# GRID LAYOUT

## **HTML**

Создадим простую разметку

<div class="grid">

<div>first column</div>

<div class="nested">second column</div>

<div>third column</div>

</div>

**CSS**

.grid {

display: grid;

**grid-template-columns: 50% 25% 25%**

grid-template-rows: 150px 150px;

grid-column-gap: 1em;

grid-row-gap: 1em;

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

grid-auto-flow: row; // column

}

**grid-template-columns** — устанавливает размер для элементов внутри по вертикали

**grid-template-rows** — устанавливает размер для элементов внутри по горизонтали. Пример выше установит высоту в 150px для первых двух рядов, остальные ниже будут высотой в 100px из-за условия grid-auto-rows

**grid-column-gap** — горизонтальные промежутки

**grid-row-gap** — вертикальныне промежутки

**grid-gap** — обьединяет промежутки по вертикали и горизонтали

**grid-auto-rows** — установка высоты для всех элементов. Функция minmax устанавливает минимульную высоту в 100 пикселей, и элемент будет растягивающимся. Остальные под него подстраиваются

**grid-auto-flow** — задает приоритет по горизонтали (row, по умолчанию) или вертикали (column). Для работы по вертикали необходимо задать свойство grid-template-rows

**grid-template:** **repeat(3, 200px) / repeat(4, 1fr)**

**grid-template** позволяет обьеденить grid-template-rows / grid-template-columns

## **РАБОТА СО МНОЖЕСТВОМ ЭЛЕМЕНТОВ**

grid-template-columns: 1fr 2fr 1fr

fr — пропорциональная установка размера. В примере выше средняя колонка будет занимать места как две колонки

grid-template-columns: repeat(4, 1fr 2fr)

Функция для повторения размеров. В данном случае вторая колонка будет в 2 раза больше первой, и так будет повторятся 4 раза (8 колонок)

grid-template-columns: 100px repeat(2, 1fr 2fr)

Первая колонка всегда будет шириной в 100px, еще 4 будут чередоваться в сотношении 1:2 в зависимости от ширины экрана

## **RESPONSIVE СТИЛИ**

grid-template-columns: repeat(auto-fill, 200px);

Колонки будут перестраиваться друг под друга если ширина экрана не позволяет колонке размером в 200px поместиться полностью на экране

grid-template-columns: repeat(auto-fill, minmax(320px, 1fr));

Тут колонки будут выстраиваться друг под друга, если ширина экрана не позволяет вместить всю колонку размером 150px. Если экан больше, они будут растягиваться по миксимуму, занимая равномерно все пространство

grid-template-columns: repeat(auto-fit, minmax(320px, 1fr));

Очень похожее свойство, отличие от auto-fill в том, что auto-fill оставляет пустые места справа, если элементов внутри него мало, а экран большой. auto-fit резинит колонки всегда на весь экран

grid-template-columns: minmax(100px, 250px) 1fr 1fr;

Установит 3 колонки: первая колонка будет занимать макс 250px и 100px минимум при уменьшении экрана, две остальные колонки резиновые

## **ПОЗИЦИОНИРОВАНИЕ ОДНОГО ЭЛЕМЕНТА**

.grid div:first-of-type {

grid-column-start: 1;

grid-column-end: 3;

}

Элемент займет расстояние с 1 по 3 клетку общей сетки.

.grid div:first-of-type {

grid-column: 1/3;

}

Сокращенный пример сверху

.grid div:first-of-type {

grid-column: 1/-1;

}

Заставит элемент занять всю ширину сетки

.grid div:nth-child(1) {

grid-row: 2;

}

.grid div:nth-child(2) {

grid-row: 1;

}

grid-row можно указывать где элемент должен находится по вертикали. В примере выше первый и второй элемент поменялись местами, безо всякого изменения html кода

## **РАСТЯГИВАНИЕ ЭЛЕМЕНТОВ**

justify-items: start\end\stretch;

align-items: start\end\stretch;

align-self: start\end\strech;

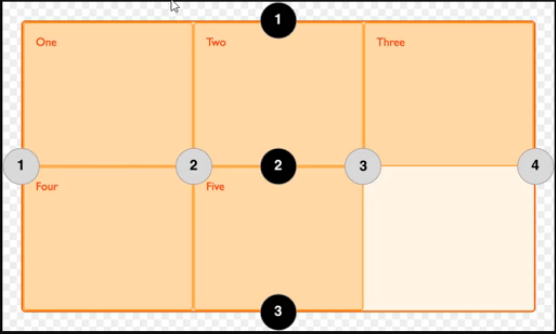
justify-self: start\end\strech;

Элементы смещаются к началу\концу\заполняют всю колонку (как в флексбоксе)

Так же элементы можно заставлять занимать нужное количество клеточек по вертикали или горизонтали, если дать отдельному элементу свой класс и прописать туда такое своейство:

grid-column: 1/3;

заставит **отдельный элемент** занять фракции с 1 по 3, или 75% экрана если всего 4 фракции

grid-row: 2/4

Позволит расширить элемент начиная со второй клетки заканчивая вниз до 4

При этом элементы могут естественно накладываться друг на друга, минуя использования отрицательных маргинов и z-index

## **GRID ОБЛАСТИ**

Например есть стандартная разметка: header, content, sidebar и footer

<main class="grid-areas">

<header>Header</header>

<article>

<h1>Article header</h1>

<p>Content here!!</p>

</article>

<aside>Sidebar</aside>

<footer>Footer</footer>

</main>

Стили к нему

.grid-areas {

display: grid;

grid-gap: 10px;

grid-template-areas:

"header"

"article"

"aside"

"footer";

}

header {

grid-area: header;

}

article {

grid-area: article;

}

aside {

grid-area: aside;

}

footer {

grid-area: footer;

}

@media (min-width: 640px) {

.grid-areas {

grid-template-columns: 1fr 2fr;

grid-template-areas:

"header header"

"aside article"

"footer footer";

}

}

В результате на небольших экранах элементы разметки выстраивается друг под другом.

На больших экранах получается разметка с хедером на всю длину, асайдом и артиклом посредине, причем артикл в 2 раза больше по размерам асайда

Если же указать так:

.grid-template-areas:

"header header"

"aside article"

"footer .";

}

footer займет только 1fr, остальная часть будет пустовать. Точка означает пропуск



.grid-template-areas:

"header header"

"aside article"

"aside footer";

}

Эти стили приведут к такому результату:

