ДИСЦИПЛИНА	Фронтенд и бэкенд разработка
ИНСТИТУТ	Институт перспективных технологий и индустриального программирования
КАФЕДРА	Кафедра индустриального программирования
ВИД УЧЕБНОГО МАТЕРИАЛА	Практические занятия
ПРЕПОДАВАТЕЛЬ	Загородних Николай Анатольевич
СЕМЕСТР	1 семестр, 2024-2025 гг.

Практическое занятие 6. Работа с сетками и разметкой в HTML

Краткая теория

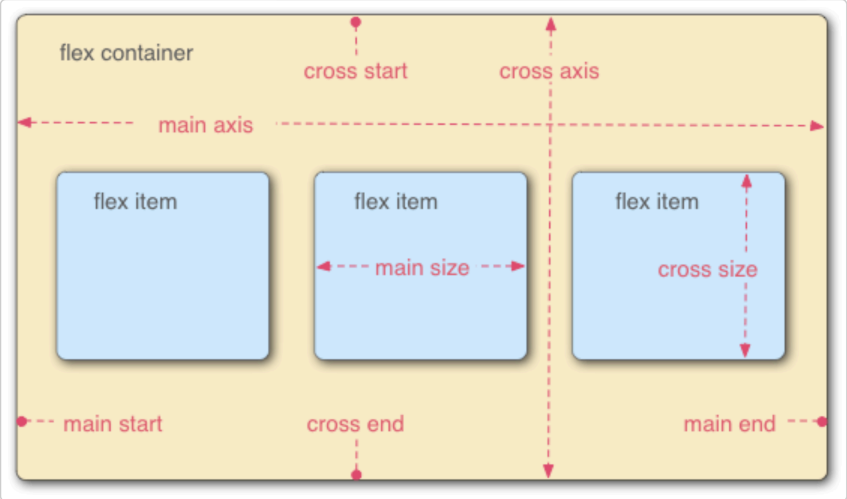
Grid Layout и Flexbox — это современные методы разметки и позиционирования элементов на веб-странице.

- Flexbox (Flexible Box Layout) позволяет удобно размещать элементы в одном направлении (по строкам или столбцам), автоматически подстраивая их размеры и отступы. Он особенно полезен для создания адаптивных интерфейсов.

- Grid Layout (CSS Grid) предоставляет более мощные возможности для создания сложных макетов, позволяя располагать элементы в двумерной сетке. Это идеальный выбор для сложных дизайнов, где необходимо контролировать как строки, так и столбцы.

Flex - модель

Когда элементы выложены как flex блоки, они располагаются вдоль двух осей:



Главная ось (main axis) проходит в том направлении, вдоль которого расположены Flex элементы (например, в строку слева направо или вдоль колонок вниз.) Начало и конец этой оси называются main start и main end.

Поперечная ось (cross axis) проходит перпендикулярно Flex элементам. Начало и конец этой оси называются cross start and cross end.

Родительский элемент, на который назначено свойство display: flex называется flex container.

Элементы, размещённые в нём как Flex блоки называются flex items.

Flexbox имеет несколько ключевых свойств, которые позволяют управлять поведением flex-контейнеров и flex-элементов.

1. display: flex;

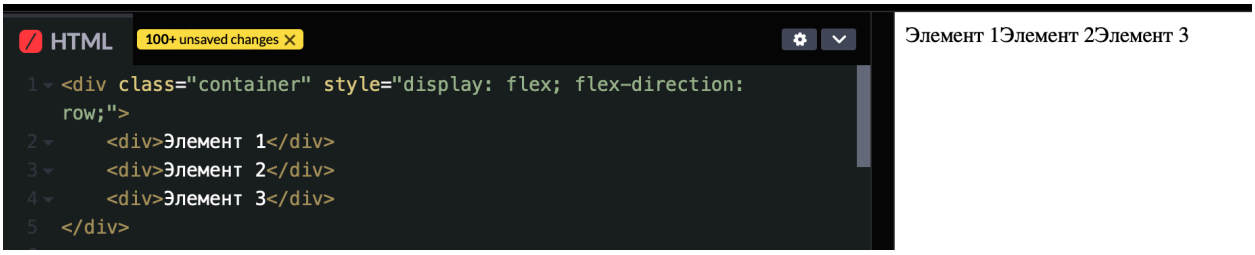
Это свойство активирует режим flex для контейнера. Все дочерние элементы становятся flex-элементами и начинают подчиняться правилам flexbox.

```
.container {
    display: flex;
}
```

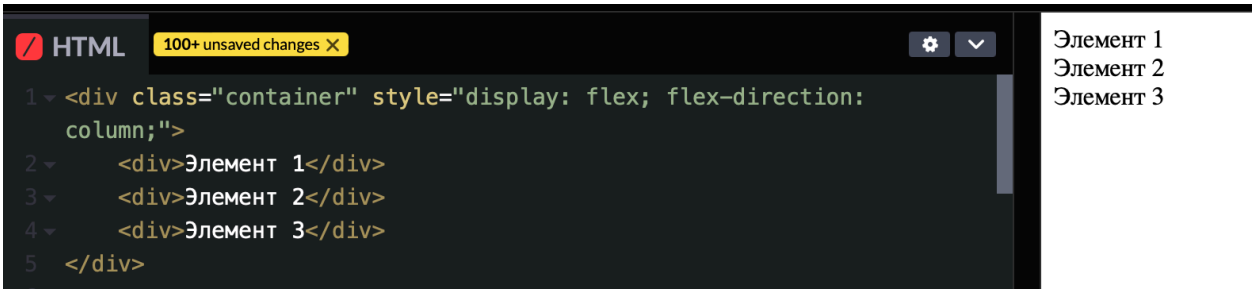
2. flex-direction

Свойство flex-direction определяет направление размещения flex-элементов в контейнере. Оно может принимать следующие значения:

2.1. row (по умолчанию). Это значение является стандартным и наиболее часто используемым. Используйте row, когда вам нужно расположить элементы в строку, например, для навигационных панелей, карточек товаров или любых других горизонтальных макетов.



