Ответы на вопросы:

1. 1fr – это одна фракция. Использование данного метода предполагает задание ширины треков с помощью единиц измерения fr, рассчитывается исходя из свободного пространства грид контейнера (после того, как пространство заберет элемент фиксированного размера или длинное слово без возможности переноса). Например, задаем треку колонки: 1fr 1fr1 1fr – доступное пространство будет разделена на 3 одинаковых колонки.
2. Задать грид 5 колонок по 20%

html

<div class="page">

<p>1</p>

<p>2</p>

<p>3</p>

<p>4</p>

<p>5</p>

</div>

css (варианты)

.page {

display: grid;

grid-template-columns: repeat(5, 20%);

}

.page {

display: grid;

grid-template-columns: 20% 20% 20% 20% 20%;

}

если понимаем, что доступного пространства 100%, то тогда можно воспользоваться фракциями

.page {

display: grid;

grid-template-columns: repeat(5, 1fr);

}

.page {

display: grid;

grid-template-columns: [full-start] repeat(5, 1fr) [full-end];

}

.page {

display: grid;

grid-template-columns: [full-start sidebar-start] 1fr [sidebar-end main-start] 1fr 1fr 1fr 1fr [main-end full-end];

}

1. auto-fit и auto-fill нужны для управления свободным местом.

auto-fill

Мы можем не указывать сколько нужно повторений колонок, мы можем указать, что нужно уместить как можно больше колонок с учетом указанной ширины.

Например: grid-template-columns: repeat(auto-fill, 100px).

Если нам надо, чтобы колонки были минимум в 100px и максимум в 1fr, то укажем:

grid-template-columns: repeat(auto-fill, minmax(100px, 1fr)). Таким образом, браузер уместит в строку столько колонок с минимальной шириной в 100px, сколько он сможет. А если останется ещё место, то оно будет поровну распределено между всеми автозаполненными колонками.

auto-fit

если мы укажем с помощью auto-fit, что делать с оставшимся местом в строке -

grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));

то если есть место для дополнительных колонок, но их нечем заполнить, то браузер сделает их ширину равную нулю. А если выставить максимальную ширину колонок на 1fr (как выше), то браузер разделит оставшееся место поровну между ними.

1. Код таблички (так как не указан размер 3го столбца сделала его 1fr, чтобы он забрал все оставшееся место)

Html

<div class="q4">

<section class="nana">nana</section>

<section class="popo">popo</section>

<section class="tutu">tutu</section>

<section class="wewe">wewe</section>

<section class="tutu">tutu</section>

<section class="wewe">wewe</section>

</div>

css

.q4 {

display: grid;

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

grid-template-rows: 200px 100px;

grid-gap:10px;

text-align: center;

}

.q4 section {

background-color: aqua;

}

1. Justify-content: space-between;
2. **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.

Расположение может указать с помощью номеров линий: старт ряд/старт колонка/конец ряд/конец колонка. Либо можем дать имя grid area и использовать его при указании grid-template-areas.

1. пример grid-template-areas

html

<div class="page">

<header class="header">header</header>

<secsion class="one">one</secsion>

<secsion class="two">two</secsion>

<secsion class="three">three</secsion>

<secsion class="four">four</secsion>

<aside class="sidebar">sidebar</aside>

<footer class="footer">footer</footer>

</div>

css.

.page {

display: grid;

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

grid-template-rows: 100px 180px 180px 100px;

grid-template-areas:

"header header header header"

"sidebar one one two"

"sidebar three four four"

"footer footer footer footer";

gap: 20px;

}

.header {

grid-area: header;

background-color: #2fa960;

}

.sidebar {

grid-area: sidebar;

background-color: darkcyan;

}

.one {

grid-area: one;

background-color: aquamarine;

}

.two {

grid-area: two;

background-color:beige;

}

.three {

grid-area: three;

background-color: blue;

}

.four {

grid-area: four;

background-color: blueviolet;

}

.footer {

grid-area: footer;

background-color: crimson;

}

8. auto-fit

9. В гридах можно задавать не только именованные области, а также индивидуальные имена даже для обычных линий.

Когда нужно опеделить сетку с помощью свойств grid-template-rows и grid-template-columns, то можно присвоить имя некоторым или всем линиям в сетке. Имена линиям можно задать внутри квадратных скобок в списке треков.

Пример:

.div {

display: grid;

grid-template-columns: [main-start] 1fr [content-start] 1fr [content-end] 1fr [main-end];

grid-template-rows: [main-start] 100px [content-start] 100px [content-end] 100px [main-end];

}

Правила именования линий:

Имя может быть любым, кроме ключевого слова span. Хорошей практикой является именование с использованием суффикса -start для начальных линий и -end для конечных. В итоге должно получиться, например, main-start и main-end.

Довольно часто конечная линия одной части сетки и начальная линия другой части совпадают, но это не проблема, так как линии могут иметь несколько имён. Задать линии несколько имён можно, добавив их через пробел внутри квадратных скобок.

10.

.page {

display: grid;

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

}