**Вопросы 💎**

1. **Что за единица измерения - fr?**

Гибкая единица измерения, подстраивается под количество доступного пространства в грид-контейнере. Можно совмещать разными единицами измерения, например Px,%.

Пример: grid-template-columns: 1fr 1fr 1fr; - доступное пространство делится на 3 и распределяется поровну между тремя колонками

grid-template-columns: 2fr 1fr; - доступное пространство делится на 2, из которых 1 часть занимает в 2раза больше места, чем вторая.

1. **Как можно задать грид с 5 колонками шириной по 20%? Минимум 2 способа.**

**1 способ, при помощи Grid-template-coloums задаем проценты.**

.Container{

Display: grid;

Gtid-template-coloums:20% 20% 20% 20% 20%;

}

Html:

<div class=”container”>

<div class=”cell1”></div>

…

</div>

**2 способ.** Тоже при помощи при помощи Grid-template-coloums, но ширину колонок задаем при помощи fr. Т к колонок 5, то они будут равными, как раз по 20%:

Grid-template-coloums:1fr 1fr 1fr 1fr 1fr;

1. **Самостоятельно разберитесь, что такое auto-fill и auto-fit ?**

Это фичи для управления свободным местом, используются внутри функции repeat. Они заменяют собой числа в repeat, и вычисляют максимально возможное число колонок в контейнере. Используются как значения при указании ширины колонок в CSS Grid.

# auto-fill автоматически заполняет строку таким количеством колонок, как это возможно с учетом заданной ширины*.* Когда в строке не остается места, то элементы переходят на следующую строку.

# auto-fit – разница с auto-fill в том, как браузер использует оставшееся свободное место. любая пустая область будет схлопываться и растягивать элементы в этой строке. Т.е. пустые полосы как будто выбрасываются. Если полос вмещается пять, но элемента только три — в итоге и полос будет только три.

1. **Как сделать такую табличку? Параметры: первая колонка шириной 100 пикселей, вторая 30%. Первая строчка высотой 200 пикселей, вторая строчка 100 пикселей.**

**html:**

<div class="container">

        <div class="cell1">

        </div>

        <div class="cell1">

        </div>

        <div class="cell1">

        </div>

        <div class="cell1">

        </div>

        <div class="cell1">

        </div>

        <div class="cell1">

        </div>

    </div>

**Css:**

.container{

    display: grid;

    grid-template-columns: 100px 30% 1fr;

    grid-template-rows: 200px 100px;

    margin: 10px;

    gap: 10px;

}

.cell1{

    background-color: rgb(98, 98, 228);

    border: 10px solid black;

}

1. **Как сделать такое выравнивание в грид-контейнере?**

**Html:**

  <div class="container">

        <div class="cell1">

            яч1

        </div>

        <div class="cell1">

            яч2

        </div>

        <div class="cell1">

            яч3

        </div>

        <div class="cell1">

            яч4

        </div>

        <div class="cell1">

            яч5

        </div>

        <div class="cell1">

            яч6

        </div>

        <div class="cell1">

            яч7

        </div>

        <div class="cell1">

            яч8

        </div>

        <div class="cell1">

            яч9

        </div>

    </div>

**Css:**

.container{

    display: grid;

    justify-items: center;

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

    grid-template-rows: 1fr 2fr 1fr;

}

.cell1{

    background-color: rgb(250, 212, 0);

    border: 2px solid green;

}

1. **Что такое и как задается *grid area*?**

Это пространство внутри grid контейнера, в которое может быть помещен один или больше grid элементов. Этот элемент может состоять из одной или более grid ячеек.

Свойство grid-area является сокращенным свойством для grid-row-start, grid-column-start, grid-row-end и grid-column-end, определяя размер и расположение элемента сетки.

Пример: .box1 { grid-area: 2 / 1 / 4 / 3; }

1. **Приведите пример использования grid-template-areas (не копированием из этого урока 😉)**

C помощью **grid-template-areas** можно удобным и логически понятным образомопределить грид-область, ее имя и месторасположение.

Для этого вначале при помощи grid-area мы задаем каждой из областей собственное имя, а затем в **grid-template-areas** прописываем эти имена (сколько строк в контейнере, столько и в **grid-template-areas,** каждую строку берем в “”.

Пример:

footer {

    display: grid;

grid-template-columns: 1fr 1fr;

    grid-template-areas: "section-footer-1 section-footer-2”

“section-footer-3 section-footer-3"

“section-footer-4 section-footer-4;

}

.section-footer-1 {

    grid-area: section-footer-1;

}

.section-footer-2 {

    grid-area: section-footer-2;

}

.section-footer-3 {

    grid-area: section-footer-3;

}

.section-footer-4 {

    grid-area: section-footer-4;

}

1. **Каким свойством можно задать такое поведение элементов?**

Ссылка на анимацию: <http://css-live.ru/Primer/grid1/1-MyO6enrNZEL9HSzYmdA8VA.gif>

Такое поведение можно задать свойством auto-fit

1. **Самостоятельно разберитесь, как работают именованные линии? Есть ли какие-то рекомендованные правила наименований? Если да, то какие?**

Имена линиям можно задать внутри квадратных скобок в списке треков. Назвав линии, можно использовать их имена вместо номеров при позиционировании элементов на сетке.

Имя может быть любым, кроме ключевого слова span.

Хорошей практикой является именование с использованием суффикса -start для начальных линий и -end для конечных.

Пример: grid-row: header-start;

grid-template-rows: [header-start] 20vh [header-end] ;

1. **Как проще всего задать 12 одинаковых по ширине колонок?**

**Задать их с помощью fr и repeat, например :**

.Container{

Display: grid;

grid-template-columns: repeat(12, 1fr);

}