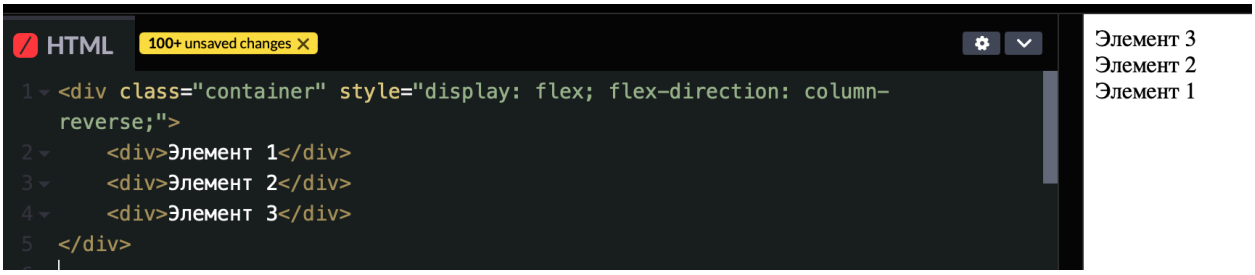
2.2. column. Используйте column, когда вам нужно расположить элементы вертикально. Это может быть полезно для форм, списков или любых других интерфейсов, где элементы должны располагаться друг под другом.



2.3. row-reverse. Это значение полезно, когда вам нужно изменить порядок отображения элементов в строке, например, для создания интерфейсов, где последние

добавленные элементы должны отображаться первыми. Может быть полезно в ситуациях, когда вы хотите показать самые новые сообщения вверху.

2.4. column-reverse. Используйте это значение, когда необходимо изменить порядок отображения элементов в столбике. Это может быть актуально для интерфейсов, где последние добавленные элементы должны быть сверху, например, в чатах или лентах новостей.

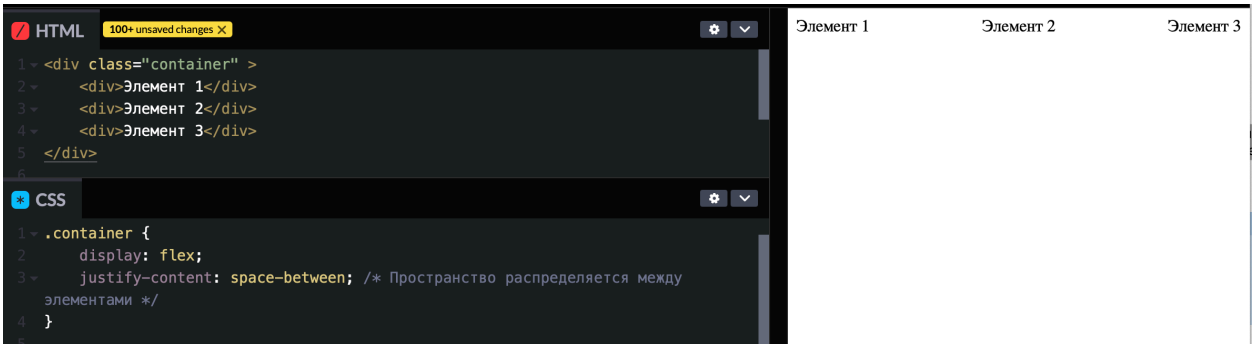


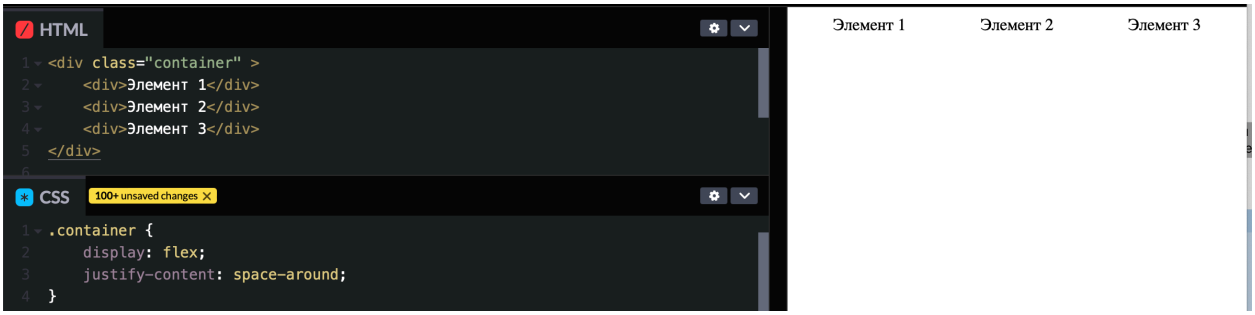
Выбор значения для flex-direction зависит от того, как вы хотите организовать элементы на странице. Правильное использование этих значений поможет создать более интуитивно понятные и адаптивные интерфейсы.

3. justify-content

Свойство justify-content определяет распределение свободного пространства между flex-элементами вдоль главной оси. Оно может принимать следующие значения:

- flex-start элементы выравниваются по началу контейнера.
- flex-end элементы выравниваются по концу контейнера.
- center элементы центрируются.
- space-between пространство распределяется между элементами.
- space-around пространство распределяется вокруг элементов.





