МЕЖДУНАРОДНЫЙ ВОСТОЧНО-ЕВРОПЕЙСКИЙ УНИВЕРСИТЕТ (МВЕУ)

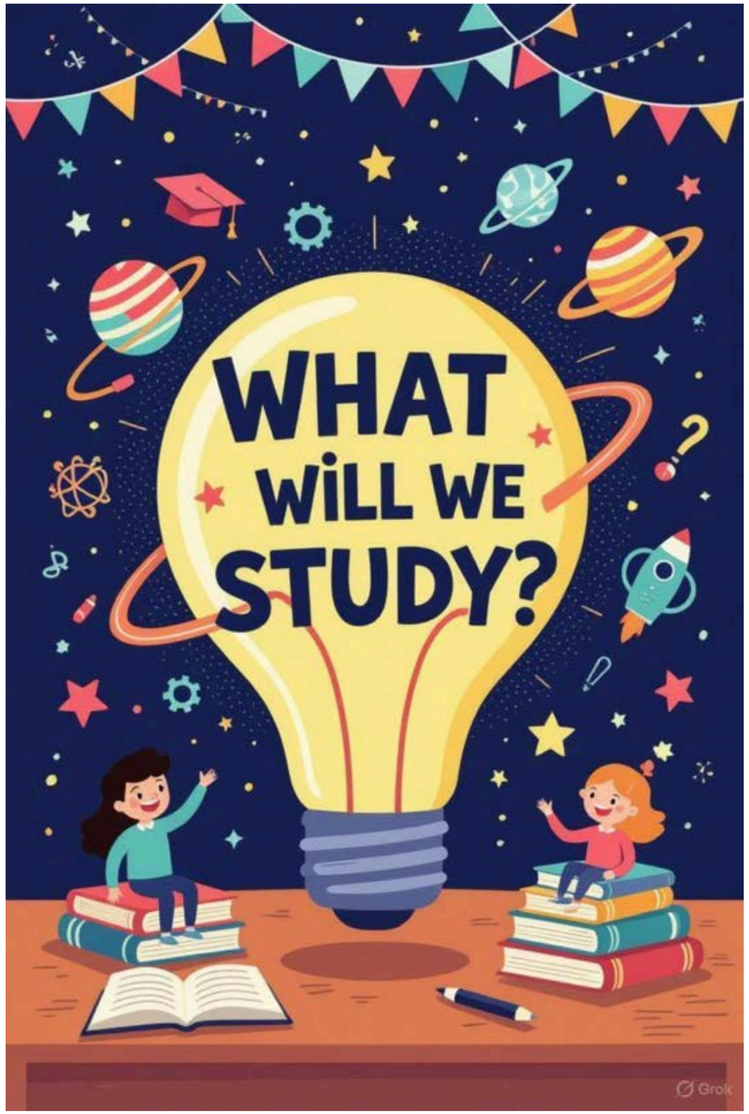
Преподаватель:
Коляда
Никита Владимирович

Обратная связь:
- сообщения на inStudy

ОСНОВЫ FLEXBOX GRID LAYOUT

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА
ИНТЕРФЕЙСОВ ПОЛЬЗОВАТЕЛЯ

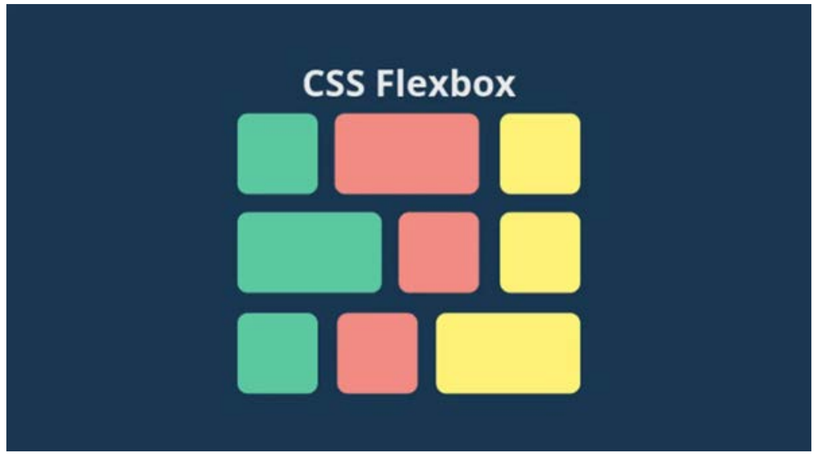
2025 Г



ЧТО ИЗУЧИМ СЕГОДНЯ?

