по краям;
* space-between - одинаковое пространство между элементами, без отступов по краям;
* space-evenly - одинаковое пространство между элементами, и полноразмерные отступы по краям;

align-content

Свойства для дочерних элементов (Grid элементы)

grid-area - даёт название элементу чтобы можно было ссылаться на него с помощью шаблона созданного через grid-template-areas свойство.

.item-d {

grid-area: header

}

min-content и max-content

Работают для нескольких слов

align-self

Это свойство помогает позиционировать содержимое по вертикали.

justify-self

Это свойство помогает позиционировать содержимое по горизонтали.

Сайт <https://tpverstak.ru/grid/>