4. align-items

Свойство align-items определяет выравнивание flex-элементов по поперечной оси. Оно может принимать следующие значения:

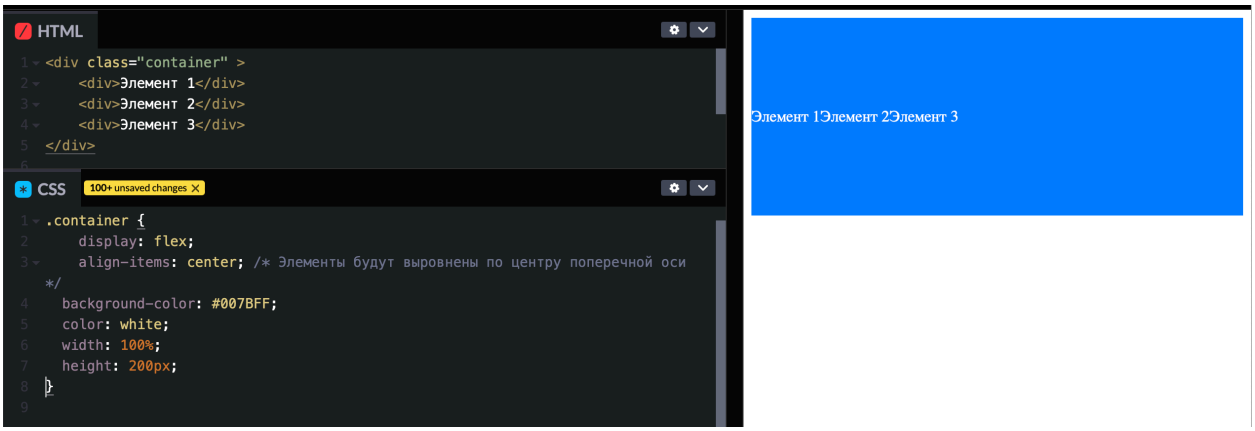
- flex-start элементы выравниваются по началу поперечной оси.

- flex-end элементы выравниваются по концу поперечной оси.

- center элементы центрируются по поперечной оси.

- baseline элементы выравниваются по базовой линии текста.

- stretch (по умолчанию) элементы растягиваются, чтобы заполнить контейнер.

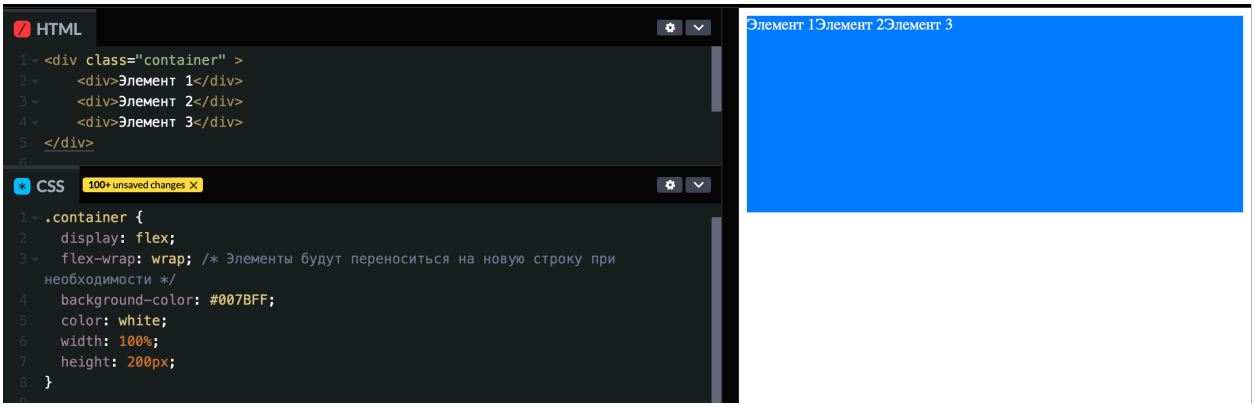


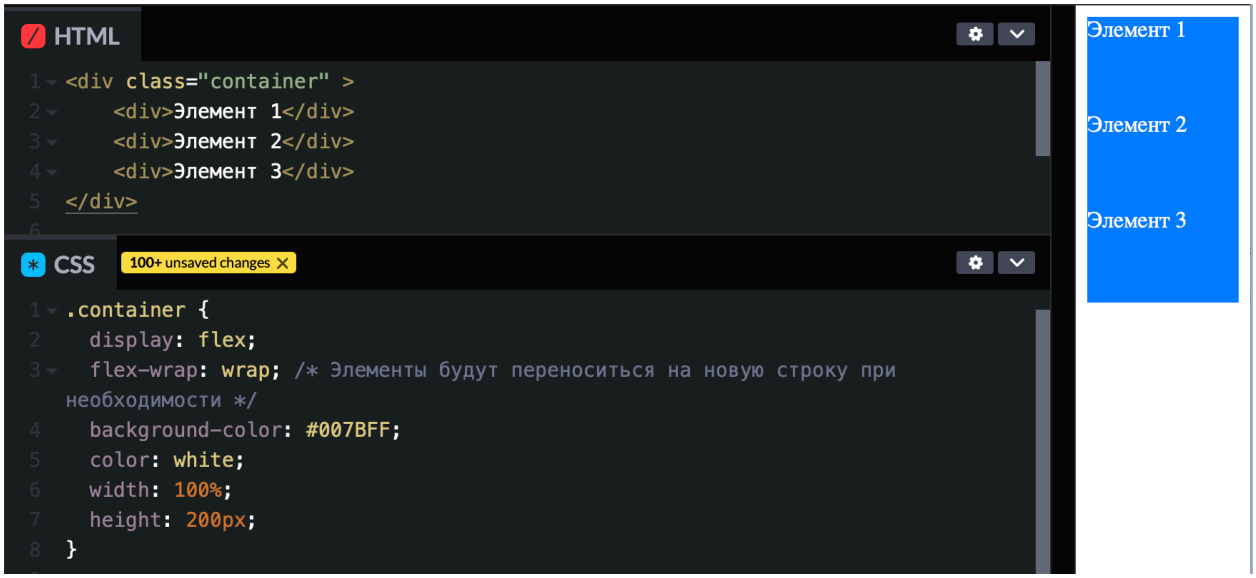
5. flex-wrap

Свойство flex-wrap определяет, должны ли flex-элементы переноситься на новую строку, если они не помещаются в контейнер. Оно может принимать следующие значения:

- nowrap (по умолчанию): все элементы остаются в одной строке- wrap: элементы переносятся на новую строку.

- wrap-reverse элементы переносятся на новую строку в обратном порядке.

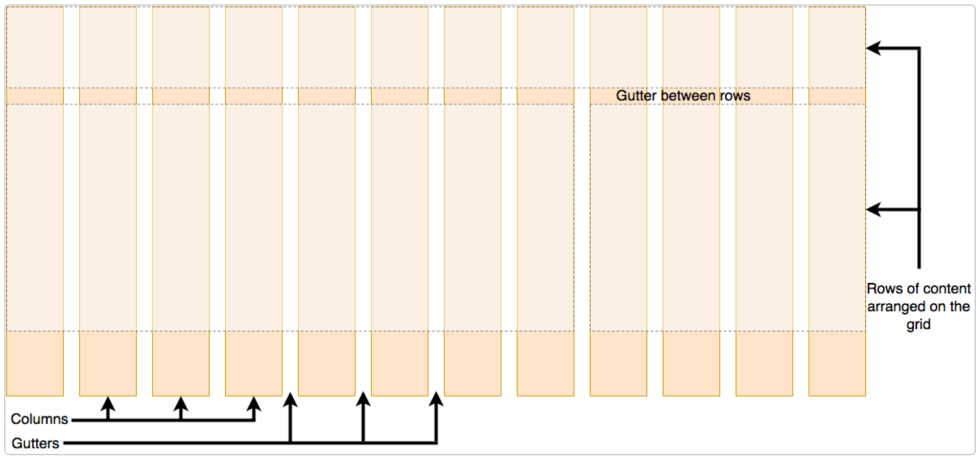




Grid Layout

Grid Layout — это двухмерная система компоновки, которая позволяет организовывать элементы на веб-странице как в строках, так и в столбцах. Это делает его идеальным инструментом для создания сложных интерфейсов, таких как сетки карточек, адаптивные галереи изображений и даже целые страницы.

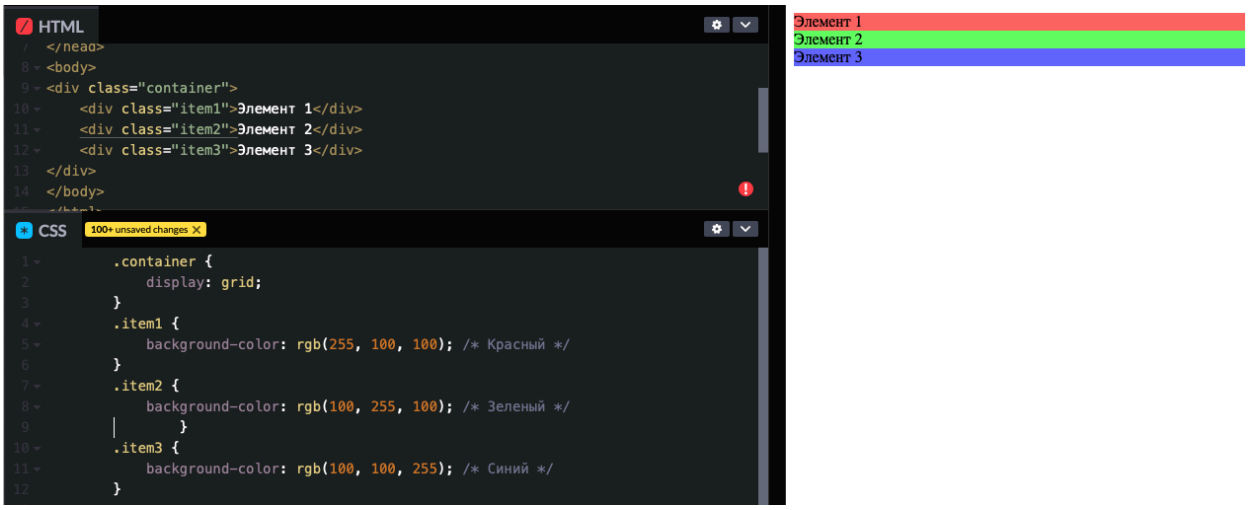
В сетке обычно будут **столбцы (columns)**, **строки (rows)**, а затем промежутки между каждой строкой и столбцом, обычно называемые **желобами (gutters)**.



Основные свойства Grid Layout

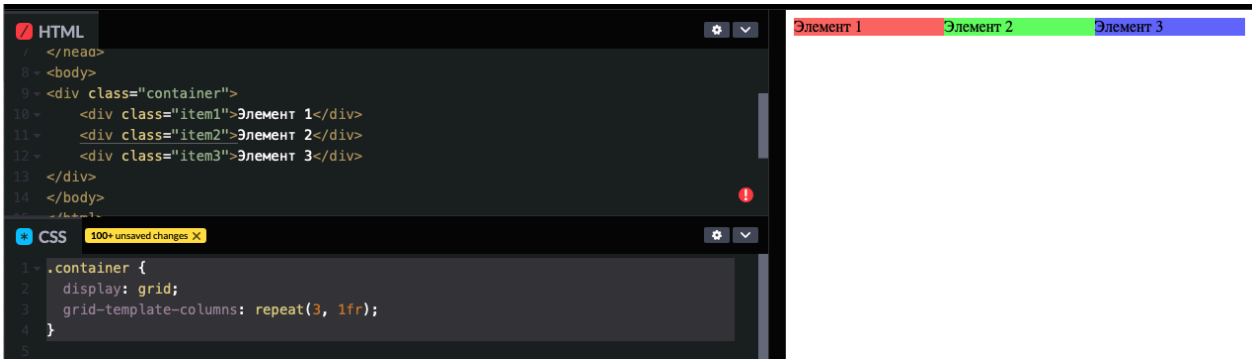
Теперь давайте рассмотрим основные свойства Grid Layout, которые помогут вам создать сетку.

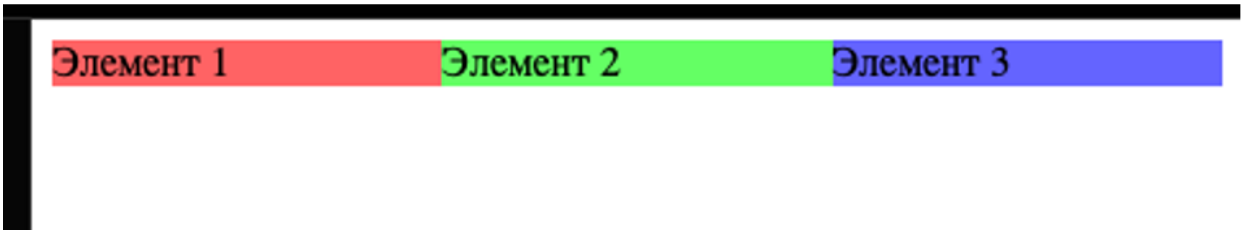
1. display: grid. Чтобы использовать Grid Layout, необходимо установить свойство display для контейнера:



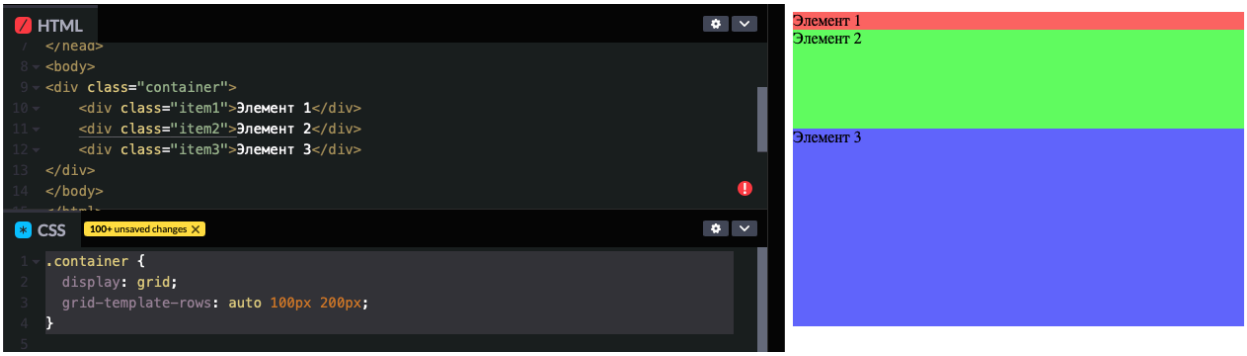
Это определяет контейнер как сетку, и все дочерние элементы становятся ячейками сетки.

2. grid-template-columns. С помощью свойства grid-template-columns можно задать количество и ширину столбцов:



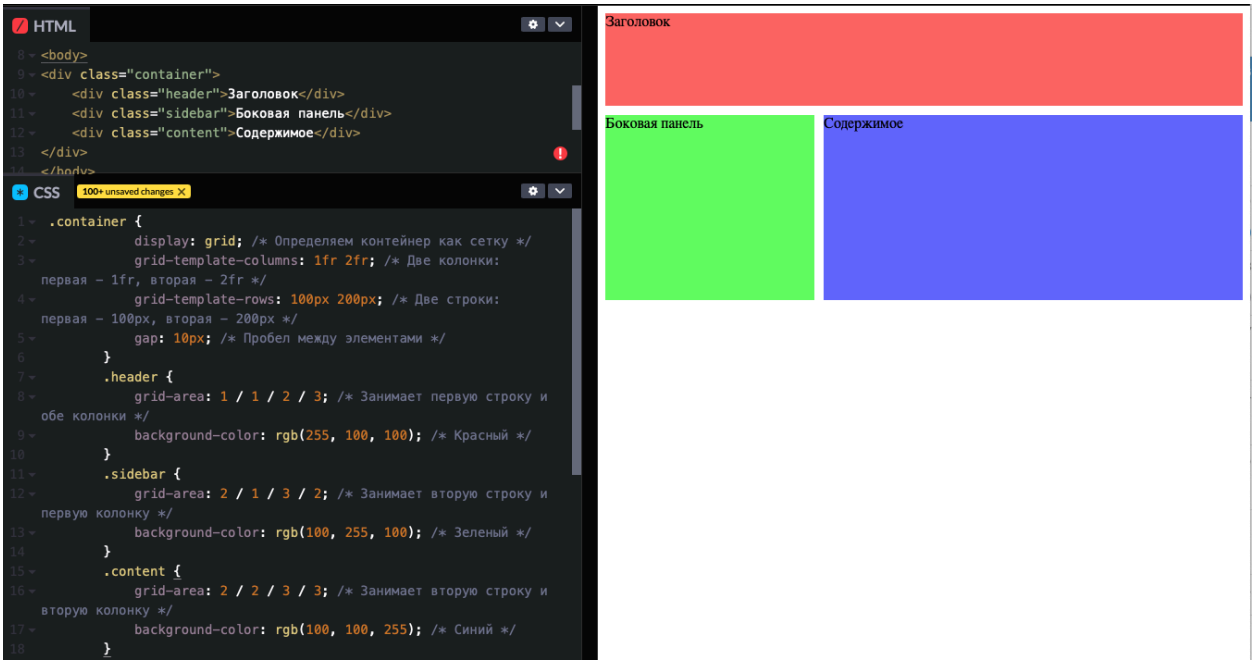


3. grid-template-rows. Аналогично свойству для столбцов, grid-template-rows задает высоту строк.



Здесь первая строка будет автоматически подстраиваться под содержимое, вторая строка будет иметь высоту 100 пикселей, а третья — 200 пикселей.

4. grid-area. Свойство grid-area позволяет задавать конкретные области для элементов сетки. Для иллюстрации этого свойства немного усложним наш пример.



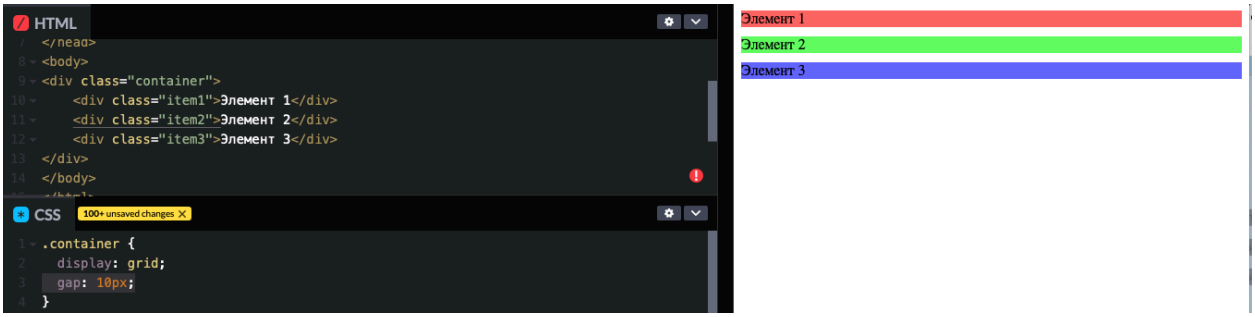
Контейнер:

- Устанавливаем сетку с двумя колонками и двумя строками.
- Первая колонка занимает одну долю (1fr), а вторая — две доли (2fr).

Элементы:

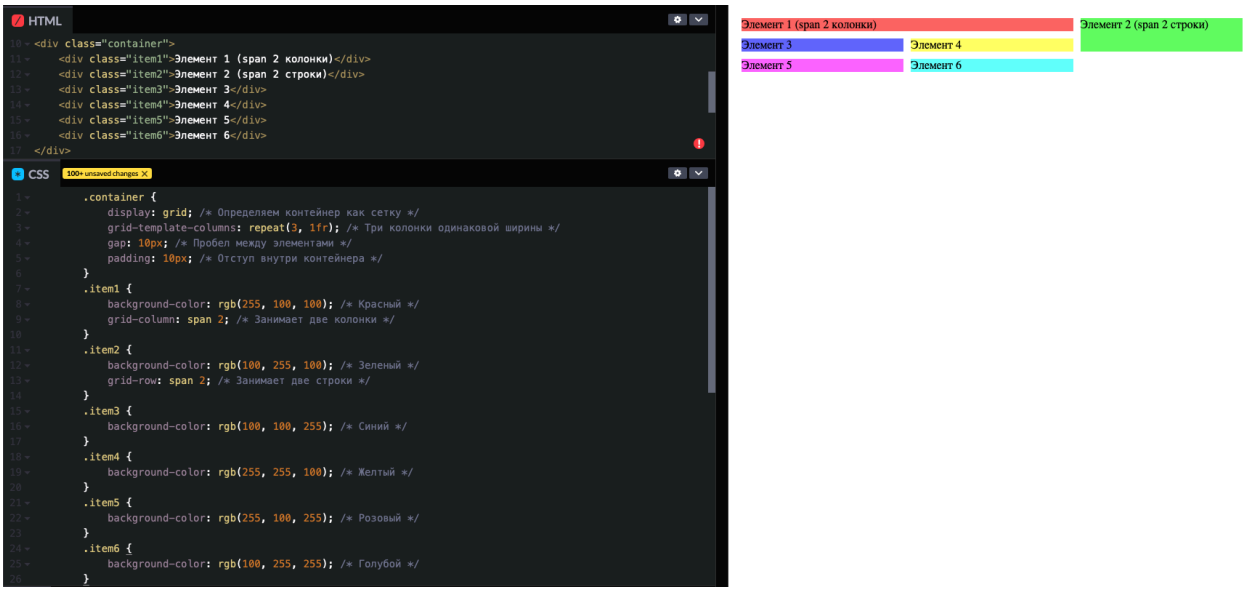
- .header
- grid-area: 1 / 1 / 2 / 3; — занимает первую строку (1) и обе колонки (от первой до третьей).
- .sidebar
- grid-area: 2 / 1 / 3 / 2; — занимает вторую строку (2) и первую колонку (1).
- .content
- grid-area: 2 / 2 / 3 / 3; — занимает вторую строку (2) и вторую колонку (2).