1. Что такое гибкая верстка (flexbox)?
2. Что такое сетка (grid layout)?

ЧТО ТАКОЕ ГИБКАЯ ВЕРСТКА (**FLEXBOX**)?



Долгое время веб-интерфейсы были **статичными** — сайты разрабатывались и просматривались **только на экранах мониторов стационарных компьютеров.**

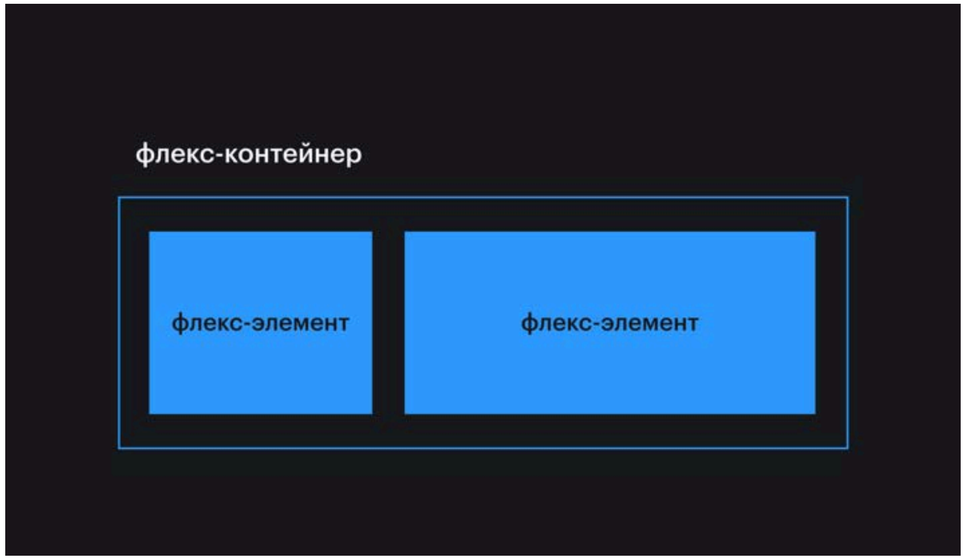
Однако с десяток лет назад, совсем недавно по историческим меркам, у нас **появилось огромное разнообразие экранов** — от мобильных телефонов до телевизоров, — на которых мы можем взаимодействовать с сайтами.

Так родилась **необходимость в гибких системах** раскладки.

Идея флексбоксов появилась ещё в 2009 году, и этот стандарт до сих пор развивается и прорабатывается.

Основная идея флексов — **гибкое распределение места между элементами**, гибкая **расстановка**, **выравнивание**, гибкое **управление.** Ключевое слово — **гибкое**, что и отражено в названии (flex — англ. гибко).

ОСНОВНЫЕ ТЕРМИНЫ

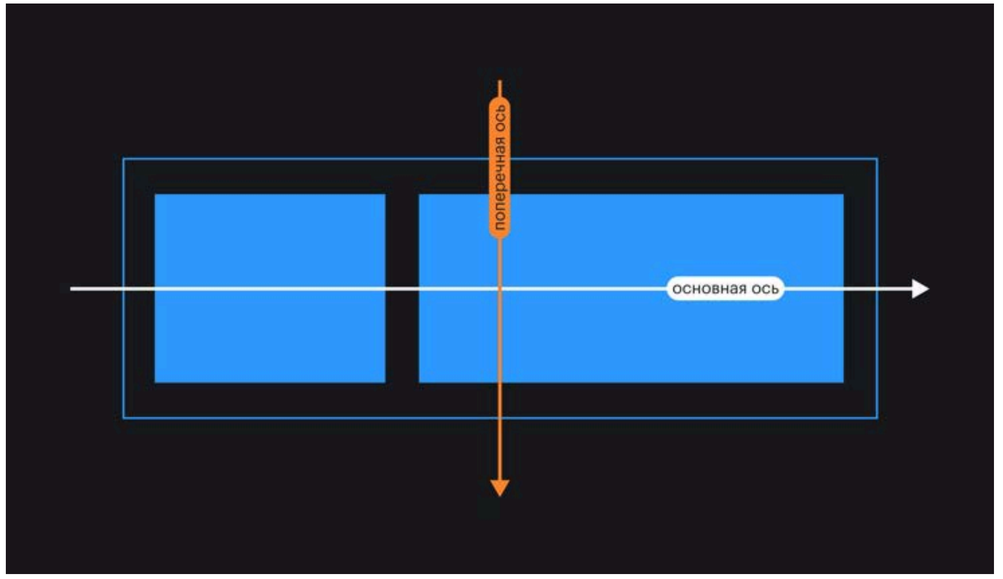


Флекс-контейнер: элемент, к которому применяется свойство **display: flex**. Вложенные в него элементы подчиняются правилам раскладки флексов.

