**1. flex**

<https://flexboxfroggy.com/#ru>

**1.1. justify-content** – выравнивание элементов **горизонтально**:

**flex-start**: элементы выравниваются по левой стороне контейнера.

**flex-end**: элементы выравниваются по правой стороне контейнера.

**center**: элементы выравниваются по центру контейнера.

**space-between**: элементы отображаются с одинаковыми отступами между ними.

**space-around**: элементы отображаются с одинаковыми отступами вокруг них

**1.2 align-items** - выравнивание элементов **вертикально**:

**flex-start**: элементы выравниваются по верхнему краю контейнера.

**flex-end**: элементы выравниваются по нижнему краю контейнера.

**center**: элементы выравниваются вертикально по центру контейнера.

**baseline**: элементы отображаются на базовой линии контейнера.

**stretch**: элементы растягиваются, чтобы заполнить контейнер.

**align-self:** eщё одно свойство, которое ты можешь применить к определенному элементу. Это свойство принимает те же значения, что и align-items.

**1.3 flex-direction** - задает направление, в котором будут расположены элементы в контейнере, и принимает следующие значения:

**row**: элементы размещаются по направлению текста.

**row-reverse**: элементы отображаются в обратном порядке к направлению текста.

**column**: элементы располагаются сверху вниз.

**column-reverse**: элементы располагаются снизу вверх.

**1.4** Иногда изменения порядка отображения элементов в контейнере недостаточно. В таких случаях мы можем применить свойство **order:** для конкретных элементов. По умолчанию, значение этого свойства у элементов **равно 0**, но мы можем задать положительное или отрицательное целое число этому свойству.

**1.5 flex-wrap** - раздвинуть элементы:

**nowrap**: размеры элементов устанавливаются автоматически, чтобы они поместились в один ряд.

**wrap:** элементы автоматически переносятся на новую строку.

**wrap-reverse:** элементы автоматически переносятся на новую строку, но строки расположены в обратном порядке.

**1.6** Два свойства **flex-direction** и **flex-wrap** используются так часто вместе, что было создано свойство **flex-flow** для их комбинирования. Это свойство принимает их значения, разделённые пробелом.

Например, ты можешь использовать **flex-flow: row wrap**, чтобы элементы располагались в ряд и автоматически переносились на новую строку.

**1.7** Ты можешь использовать **align-content**, чтобы указать, как несколько рядов должны отделяться друг от друга. Данное свойство принимает следующие значения:

**flex-start**: ряды группируются в верхней части контейнера.

**flex-end**: ряды группируются в нижней части контейнера.

**center**: ряды группируются вертикально по центру контейнера.

**space-between**: ряды отображаются с одинаковыми расстояниями между ними.

**space-around**: ряды отображаются с одинаковыми расстояниями вокруг них.

stretch: ряды растягиваются, чтобы заполнить контейнер равномерно.

Это может запутать, но **align-content** отвечает за расстояние между рядами, в то время как align-items отвечает за то, как элементы в целом будут выровнены в контейнере. Когда только один ряд, align-content ни на что не влияет.

Задача 24 уровня:

flex-flow: column-reverse wrap-reverse;

justify-content: center;

align-content: space-between

**2. grid**

grid-column-start

grid-column-end

Вместо определения grid-элемента на основе начальной и конечной позиции grid-строк вы можете задавать их с помощью необходимой вам ширины строк, используя **span**. Помните, что span работает только с положительными значениями: **grid-column-end: span 2.**

Печатать каждый раз grid-column-start и grid-column-end может быть немного утомительно. К счастью, есть краткая форма **grid-column**, которая принимает оба значения сразу через косую черту: /.

Например, grid-column: 2 / 4 скажет grid-элементу начаться на второй вертикальной grid-линии и закончиться на четвёртой.

**grid-row-start**

Одна из вещей, которая отличает CSS Grid Layout от Flexbox'а, — это то, что вы можете легко позиционировать элементы в двух направлениях: в столбцах и в строках. **grid-row-start** работает почти как grid-column-start, но только по вертикальной оси.

**grid-area**

Если вас также утомляет печатать grid-column и grid-row, есть сокращённое свойство и для этого. **grid-area** принимает 4 значения, разделенные косой чертой /: grid-row-start, grid-column-start, grid-row-end и grid-column-end.

Пример: grid-area: 1 / 1 / 3 / 6;

**order**

Если grid-элементы не имеют конкретного расположения с grid-area, grid-column, grid-row и т.д., они автоматически размещаются, следуя порядку, написанному в коде. Мы можем изменить это с помощью свойства **order**, которое является одним из преимуществ CSS Grid Layout перед табличной разметкой.

Изначально все grid-элементы имеют order, равный 0, но этому свойству можно присвоить любое положительное или отрицательное значение, примерно как у z-index.

**grid-template-columns**

До этого момента сад имел grid-сетку с пятью столбцами по 20% ширины и пятью строками, каждая по 20% высоты.

Это было сделано свойствами **grid-template-columns**: 20% 20% 20% 20% 20%; и grid-template-rows: 20% 20% 20% 20% 20%; Каждое свойство имеет пять значений, которые создают пять столбцов, где ширина каждого равна 20% от общей ширины.

Определять несколько столбцов с одинаковой шириной может быть утомительно. К счастью, есть функция **repeat**, которая призвана помочь в этом.

Например, раньше мы определяли пять столбцов по 20% ширины с помощью grid-template-columns: 20% 20% 20% 20% 20%; Это можно упростить до **grid-template-columns: repeat(5, 20%)**.

grid-template-columns принимает значения не только в процентах, но и в единицах длины, например px или em. Вы даже можете комбинировать разные единицы измерения.

Присвойте трём столбцам значение **100px, 3em и 40% соответственно.**

Когда одни столбцы определены с помощью пикселей, процентов или em, а любые другие столбцы — с помощью fr, то последние просто разделят всё возможное оставшееся пространство.

Сейчас морковь занимает 50 пикселей слева, а сорняки 50 пикселей справа. С помощью grid-template-columns создайте два столбца и используйте fr, чтобы сделать ещё 3 столбца, которые займут оставшееся простанство между ними.

**grid-template**

grid-template — сокращённый вариант комбинации grid-template-rows и grid-template-columns.

Например, grid-template: 50% 50% / 200px создаст grid-сетку с двумя строками по 50% каждая и одним столбцом шириной 200 пикселей.