Вітаю всіх на 3-й лекції з вивчення CSS. Як завжди на початку трохи теорії.

Для роботи з HTML сторінкою та CSS використовуються 2 основних атрибути HTML-елементів — це "id" та "class". Поки ми будемо використовувати в основному атрибут "class", але потрібно, щоб у майбутньому ви розуміли коли який атрибут використовувати.

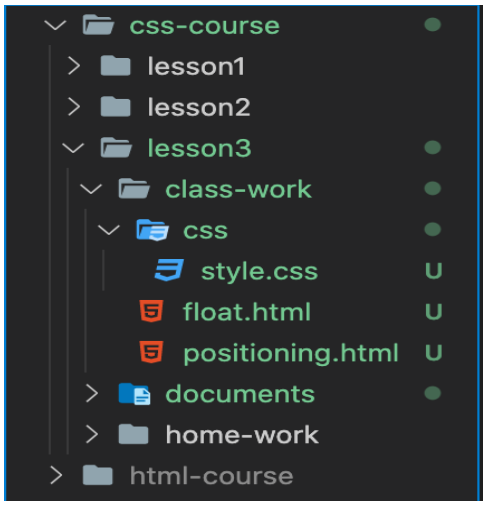
id - унікальне власне ім'я елемента на сторінці, тобто на сторінці не повинно бути декілька елементів з одним ID. Наприклад, блоку з шапкою сайту можна дати id="title".

class - вільна ознака, яка дається зазвичай декільком елементам, щоб відрізняти їх від інших. Наприклад, малюнкам, які просто супроводжують текст, можна дати class="img-decor", а малюнкам, які важливі для розуміння тексту — class="img-content".

Давайте, щоб запам'ятати синтаксис атрибуту id, будемо використовувати в цьому уроці тільки його.

Головною темою нашого уроку буде вивчення відносного розташування елементів сторінки один щодо одного, так зване ***позиціювання***. Паралельно ми вивчатимемо нові CSS властивості. Почнемо.

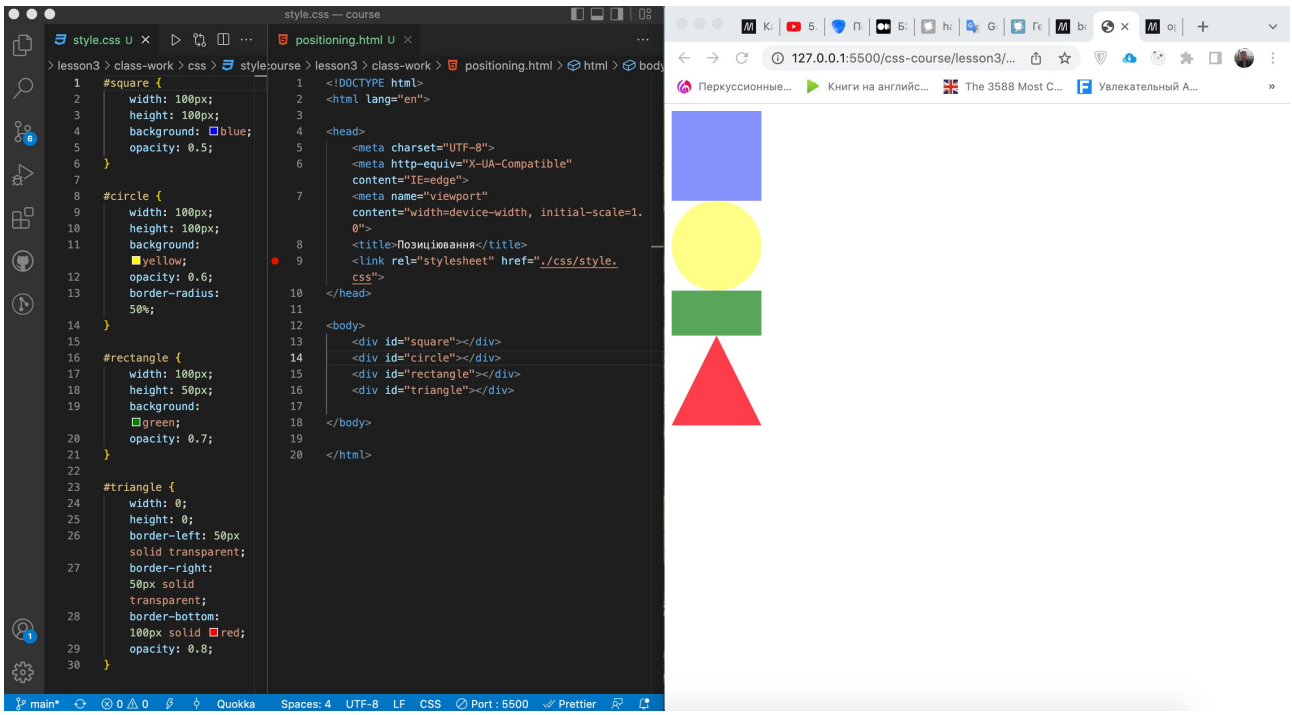
Створюємо наступну структуру нашого уроку:



Створіть для зручності 2 окремі папки - class-work(у цій папці ми будемо робити нашу спільну класну роботу) та home-work з аналогічною структурою для домашньої роботи. Файли float.html та positioning.html знаходяться у папці class-work.

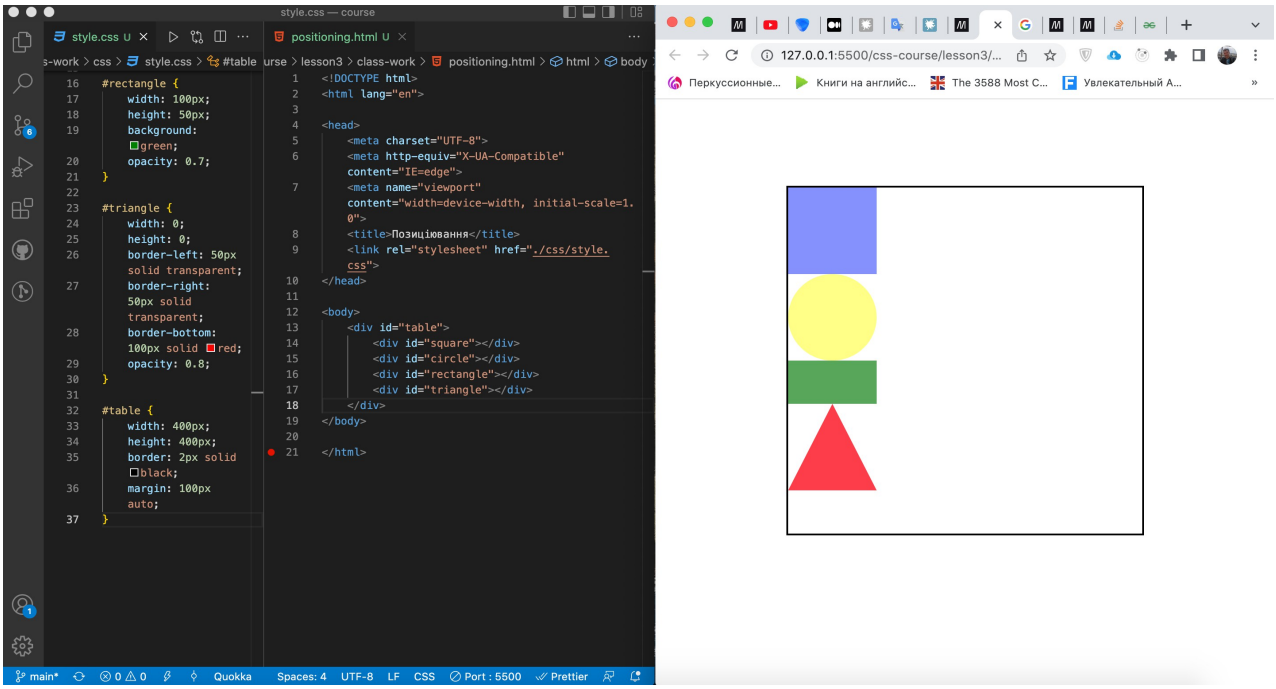
Напочатку давайте згадаємо геометричні фігури з початкової школи і намалюємо на сторінці квадрат, коло, прямокутник і трикутник. Першу спільну класну роботу робитимемо у файлі positioning.html.
Усі наші фігури будуть блоками з відповідними **id**. Для того, щоб швидко створити блок з відповідним id — набираємо, наприклад #square та натискаємо Enter.

Давайте створимо всі фігури з однаковою шириною 100px. Отримуємо таку картинку:



Давайте розберемось, що ми зробили. З квадратом, думаю, все зрозуміло, окрім властивості ***opacity***. Відразу читаємо тут: https://developer.mozilla.org/en-US/docs/Web/CSS/opacity Коло ми отримали, додавши властивість **border-radius**: 50%; (тобто половину ширини квадрата). Можете погратися змінюючи значення. З прямокутником також все зрозуміло. Давайте перейдемо до трикутника. Ми його створюємо використовуючи властовості border(знизу 100px, праворуч та ліворуч 50px(в сумі = 100px) - оскільки ми поставили ширину рівною 0, таким чином ми отримуємо звуження догори елемента червоного кольору). Значення **transparent** означає — прозорий. Видаліть значення transparent для властивостей border-left та border-left. Ми побачимо квадрат, який складається з 3-х трикутників. Робимо чорні трикутники прозорими — додаємо властивості transparent і отримуємо наш червоний трикутник. Йдемо далі.

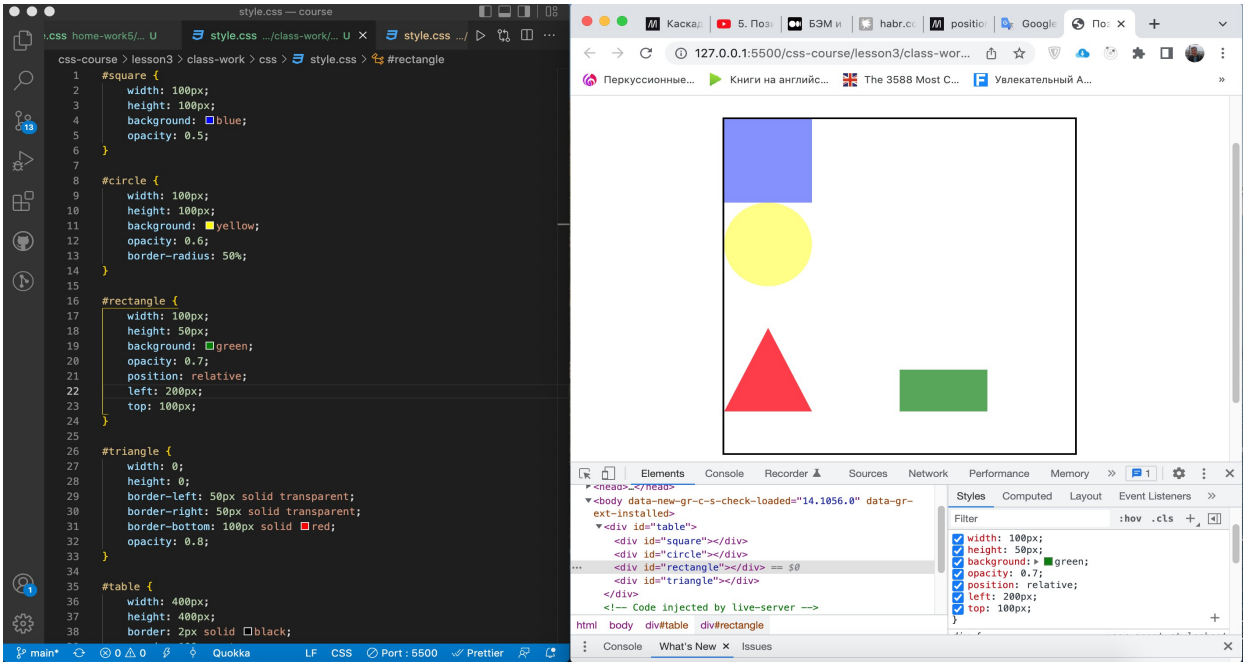
Давайте зробимо стіл (id="table") розміром 400*400px, обведемо його рамкою, вирівняємо по центру з відступами зверху та знизу 100px та покладемо на нього наші фігури :-).



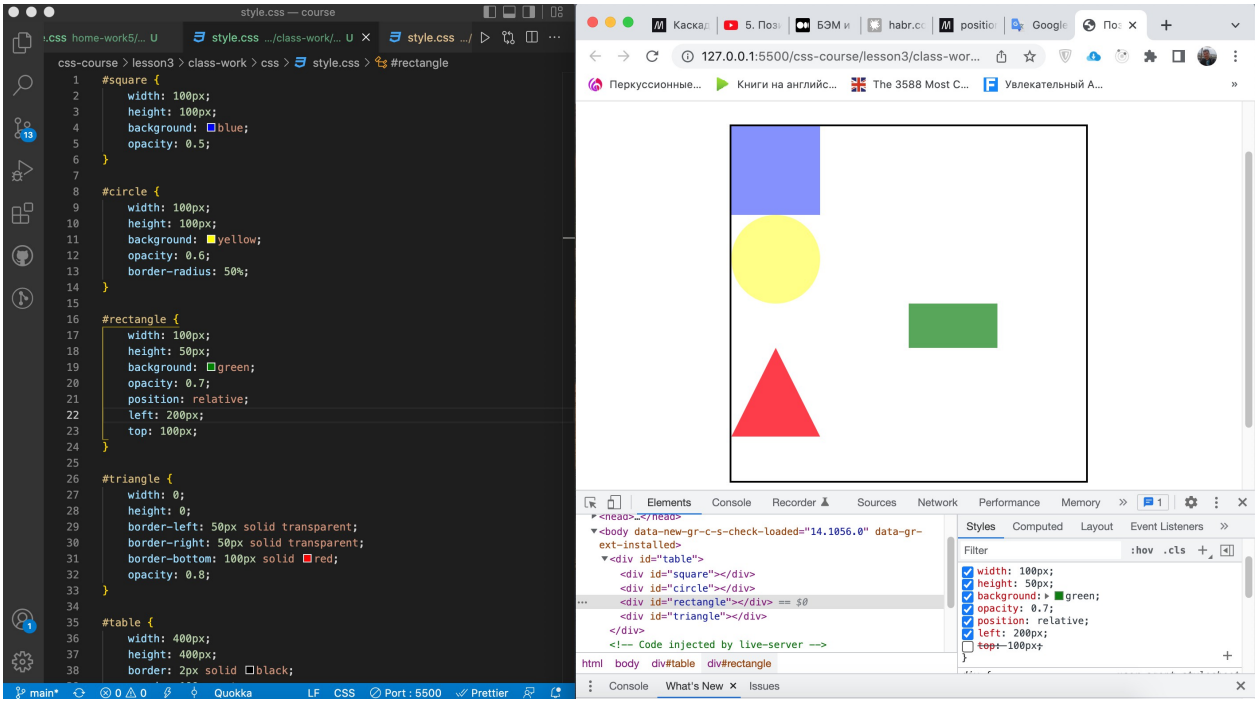
Перед тим, як спробувати почати рухати наші елементи, давайте розберемо таку властивість, як ***position***. Посилання на додаткові матеріали: https://developer.mozilla.org/en-US/docs/Web/CSS/position

Основні властивості управління позицією: ***top, right, bottom, left***. Важливо пам'ятати, що елемент зміщується ***від*** того боку, який ми вказали.

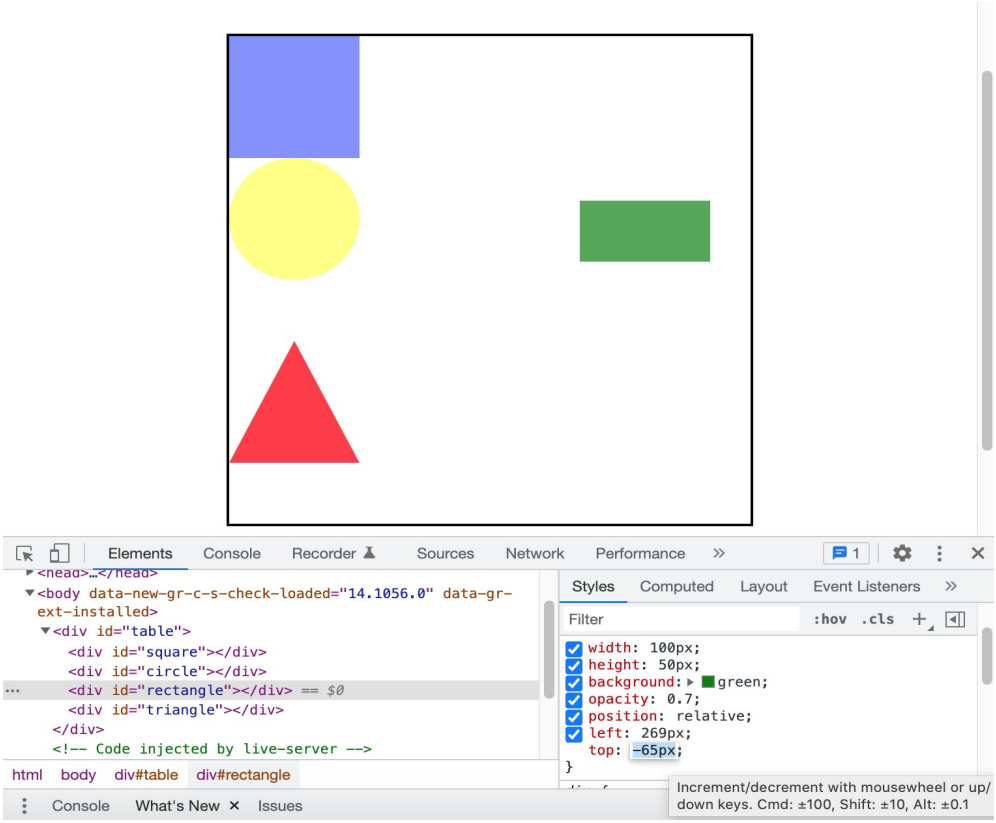
Давайте почнемо з position: relative; - наш елемент буде зміщуватися відносно батьківського елемента на ті значення, які ми вказали. Давайте задамо нашому прямокутнику властивості left: 200px; та top: 100px; . Ось що ми отримаємо в результаті



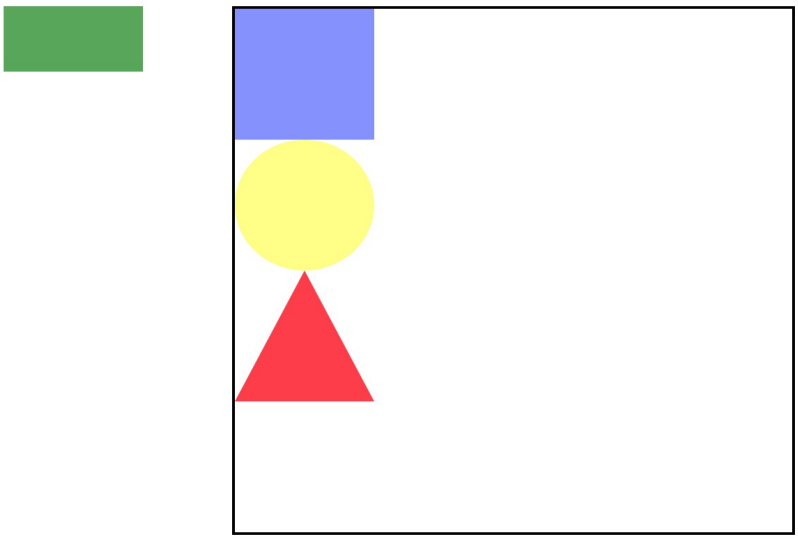
Ми бачимо, що наш прямокутник змістився від лівого краю столу на 200px і верхнього краю на 100px. Давайте у панелі розробника видалимо галку check властивості top. Ми бачимо, що прямокутник піднявся вгору на 100px.



Також зверніть увагу, що ми можемо в панелі розробника виділити потрібне нам значення і за допомогою курсора вгору або вниз змінювати його і зразу бачити, як елемент відображується на сторінці, причому значення може бути як позитивним, так і негативним.

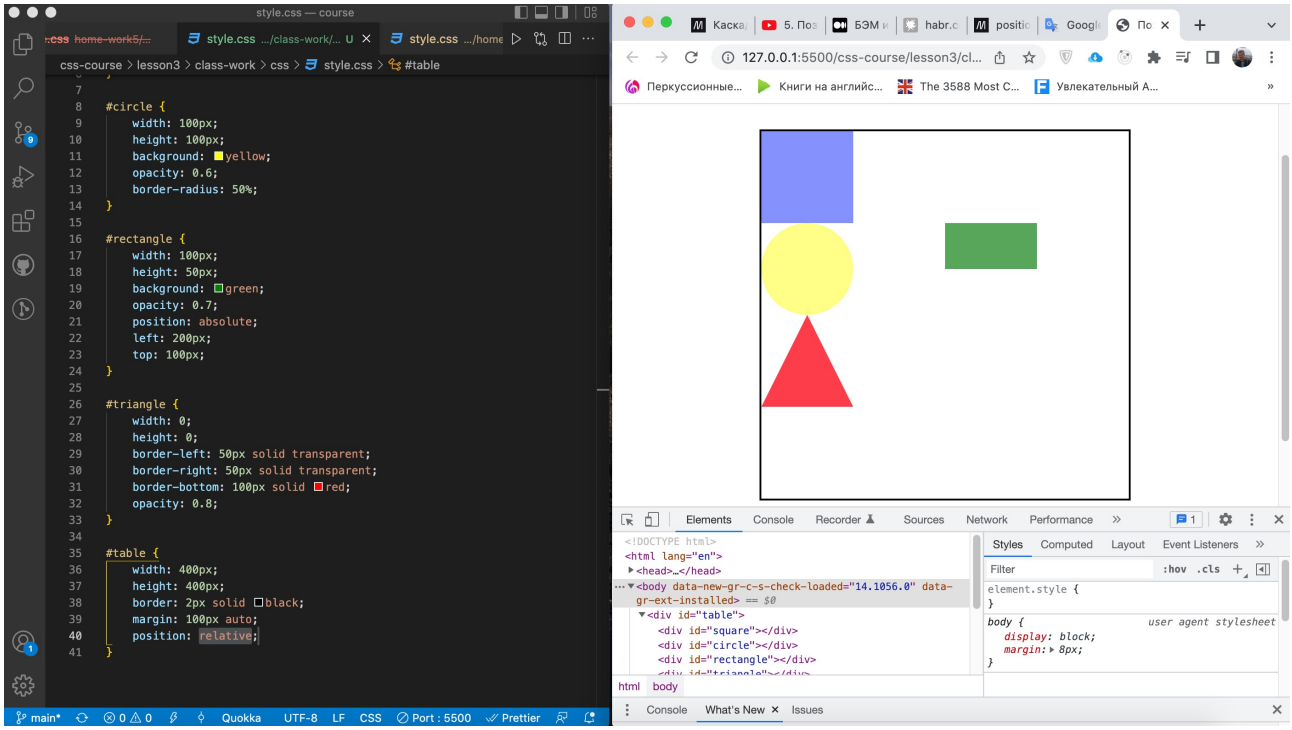


Давайте подивимось як працює властивість position: absolute;. Змінюємо значення position для прямокутника на absolute.



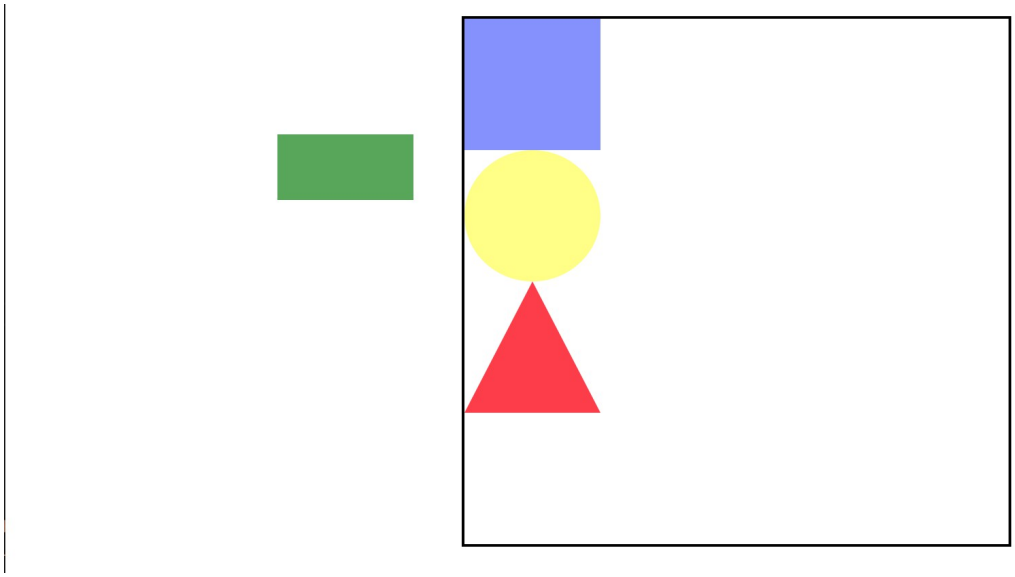
Ми бачимо, що елемент змістився відносно нашого body, так сталося тому, що в жодного з батьківських елементів немає властивості position. Для того, щоб позиція нашого елемента

змінювалась відносно table, потрібно нашому батьківському елементу table надати властивість relative. Отримаємо такий результат



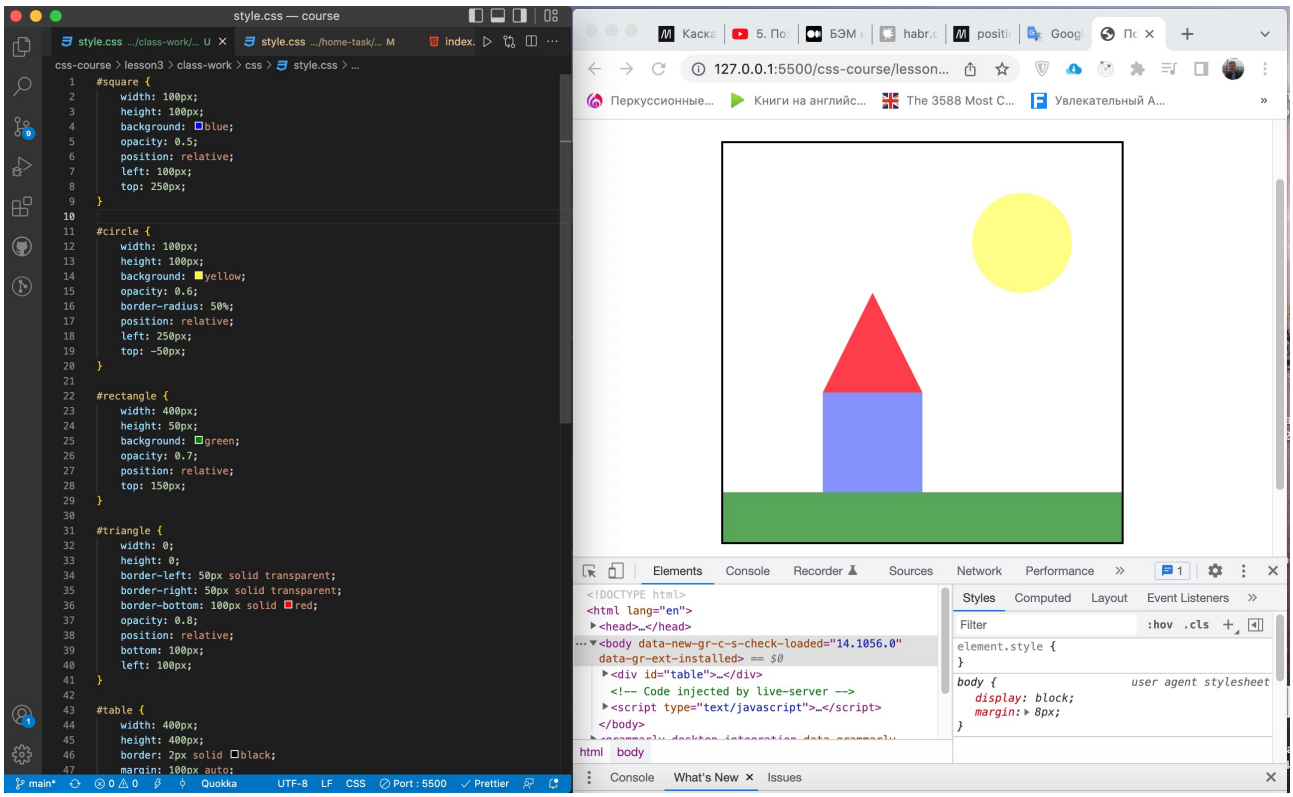
Як ми бачимо, цей елемент перестав існувати для нашого трикутника і змістився відносно нашого блока table.

І останнє. Давайте присвоємо нашому прямокутнику position: fixed; . Ми бачимо, що наш елемент став зафіксованим в одному положенні, не залежно від контенту сторінки.



Ця властивість використовується, коли потрібно закріпити плаваючі кнопки вгору-вниз, або плаваюче меню "Зворотній зв'язок". Давайте видалимо всі наші position, left і right і повернемо фігури на початкову позицію

Останнє що ми з вами зробимо, давайте розширимо прямокутник до 400px і за допомогою властивостей та панелі інструментів зробимо простий малюнок.



Анімація

Давайте закоментуємо ті зміщення, які ми ставили нашим фігурам і повернемо їх в початкове положення та зробимо просту анімацію, як на відео *animation.* Які CSS властивості ми будемо використовувати в нашій анімації:

animation-name: move-rectangle; - назва анімації
animation-duration: 2s; - тривалість анімації(секунди)
animation-iteration-count: 1; - кількість повторів анімації
animation-fill-mode: both; - визначає, як слід застосовувати стилі до об'єкта анімації до і після виконання(фіксує вихідний та кінцевий стан).

Детальніше тут: https://developer.mozilla.org/en-US/docs/Web/CSS/animation

Робити анімацію ми будемо використовуючи правило **@keyframes**

Найпростіший варіант, коли у нас є лише два ключові кадри — вихідний та кінцевий стан.

У такому разі @keyframes запишеться у такому вигляді:

```
@keyframes box {
 from { left: 0; }
 to { left: 300px; }
}
```

В даному випадку ми даємо нашому набору ім'я box, воно потім буде задіяне у властивості animation-name і визначаємо, що елемент змінюватиме значення властивості left від 0 до 300 px. Замість ключових слів from і to можна використовувати відповідно 0% і 100%.

Ключові кадри не обов'язково мають починатися з 0% та закінчуватися 100%. Анімація тоді відбуватиметься не одразу.

```
@keyframes box {
 50% { left: 0; }
 90% { left: 300px; }
}
```

Давайте зробимо анімацію, як на відео animation.

- Першим починає рухатися прямокутник. Давайте зробимо затримку, щоб ми могли побачити початкову позицію елемента. Також змінимо відступ зверху та ширину прямокутника.

```
#rectangle {
    width: 100px;
    height: 50px;
    background: green;
    opacity: 0.7;
    position: relative;
    /* top: 150px; */
    animation-name: move-rectangle;
    animation-duration: 2s;
    animation-iteration-count: 1;
    animation-fill-mode: both;
}

@keyframes move-rectangle {
    50% {
        top: 0px;
        width: 100px;
    }
    100% {
        top: 150px;
        width: 400px;
    }
}
```

- Наступним починає рухатися квадрат

```css
#square {
    width: 100px;
    height: 100px;
    background: blue;
    opacity: 0.5;
    position: relative;
    /* left: 100px;
    top: 250px; */
    animation-name: move-square;
    animation-duration: 3s;
    animation-iteration-count: 1;
    animation-fill-mode: both;
}

@keyframes move-square {
    50% {
        left: 0px;
        top: 0px;
    }
    100% {
        left: 100px;
        top: 250px;
    }
}
```

– Далі — трикутник

```css
#triangle {
    width: 0;
    height: 0;
    border-left: 50px solid transparent;
    border-right: 50px solid transparent;
    border-bottom: 100px solid red;
    opacity: 0.8;
    position: relative;
    /* bottom: 100px;
    left: 100px; */
    animation-name: move-triangle;
    animation-duration: 4s;
    animation-iteration-count: 1;
    animation-fill-mode: both;
}

@keyframes move-triangle {
    50% {
        bottom: 0px;
        left: 0px;
    }
    100% {
        bottom: 100px;
        left: 100px;
    }
}
```

- І останній елемент — коло

```css
#circle {
    width: 100px;
    height: 100px;
    background: yellow;
    opacity: 0.6;
    border-radius: 50%;
    position: relative;
    /* left: 250px;
    top: -50px; */
    animation-name: move-circle;
    animation-duration: 8s;
    animation-iteration-count: 1;
    animation-fill-mode: both;
}

@keyframes move-circle {
    50% {
        left: 0px;
        top: 0px;
    }
    100% {
        left: 250px;
        top: -50px;
    }
}
```

Перезавантажуємо сторінку і ми повинні отримати анімацію як на відео.

До зустрічі на наступному уроці.