Флекс-элемент: элемент, вложенный во флекс-контейнер.

Основная ось: основная направляющая флекс-контейнера, вдоль которой располагаются флекс-элементы.

Поперечная (побочная, перпендикулярная) ось: ось, идущая перпендикулярно основной. Позже вы поймёте, для чего она нужна.



СВОЙСТВА ФЛЕКС-КОНТЕЙНЕРА

```
.flex-container {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  justify-content: space-between;
  align-items: center;
  gap: 10px;
```

display: flex / inline-flex превращает элемент в flex контейнер. При flex — блочный по внешнему поведению; inline-flex — строчный.

flex-direction задаёт направление основной оси: row, row-reverse, column, column-reverse.

flex-wrap управляет тем, будут ли флекс-элементы переноситься на новую строку/ряд при переполнении: nowrap (по умолчанию), wrap, wrap-reverse.

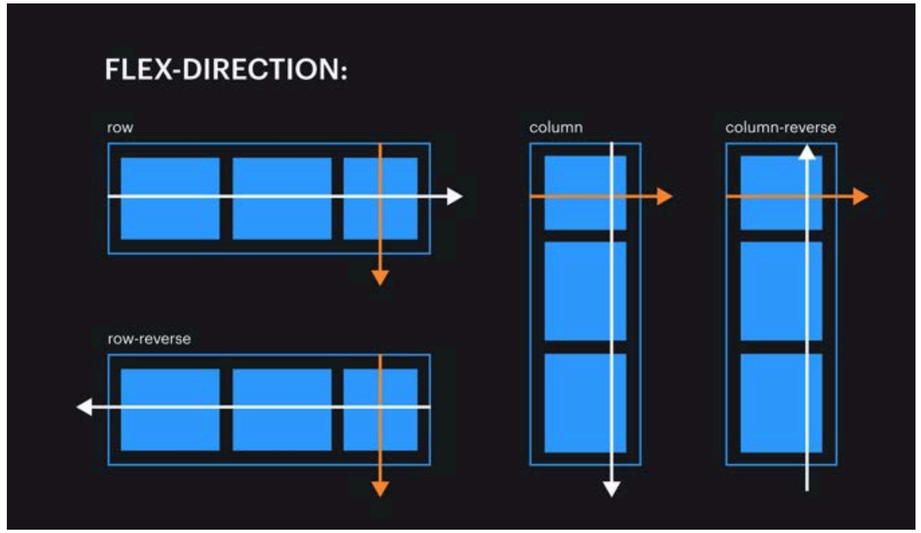
justify-content выравнивание элементов вдоль основной оси: начало, конец, центр, распределение пространства между элементами.

align-items выравнивание вдоль поперечной оси: как элементы растягиваются/выровнены относительно поперечной оси.

align-content как распределяется свободное пространство между рядами / линиями по поперечной оси, если элементов несколько рядов.

gap задаёт отступы между элементами; шорткат row-gap и column-gap.

