

Вітаю всіх студентів на 6 лекції з вивчення CSS.

На цьому уроці ми почнемо вивчати одну з найновіших технологій — **Grid**.

Grid модуль для CSS був розроблений робочою групою CSS для того, щоб зробити створення шаблонів CSS максимально зручним. Він потрапив у рекомендації щодо офіційного впровадження у лютому 2017 року, а основні браузери розпочали його підтримку вже у березні 2017 року.

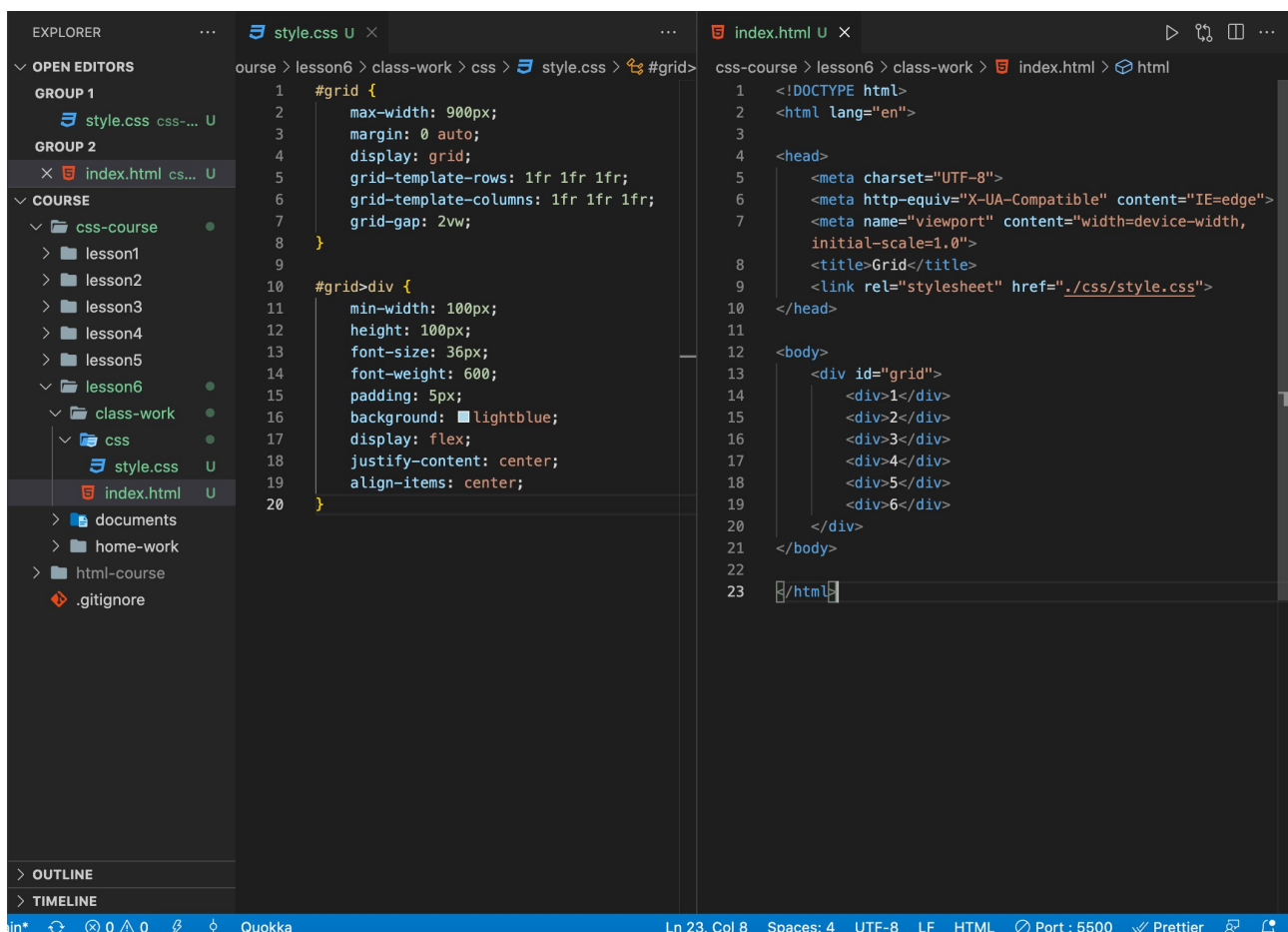
CSS **Grid** - це нова модель для створення шаблонів, оптимізована для створення двовимірних макетів. Вона ідеально підходить для: шаблонів сайтів, форм, галерей та всього, що вимагає точного та чуйного позиціонування.

Grid шаблон працює за системою сіток. Grid це набір горизонтальних і вертикальних ліній, що перетинаються, які створюють розмірність і позиціонують систему координат для контенту в самому grid-контейнері.

Щоб створити розмітку **Grid**, вам просто потрібно виставити елементу `display: grid`. Цей крок автоматично зробить всіх прямих нащадків цього елемента grid елементами. Після цього ви можете сміливо використовувати різноманітні grid властивості для вирівнювання розмірів та позиціонування елементів належним чином.

Давайте почнемо на практиці розбиратися з тим, як працює Grid.