5. gap. Свойство gap (или grid-gap) позволяет задавать промежутки между ячейками:



Это добавит отступы как между строками, так и между столбцами.

Пример использования нескольких свойств:



1. display: grid Устанавливает контейнер .container как сетку.
2. grid-template-columns: repeat(3, 1fr) Создает три колонки одинаковой ширины.
3. gap: 10px Устанавливает промежуток между элементами сетки.
4. .item1 Занимает две колонки.
5. .item2 Занимает две строки.
6. Остальные элементы занимают одну ячейку.

Позиционирование элементов

Позиционирование элементов — это один из основных аспектов веб-разработки, который позволяет контролировать, как элементы отображаются на веб-странице. Понимание различных типов позиционирования и их особенностей имеет решающее значение для создания адаптивного и удобного пользовательского интерфейса.

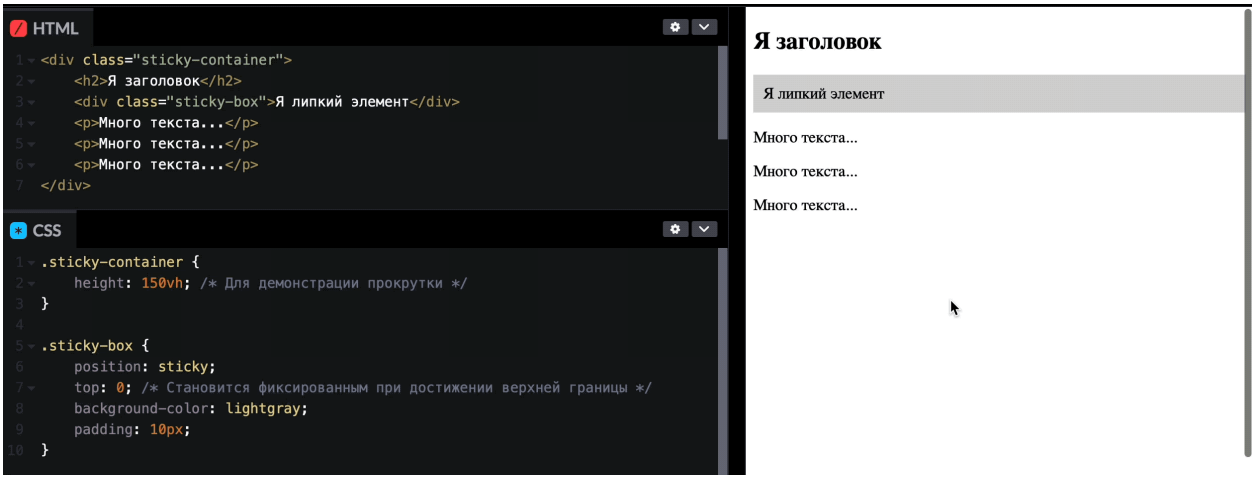
В CSS существует несколько типов позиционирования, каждый из которых имеет свои особенности и применяется в различных ситуациях. В данной лекции мы рассмотрим основные типы позиционирования: static, relative, absolute, fixed и sticky.

Элемент с position: fixed фиксируется относительно окна браузера, а не относительно родительского элемента. Это означает, что он остается на одном месте даже при прокрутке страницы.

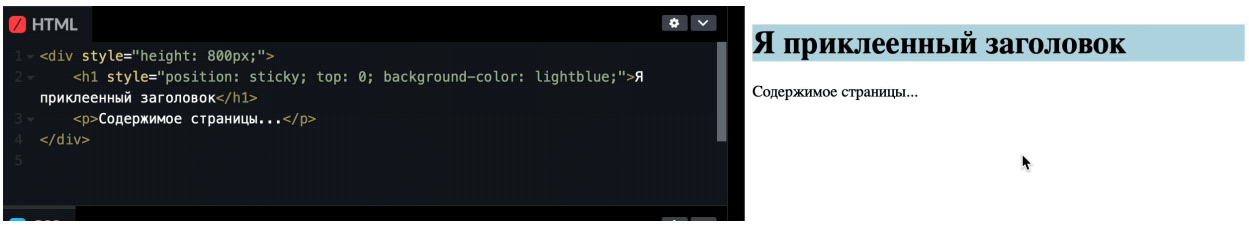
Пример

Элемент с position: sticky сочетает в себе свойства relative и fixed. Он ведет себя как обычный элемент до тех пор, пока не достигнет определенной позиции при прокрутке, после чего становится фиксированным.

Пример 1:



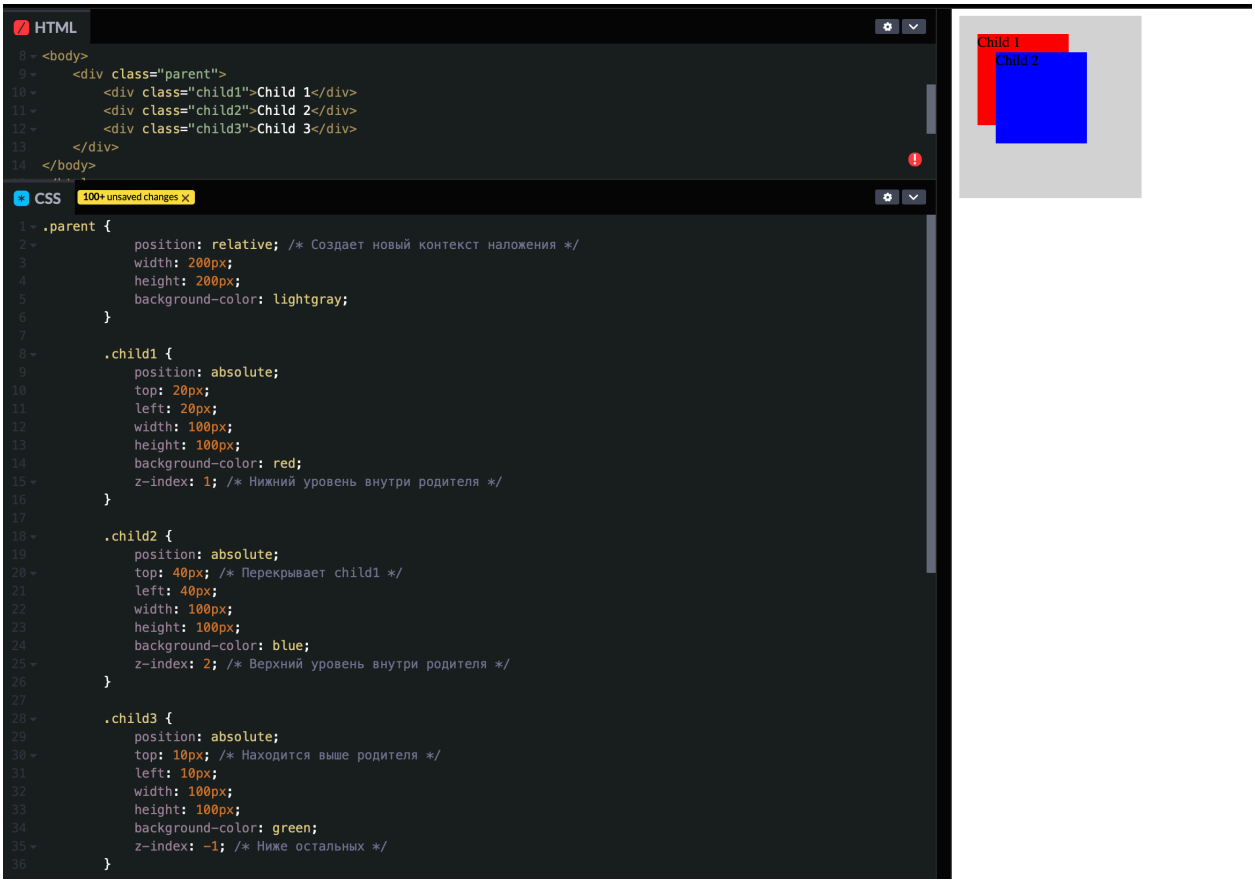
Пример 2:



Свойство z-index в CSS используется для управления порядком наложения элементов на странице. Оно определяет, какой элемент будет находиться выше или ниже других элементов, когда они перекрываются. z-index работает только для элементов с

установленным свойством position, отличным от static (например, relative, absolute, fixed или sticky).

Пример: Контекст наложения



В этом примере:

- child2 будет отображаться поверх child1, так как у него больший z-index.

- child3 будет находиться ниже обоих дочерних элементов, несмотря на то, что его положение выше в коде, потому что у него отрицательное значение z-index.

Полезные ссылки

- Flexbox на hcdev

https://hcdev.ru/learn/flex

- Flexbox на MDN

https://developer.mozilla.org/ru/docs/Web/CSS/CSS_Flexible_Box_Layout

- Grid на hcdev

https://hcdev.ru/learn/grid/

- Grid Layout на MDN

https://developer.mozilla.org/ru/docs/Web/CSS/CSS_Grid_Layout

Тренажеры

- Flexbox Froggy (игра для изучения Flexbox)

https://flexboxfroggy.com

- CSS Grid Garden (игра для изучения Grid Layout)

https://cssgridgarden.com/

Задание

1. Создание сетки с использованием Grid Layout:

- В теле сайта создайте новый элемент с классом grid-container.

- Внутри этого контейнера добавьте несколько элементов (например, заголовки, изображения, карточки товаров) и оформите их с помощью CSS Grid.

- Настройте количество колонок и строк, используя свойства grid-template-columns и grid-template-rows. Попробуйте создать сетку с 3 колонками и 2 строками.

2. Использование Flexbox для навигационного меню:

- Измените стили навигационного меню, чтобы оно использовало Flexbox.

- Убедитесь, что элементы меню равномерно распределены по горизонтали с помощью свойства justify-content.

- Добавьте отступы и выравнивание текста внутри элементов меню.

3. Стилизация карточек товаров с помощью Flexbox:

- Создайте контейнер для карточек товаров и примените к нему Flexbox.

- Настройте выравнивание карточек, чтобы они были равномерно распределены по горизонтали и вертикали.

- Убедитесь, что карточки адаптируются к размеру экрана.

4. Работа с позиционированием:

- Примените абсолютное позиционирование к одному из элементов внутри сетки (например, к изображению), чтобы оно перекрывало другие элементы.

Критерии оценивания

- Правильность использования Grid Layout и Flexbox. Элементы размещены корректно, используются соответствующие свойства CSS.

- Качество стилей. Элементы выглядят эстетично, соблюдены отступы и выравнивание.

- Использование позиционирования. Элементы правильно перекрываются и располагаются.

Форма сдачи

Студенты должны представить свой код в виде файлов .html .css, который можно открыть в браузере. Также важно прокомментировать код, чтобы объяснить, что делает каждый элемент.