СВОЙСТВО **FLEX-DIRECTION**

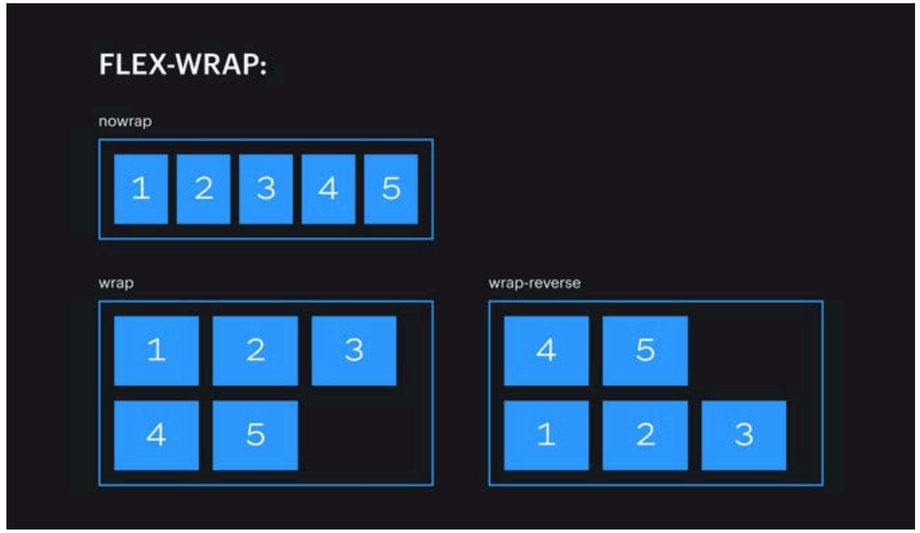



```
1 .container {
2   display: flex;
3   flex-direction: row;
4 }
```

flex-direction задаёт направление основной оси:

- **row** (значение по умолчанию) — основная ось идёт горизонтально слева направо, поперечная ось идёт вертикально сверху вниз.
- **row-reverse** — основная ось идёт горизонтально справа налево, поперечная ось идёт вертикально сверху вниз.
- **column** — основная ось идёт вертикально сверху вниз, поперечная ось идёт горизонтально слева направо.
- **column-reverse** — основная ось идёт вертикально снизу вверх, поперечная ось идёт горизонтально слева направо.

СВОЙСТВО **FLEX-WRAP**




```
1 .container {
2   display: flex;
3   flex-wrap: nowrap;
4 }
```

flex-wrap управляет тем, будут ли флекс-элементы переноситься на новую строку/ряд при переполнении:

- По умолчанию значение у свойства **flex-wrap — nowrap**. При этом флекс-элементы помещаются (или **пытаются уместиться**) в один ряд и **не переносятся** в новый ряд, даже если не влезают в размеры родителя.
- Установив значение **wrap**, мы можем изменить это поведение, и флекс-элементы будут иметь возможность **перенестись** в новый ряд, **если не влезают** в одну линию в рамках родителя.
- Ещё одно возможное значение — **wrap-reverse**. В этом случае элементы будут располагаться **снизу вверх**, заполнив собой сперва нижний ряд, а те, что не влезли, перепрыгнут в ряд выше.

СВОЙСТВО **JUSTIFY-CONTENT**

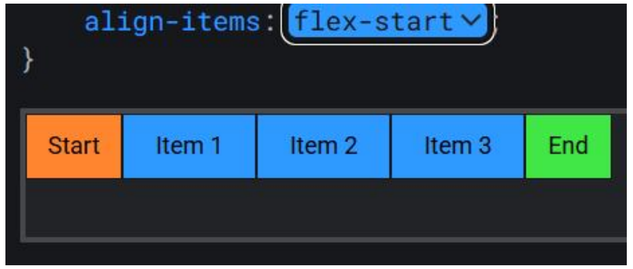
justify-content Свойство позволяет выравнивать флекс-элементы внутри флекс-контейнера по основной оси:

- **start** — элементы прижимаются к тому краю, откуда начинается чтение на том языке, на котором отображается сайт.
- **end** — элементы прижимаются к краю, противоположному началу направления чтения на языке сайта.
- **flex-start** — элементы прижимаются к краю, от которого начинается основная ось.
- **flex-end** — элементы прижимаются к краю, у которого основная ось заканчивается.
- **left** — элементы прижмутся к левому краю родителя.
- **right** — элементы прижмутся к правому краю родителя.
- **center** — элементы выстраиваются по центру родителя.
- **space-between** — крайние элементы прижимаются к краям родителя, оставшиеся выстраиваются внутри контейнера равномерно, так, чтобы между ними были одинаковые отступы.
- **space-around** — свободное пространство делится поровну между элементами и по половине от этой доли размещается по бокам от каждого элемента. Таким образом, между соседними элементами будет равное расстояние, а снаружи крайних элементов — по половине этого расстояния.
- **space-evenly** — свободное место будет распределено так, чтобы расстояние между любыми двумя элементами было одинаковым и расстояние от крайних элементов до края было таким же.