Створюємо наступну структуру проекту та пишемо код, як на малюнку



The screenshot shows a code editor with two files open: `style.css` and `index.html`. The `style.css` file contains the following CSS code:

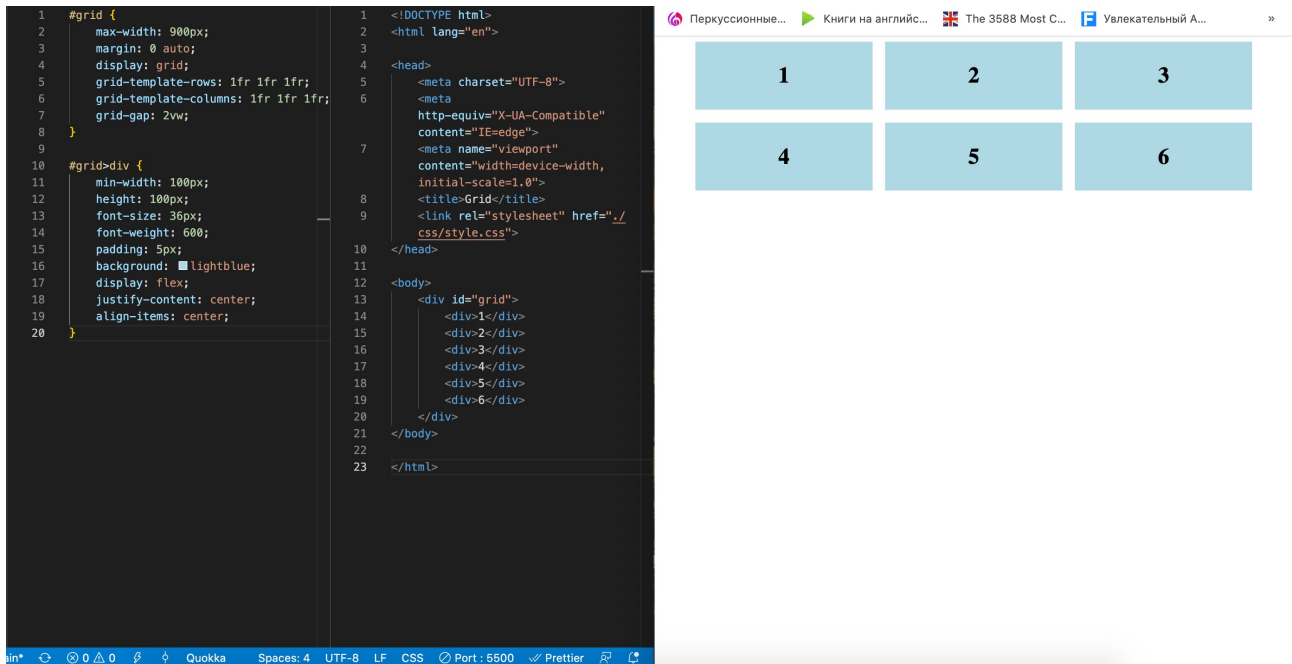
```
1 #grid {
2   max-width: 900px;
3   margin: 0 auto;
4   display: grid;
5   grid-template-rows: 1fr 1fr 1fr;
6   grid-template-columns: 1fr 1fr 1fr;
7   grid-gap: 2vw;
8 }
9
10 #grid>div {
11   min-width: 100px;
12   height: 100px;
13   font-size: 36px;
14   font-weight: 600;
15   padding: 5px;
16   background: lightblue;
17   display: flex;
18   justify-content: center;
19   align-items: center;
20 }
```

The `index.html` file contains the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width,
8     initial-scale=1.0">
9   <title>Grid</title>
10   <link rel="stylesheet" href="css/style.css">
11 </head>
12 <body>
13   <div id="grid">
14     <div>1</div>
15     <div>2</div>
16     <div>3</div>
17     <div>4</div>
18     <div>5</div>
19     <div>6</div>
20   </div>
21 </body>
22
23 </html>
```

The editor interface includes a file explorer on the left showing the project structure, and a status bar at the bottom indicating the current line and column (Ln 23, Col 8).

Отримуємо наступну картинку

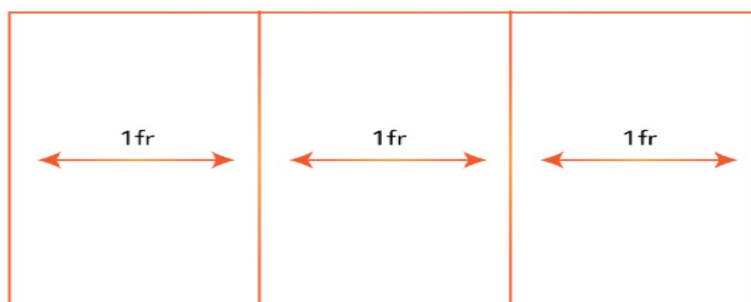


Ми отримали так званий grid контейнер. Це все, що потрібно для того, щоб створити grid. Тепер у нас є грід-контейнер та грід-елементи.

grid-template-rows: 1fr 1fr 1fr - вибудовує ряди у гріді, кожне значення становить розмір ряду.

grid-template-columns: 1fr 1fr 1fr - те саме, що й вище, тільки визначає колонки в грідах.

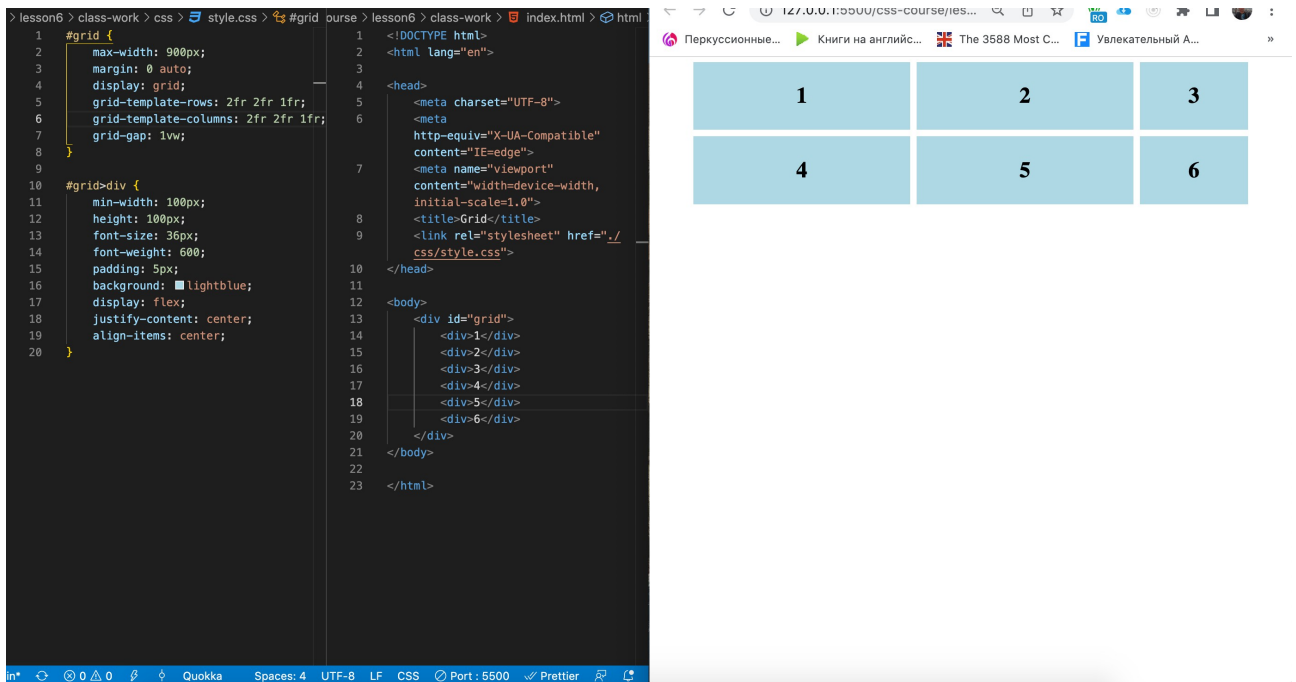
fr - це нова форма одиниці виміру, в основному асоційована з CSS Grid. Якщо ви вказуєте ширину 1fr, то ви можете далі додавати стільки елементів, скільки можливо і це вона про це подбає. Ширина кожного елемента буде рівномірно поділена серед дочірніх елементів.



grid-gap: 2vw - виставляє відступи між грід елементами.

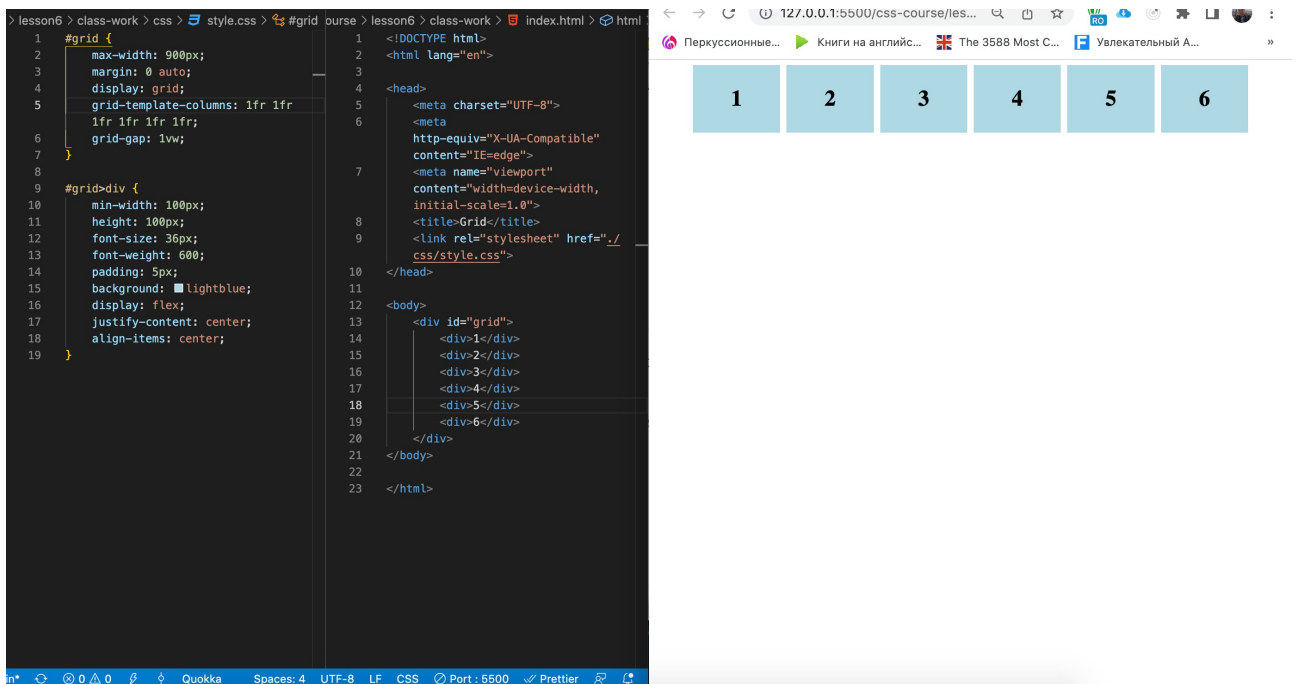
vw — viewport width - є відсотковим виміром ширини кореневого елемента, тобто відступи будуть змінюватися пропорційно ширині екрана.

Давайте змінимо значення властивостей grid і подивимось, як зміниться малюнок.



Ми бачимо, що блоки пропорційно розподілилися по ширині екрана та зменшилися відступи між ними.

Давайте зробимо всі блоки однаковими і розмістимо їх в один рядок

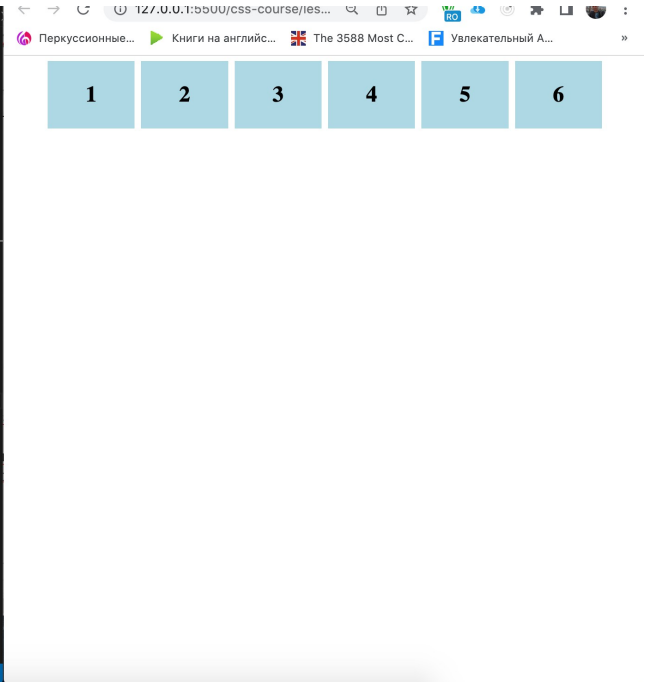


Функція *repeat()*.

Ви можете використовувати функцію `repeat()` для повторюваних об'яв значення розміру елемента. Давайте скоротимо наш код, використовуючи функцію `repeat()`

```
lesson6 > class-work > css > style.css > #grid
1 #grid {
2   max-width: 900px;
3   margin: 0 auto;
4   display: grid;
5   grid-template-columns: repeat
6     (6, 1fr);
7   grid-gap: 1vw;
8 }
9 #grid>div {
10  min-width: 100px;
11  height: 100px;
12  font-size: 36px;
13  font-weight: 600;
14  padding: 5px;
15  background: lightblue;
16  display: flex;
17  justify-content: center;
18  align-items: center;
19 }
```

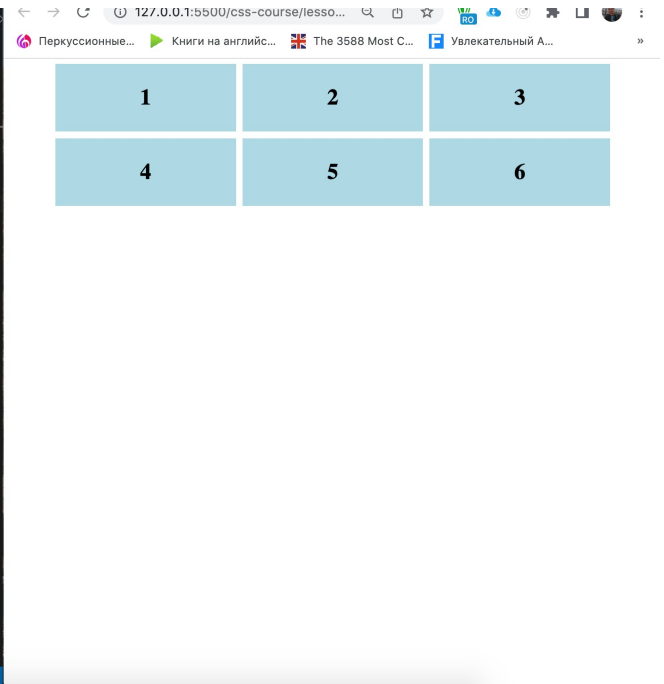
```
lesson6 > class-work > index.html > html
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta
7     http-equiv="X-UA-Compatible"
8     content="IE=edge">
9   <meta name="viewport"
10     content="width=device-width,
11     initial-scale=1.0">
12   <title>Grid</title>
13   <link rel="stylesheet" href="./
14     css/style.css">
15 </head>
16 <body>
17   <div id="grid">
18     <div>1</div>
19     <div>2</div>
20     <div>3</div>
21     <div>4</div>
22     <div>5</div>
23     <div>6</div>
24   </div>
25 </body>
26 </html>
```



Як ми бачимо, нічого не змінилося. Першим аргументом функції ми задаємо кількість елементів, другим - їх розмір відносно батьківського блоку.
Давайте змінимо кількість на 3 елементи

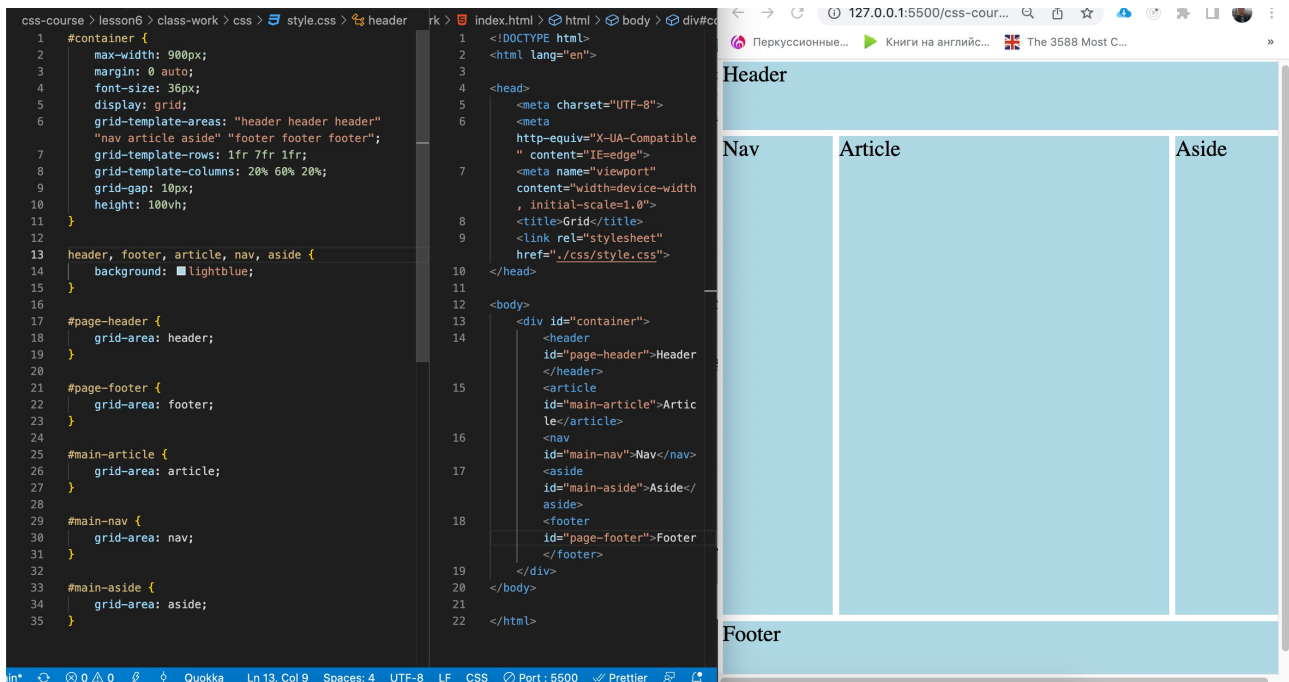
```
css-course > lesson6 > class-work > css > style.x
1 #grid {
2   max-width: 900px;
3   margin: 0 auto;
4   display: grid;
5   grid-template-columns: repeat
6     (3, 1fr);
7   grid-gap: 1vw;
8 }
9 #grid>div {
10  min-width: 100px;
11  height: 100px;
12  font-size: 36px;
13  font-weight: 600;
14  padding: 5px;
15  background: lightblue;
16  display: flex;
17  justify-content: center;
18  align-items: center;
19 }
```

```
css-course > lesson6 > class-work > index.html
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta
7     http-equiv="X-UA-Compatible"
8     content="IE=edge">
9   <meta name="viewport"
10     content="width=device-width,
11     initial-scale=1.0">
12   <title>Grid</title>
13   <link rel="stylesheet" href="./
14     css/style.css">
15 </head>
16 <body>
17   <div id="grid">
18     <div>1</div>
19     <div>2</div>
20     <div>3</div>
21     <div>4</div>
22     <div>5</div>
23     <div>6</div>
24   </div>
25 </body>
26 </html>
```



Ми бачимо, що елементи рівномірно розмістилися по 3 у 2 рядки.

Давайте створимо стандартний шаблон сторінки використовуючи grid і семантичні елементи та поекспериментуємо з ним.



Що ми зробили. По-перше ми зробили шаблон, який виглядає таким чином

"header header header"
"nav article aside"
"footer footer footer"

- 1-й рядок займають 3 комірки header
- 2-й рядок складається з 3-х комірок: nav article aside
- 3-й рядок займають 3 комірки footer

grid-template-rows: 1fr 7fr 1fr; - висота кожного наступного рядка відносно загальної висоти сторінки(зверху-вниз).

grid-template-columns: 20% 60% 20%; - ширина кожної наступної колонки відносно загальної ширини сторінки(зліва-направо).

grid-gap: 10px; - відступ між елементами.

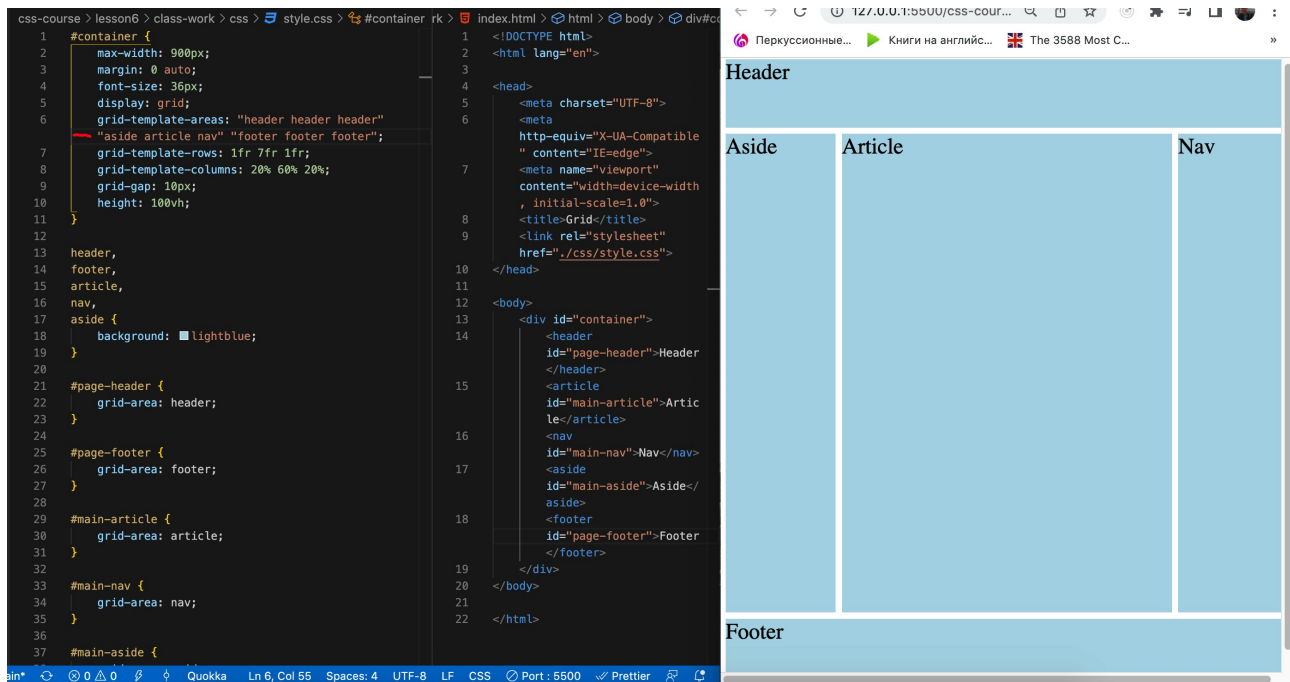
height: 100vh;

vh — viewport height - є відсотковим виміром висоти кореневого елемента. 100vh — 100% висоти сторінки.

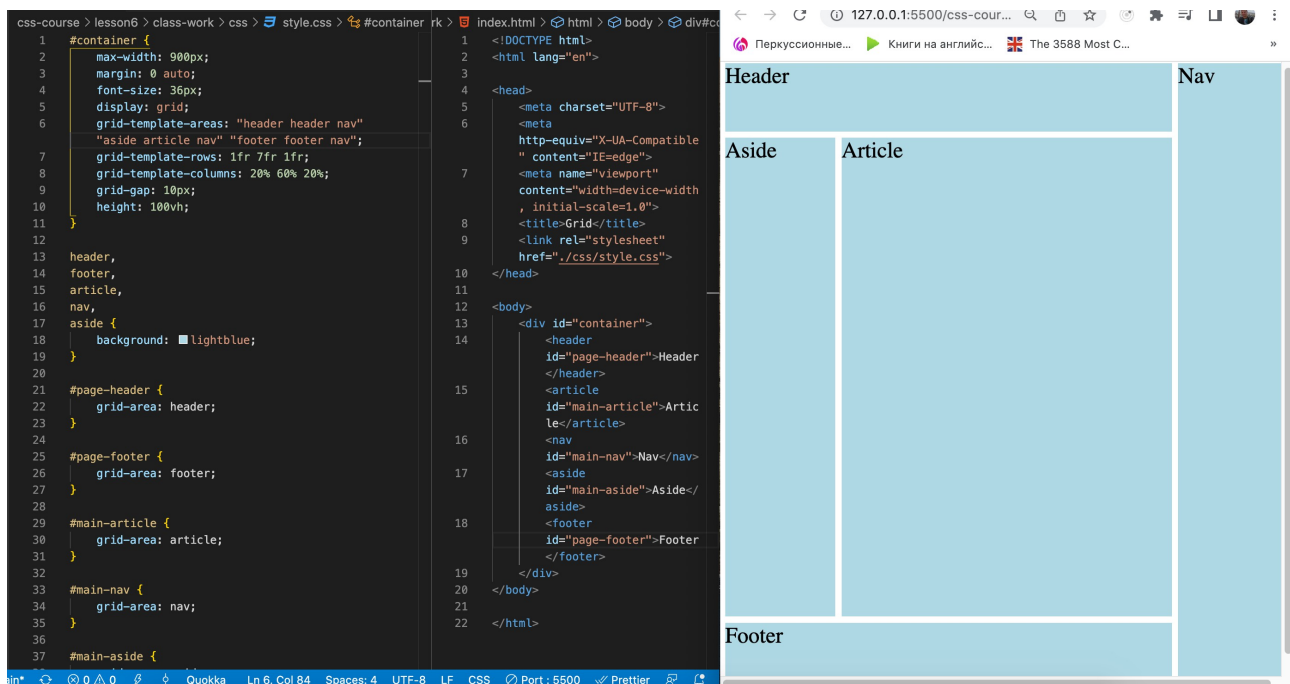
Далі ми задали однаковий колір для всіх елементів. І нарешті:

grid-area: name; - прив'язали ім'я елемента до відповідного блоку з відповідним id.

