Css Grid

The CSS Grid layout module

Введение

CSS Grid Layout - это двумерная система позиционирования, основанная на сетке. Помимо grids в нашем арсенале есть такие инструменты как таблицы, обтекания (floats), позиционирование и инлайновые блоки (inline-block), каждый из которых имеет свои ограничения.. Flexbox помог добиться большей гибкости, но он предназначен для более простых одномерных макетов, а не для сложных двумерных (на самом деле Flexbox и Grid очень хорошо работают вместе). CSS Grid'ы - это первый модуль созданный **специально** для решения проблем позиционирования, которые до сих пор мы решали с помощью хаков при создании сайтов

Терминология

1. Grid-container: Элемент, со свойством display: grid.
2. Grid-item: Прямые потомки Grid-container.
3. Grid line: Разделительные линии, составляющие структуру для сетки. Они могут быть вертикальными ("линии колонок") или горизонтальными ("линии строк") и располагаться по обе стороны от строки или столбца.
4. Grid track: Пространство между двумя соседними линиями. Их можно визуализировать как столбцы или строки сетки.

Терминология (продолжение)

5. Grid gap: Расстояние между grid tracks.

6. Grid cell: Пространство между линиями двух соседних строк и двух соседних столбцов. Это отдельная "единица измерения" сетки.

7. Grid area: Общее пространство окруженное четырьмя линиями. Область может состоять из любого количества ячеек.

Базовые свойства Grid container-а

Чтобы сделать контейнер областью сетки, нужно применить свойство display: grid, после чего у контейнера и у его прямых потомков (grid-items) появляется ряд новых свойств.

Базовыми свойствами для формирования сетки являются:

- grid-template-columns

- grid-template-rows

Базовые свойства Grid container-а

Значения вышеуказанных свойств - размеры grid-track-ов (строк или колонок) по желаемому количеству последних.

Например:

```
.container {
    display: grid;
    grid-template-columns: 200px 500px;
    grid-template-rows: 200px auto 400px;
  }
```

Dev tools

Самым удобным на текущий момент инструментом для отладки css grid является Firefox dev tools.

f12> Инспектор > разметка > сетка.

Визуализирует все составляющие вашей сетки, показывает lines, tracks, gaps, items

Явные и неявные track-и

Explicit and explicit tracks.

Когда мы явно указываем строки либо колонки - это явные треки. Если же grid-item-ов больше, чем явно указанных треков - новые треки могут создаваться автоматически (implicit tracks).

В Firefox dev tools явные и неявные треки отображаются по разному.

Явные и неявные track-и

Так как неявные треки создаются автоматически и мы не указывали желаемого размера, мы можем указать для всех implicit треков стандартные размеры с помощью свойств: grid-auto-columns, grid-auto-rows.

Пример

```
.container {
    display: grid;
    grid-template-columns: 150px 300px;
    grid-template-rows: 80px 90px;
    grid-auto-rows: 50px;
    grid-auto-columns: 110px;
    grid-gap: 15px;
}
```

Auto-flow

По умолчанию, если grid-item-ов больше, чем явных колонок, создаются новые неявные строки.

Стоит обратить внимание, что в таком случае свойство grid-auto-columns не будет применяться

Мы можем изменить это поведение с помощью свойства grid-auto-flow

grid-auto-flow: row | column | row dense | column dense

Размеры в css grid

Единицы измерений:

px - статические

% - динамические, однако если у нас есть gaps, то к сумме процентов добавится еще и сумма занимаемых gaps, что может привести к нежеланным последствиям

rems, ems, vw, vh и т.п.

Размеры в css grid

Новая единица измерений - fr (fractional unit)

Обозначает количество места, оставшегося после того, как все элементы были расположены в сетке.

пример: grid-template-columns: 200px 1fr 2fr;

Одна колонка в сетке будет занимать 200px, остальное место будет распределено таким образом: 1fr + 2fr = 3fr (все свободное место). 1\3 будет занимать вторая колонка, 2\3 - третья.

auto + fr: колонка с размером auto - размер всего трека по контенту, оставшееся место - fr.

Вместо auto мы можем использовать функцию css fit-content(arg), занимать место по контенту, но не больше чем размер указанный в arg: grid-template-columns: fit-content(150px) 200px 200px

repeat function

Если нужно указать несколько одинаковых треков, есть удобный инструмент - repeat function.

Например, вместо:

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

Можно просто указать:

grid-template-columns: repeat(4, 1fr); grid-template-columns: repeat(2 ,200px 1fr) 35px;

Первый аргумент - количество повторов, второй - размер трека\треков.

Размеры grid-item-ов

1. Можем указать явно конкретному grid-item-у размер. В таком случае, весь трек будет занимать такой же размер (как и в случае, если размер явно не указан, но есть много контента)

Такой вариант очень неточный, так как мы нарушаем явно указанную сетку. Поэтому есть второй вариант:

2. Spanning. можем указать, сколько ячеек будет занимать grid-item

Размеры grid-item-ов

Для того, чтобы указать, сколько ячеек будет занимать grid-item, есть такие свойства:

grid-column, grid-row.

Эти свойства применяются к grid-item-ам.

Например: grid-column: span 2 *//элемент будет занимать 2 колонки*

Размещение grid-items

Grid-column(row) - shorthand для свойств:

grid-column-start и grid-column-end (указываем линии на между которыми будет располагаться grid item.

grid-column-start: 2;

grid-column-end: 5;

Или: grid-column: 2 / 5; grid column: 2 / span 5;

grid-column: 1 / -1 // ***если мы хотим считать с конца, например, не зная точного количества колонок***

auto-fit и auto-fill

Если нам не важно, сколько нужно треков, значения grid-template-columns и grid-template-rows можно указать так:

grid-template-columns: repeat(auto-fill, 180px) или

grid-template-columns: repeat(auto-fit, 180px)

Разница: fill - “наполнит” трек указанными item-ами до конца, даже если не хватает элементов (так, например, последний элемент мы можем поместить в крайнюю ячейку)

fit - создаст столько ячеек, сколько у нас есть элементов для текущего трека.

minmax

minmax - функция css, которая позволяет указывать не 1 статический\динамический размер, а промежуток между минимальным и максимальным значениями item-ов в сетке. Позволяет уменьшить количество медиа-запросов.

В примере с auto-fill, если контент item-а будет больше 180px - верстка может поплыть, поэтому, вместо: grid-template-columns: repeat(auto-fill\fit, 180px) мы можем указать - grid-template-columns: repeat(auto-fill,\fit minmax(180px, 1fr));

Мы указываем, что нужно создать столько колонок, сколько поместится в трек, с минимальной шириной колонки в 180px, и максимальной - все оставшееся пространство.