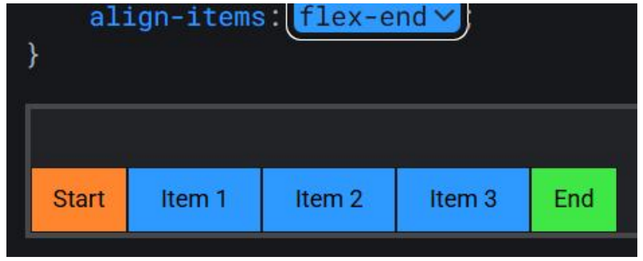
СВОЙСТВО **ALIGN-ITEMS**

```
.container {
    width: 100%;
    height: 320px;
    display: flex;
    flex-direction: row;
    flex-wrap: nowrap;
    justify-content: flex-start;
    align-items: stretch;
}
```


```
    align-items: flex-start;
}
```




```
    align-items: flex-end;
}
```

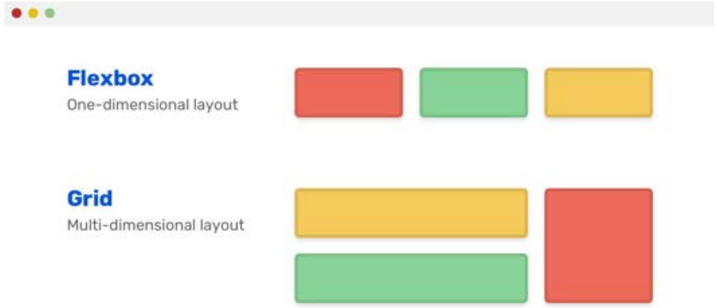


align-items Свойство выравнивания элементов внутри контейнера по поперечной оси:

- **stretch** (значение по умолчанию) — элементы растягиваются вдоль поперечной оси так, чтобы заполнить всего родителя. Это очень удобно, если вы делаете двухколоночный макет. Раньше приходилось при помощи разных костылей добиваться одинаковой высоты, а теперь достаточно сделать контейнер флексом, и колонки по умолчанию будут одной высоты.
- **flex-start или start** — элементы выстраиваются у начала поперечной оси. Разница между ними лишь в том, что второе значение «уважает» направление чтения выбранного языка.
- **flex-end или end** — элементы выстраиваются у конца поперечной оси. Разница между первым и вторым значениями аналогична предыдущему пункту.
- **center** — элементы выстраиваются по центру поперечной оси.
- **baseline** — элементы выравниваются по базовой линии текста. «Базовая линия» — baseline — воображаемая линия, проходящая по нижнему краю знаков шрифта (без учёта выносных элементов).

```
    align-items: center;
}
```

ЧТО ТАКОЕ СЕТКА (**GRID LAYOUT**)?



Что такое Grid Layout

- **CSS Grid Layout** — это двухмерная система раскладки: можно управлять расположением элементов и по горизонтали, и по вертикали одновременно.
- Сетка задаётся в **контейнере**, а элементы внутри автоматически становятся **ячейками.**
- **Grid** — это современная альтернатива таблицам и сложным сеткам на float или Flexbox, **когда нужна чёткая структура.**

Основные термины

- **Grid-контейнер** — элемент, которому задано display: grid (или inline-grid).
- **Grid-элементы** — дочерние элементы контейнера (ячейки сетки).
- **Линии сетки** — границы колонок и рядов (grid lines).
- **Треки** — сами ряды и колонки (grid tracks).
- **Ячейка сетки** — пересечение ряда и колонки (grid cell).

СВОЙСТВА GRID-КОНТЕЙНЕРА

Свойства Grid-контейнера

display: grid / inline-grid превращает элемент в grid-контейнер

grid-template-columns задаёт количество и ширину колонок (например 200px 200px или repeat(3, 1fr))

grid-template-rows задаёт количество и высоту рядов

grid-auto-rows / grid-auto-columns размеры автоматически создаваемых рядов/колонок

gap / row-gap / column-gap расстояния между ячейками

justify-items выравнивание содержимого ячеек по горизонтали

align-items выравнивание содержимого ячеек по вертикали

justify-content выравнивание всей сетки внутри контейнера по горизонтали

align-content выравнивание всей сетки по вертикали

Свойства Grid-элемента

grid-column-start / grid-column-end указывает, между какими линиями колонок разместить элемент

grid-row-start / grid-row-end аналогично, для рядов

grid-column / grid-row шорткаты: grid-column: start / end

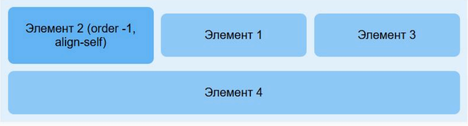
grid-area позволяет одновременно указать позицию по рядам и колонкам

justify-self выравнивание конкретного элемента по горизонтали внутри своей ячейки

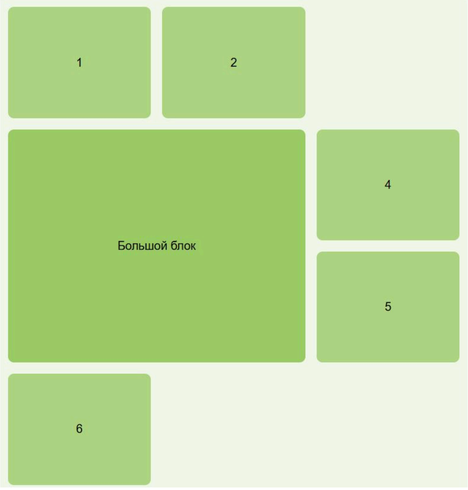
align-self выравнивание конкретного элемента по вертикали внутри своей ячейки

Flexbox и Grid Layout — пример

Flexbox контейнер



Grid контейнер



index.html

```html
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Flexbox и Grid — пример</title>
  <style>
    /* ====== FLEXBOX ====== */

    .flex-container {
      display: flex; /* превращаем контейнер в Flexbox */
      flex-direction: row; /* элементы расположены по основной оси — горизонтальн
      flex-wrap: wrap; /* разрешаем перенос элементов на новую строку */
      justify-content: space-between; /* равномерно распределяем элементы по осно
      align-items: center; /* выравниваем элементы по поперечной оси (вертикали)
      gap: 10px; /* зазор между элементами */
      background: #e3f2fd;
      padding: 10px;
      margin-bottom: 20px;
    }

    .flex-item {
      flex: 1 1 150px;
      /*
      flex-grow:1 (может расти);
      flex-shrink:1 (может сжиматься);
      flex-basis:150px (базовый размер)
      */
      background: #90caf9;
      color: #000;
      text-align: center;
      padding: 20px;
      border-radius: 8px;
    }

    .flex-item.special {
      order: -1; /* изменяем порядок элемента — он появится первым */
      align-self: flex-end; /* переопределяем выравнивание для одного элемента */
      background: #64b5f6;
    }

    /* ====== GRID LAYOUT ====== */

    .grid-container {
      display: grid; /* превращаем контейнер в Grid */
      grid-template-columns: repeat(3, 1fr);
      /* 3 колонки одинаковой ширины */
      grid-auto-rows: 150px; /* автоматическая высота строк */
      gap: 15px; /* отступы между ячейками */
      background: #f1f8e9;
      padding: 10px;
    }

    .grid-item {
      background: #aed581;
      color: #000;
      display: flex; /* внутри Grid используем Flexbox для выравнивания текста */
      justify-content: center;
      align-items: center;
      border-radius: 8px;
    }

    .grid-item.big {
      grid-column: span 2; /* растянуть элемент на две колонки */
      grid-row: span 2; /* растянуть элемент на две строки */
      background: #9ccc65;
    }

    /* Для адаптивности: при узком экране — 1 колонка */
    @media (max-width: 600px) {
      .grid-container {
        grid-template-columns: 1fr;
      }
    }

    body {
      font-family: Arial, sans-serif;
      margin: 20px;
      background: #fafafa;
    }

    h1 {
      text-align: center;
    }
  </style>
</head>
<body>
  <h1>Flexbox и Grid Layout — пример</h1>

  <!-- FLEXBOX -->
  <h2>Flexbox контейнер</h2>
  <div class="flex-container">
    <!-- flex: 1 1 150px -->
    <div class="flex-item">Элемент 1</div>
    <div class="flex-item special">Элемент 2 (order -1, align-self)</div>
    <div class="flex-item">Элемент 3</div>
    <div class="flex-item">Элемент 4</div>
  </div>

  <!-- GRID -->
  <h2>Grid контейнер</h2>
  <div class="grid-container">
    <!-- стандартный элемент сетки -->
    <div class="grid-item">1</div>
    <div class="grid-item">2</div>
    <!-- элемент, растянутый на 2 колонки и 2 строки -->
    <div class="grid-item big">Большой блок</div>
    <div class="grid-item">4</div>
    <div class="grid-item">5</div>
    <div class="grid-item">6</div>
  </div>
</body>
</html>
```