Давайте трохи поекспериментуємо і змінимо місцями **nav** та **aside**

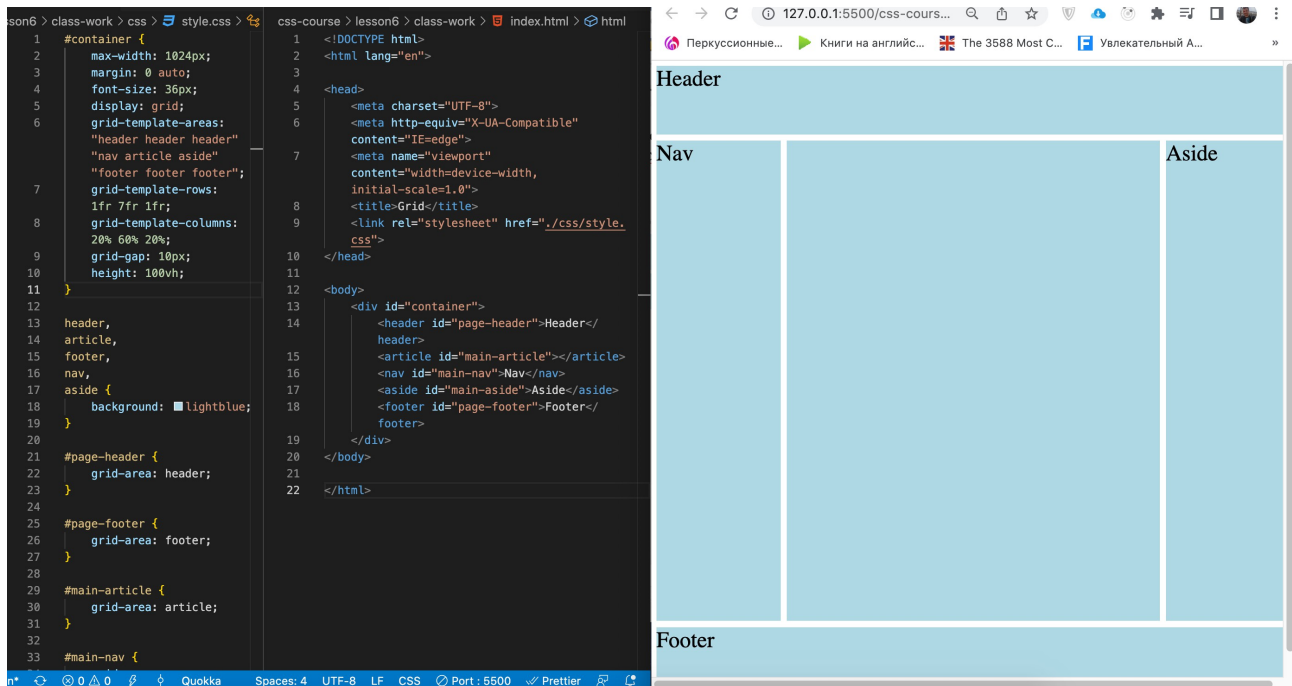


Тепер давайте зробимо **nav** на всю висоту екрану

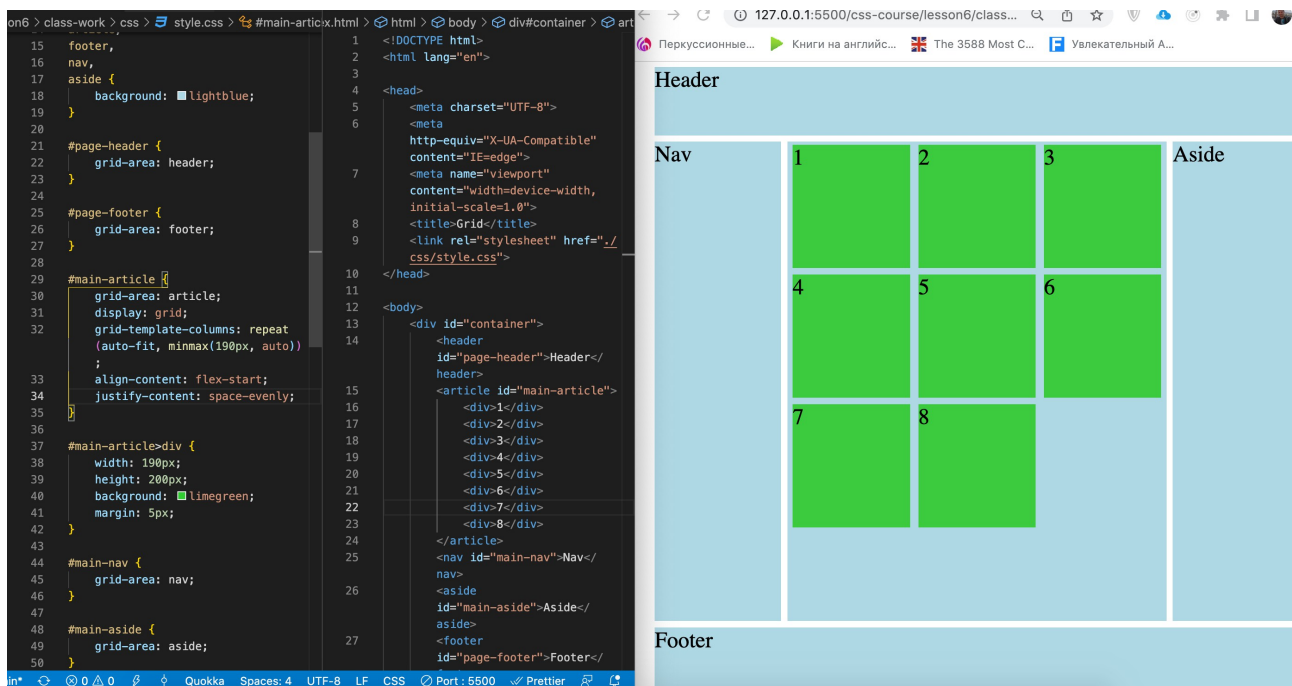


Як бачите, наскільки просто можна змінювати розташування елементів використовуючи шаблон grid.

Давайте повернемо все на свої місця і зробимо ширину контейнера **#container** 1024px.



І ще, давайте додамо до блоку article властивість `display: grid` а також додамо 8 однакових блоків і стилізуємо їх.



Поясню, що ми зробили.

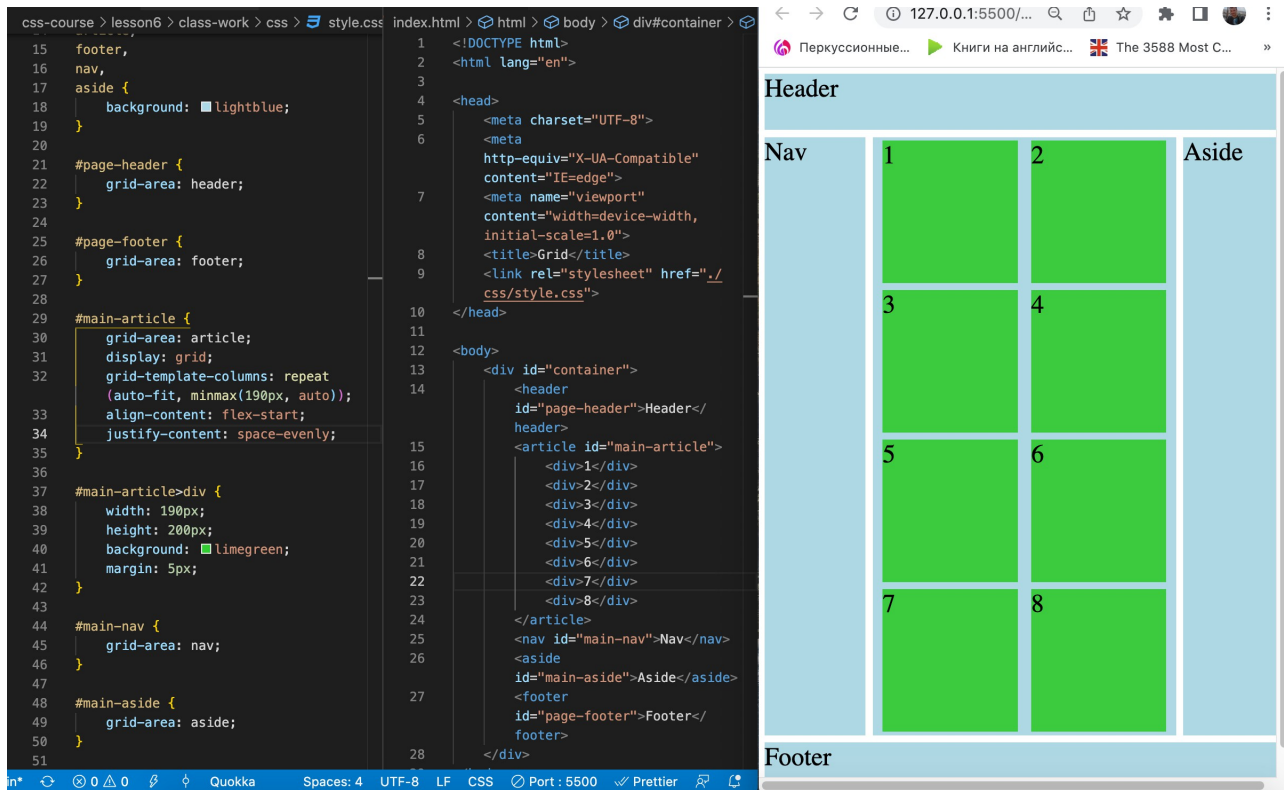
Функція `repeat()` повторює визначення треку кількість разів, задане першим параметром.

Використання `auto-fill` змусить треки повторюватися таку кількість разів, доки вони не заповнять контейнер.

`minmax()` це CSS функція, що визначає діапазон розмірів, який більший або дорівнює `min` і менше або дорівнює `max`.

Колонкам задається мінімальний розмір 190px і вони автоматично заповнюють батьківський блок. Далі ми вирівнюємо контент по верхній частині блоку та рівномірно по ширині. Ширину і висоту блоків `div` ми задали фіксованими — 190 та 200px.

Таким чином ми отримали адаптивний блок article. Якщо зменшити ширину екрану - отримаємо таку картинку



Залишаємо все як є і переходимо до наступного уроку.