Вітаю всіх на наступній лекції з вивчення CSS.

Почнемо, як завжди, з теорії. При написанні коду на нашому рівні потрібно намагатися дотримуватися 3-х основних принципів.

1. **YAGNI** (You Aren't Gonna Need It - вам це не знадобиться) - якщо пишете код, будьте впевнені, що він вам знадобиться. Не пишіть код, якщо думаєте, що він знадобиться пізніше.

2. **DRY** (Don't Repeat Yourself — не повторюйте себе). Дублювання коду – марна трата часу та ресурсів. Вам доведеться підтримувати ту саму логіку і тестувати код відразу в двох місцях, причому якщо ви зміните код в одному місці, його потрібно буде змінити і в іншому. Тому, перед тим як писати код — проаналізуйте те, що вам потрібно зробити в цілому, щоб потім не робити дублікати того самого коду.

3. **KISS** (Keep It Simple, Stupid - робіть якомога простіше). Цей принцип вказує на те, що не потрібно вигадувати для завдання складнішого рішення, ніж йому потрібно. Як приклад, якщо ви використовуєте якісь нові інструменти тільки тому, що вони нові, при тому що вони не додають ніякого нового функціоналу.

Не намагайтеся все одразу зрозуміти. Ми будемо використовувати ці принципи поступово при виконанні домашніх завдань, а більш досконале зрозуміння та вміння застосовувати ці принципи прийде лише після отримання великого практичного досвіду. Йдемо далі.

Зразу хочу торкнутися такої важливої теми як ***неймінг*** (назва, ім'я). У сфері програмування неофіційно існує такий термін, як "говнокод"))). Що він означає? Наведу приклад оголошення змінних та присвоєння їм значення мовою Java Script.

```
let a = "Andrii"
let b = 52
```

За змістом ми розуміємо, що a - це ім'я, а b - це вік(це і є приклад говнокоду). Але коли ми будемо викликати ці змінні в листінгу коду програми - нам буде абсолютно не зрозуміло, що це за змінні і які повинні бути у них значення. Тому з назви змінних ми повинні розуміти їх призначення. Наприклад:

```
let name = "Andrii"
let age = 52
```

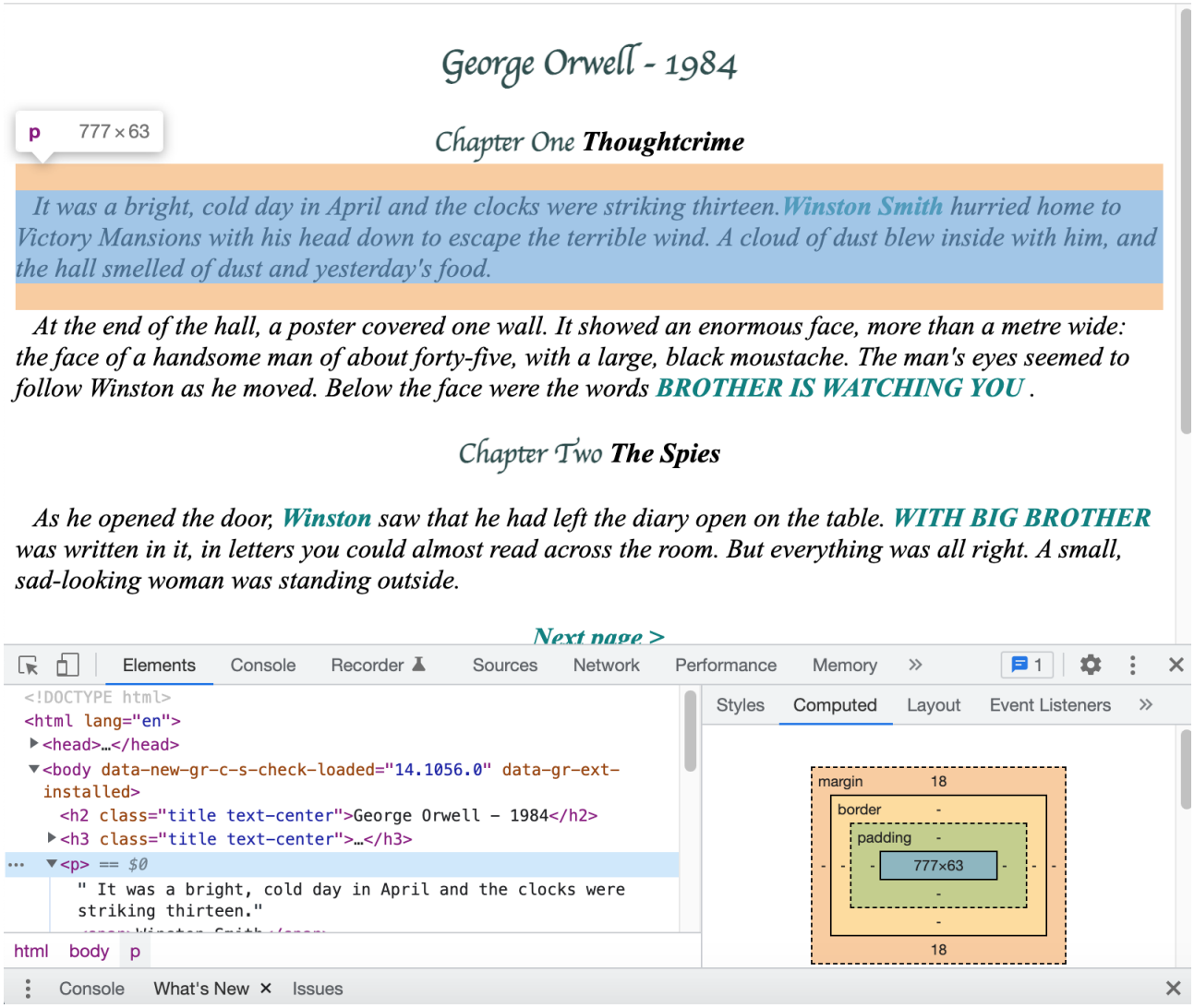
Для чого я це все написав? На минулому уроці ми почали вивчати такий важливий селектор як ***class***. Давайте закріпимо те, що ми вивчали на попередньому занятті.

- Ми можемо використовувати стилі для декількох селекторів, написавши їх через кому. Наприклад: **h1, h2, h3** {font-family: helvetica;}
- Синтаксис для класу - **.classname** { style properties }. Ми можемо використовувати декілька класів для одного елемента, використовуючи пропуск. Наприклад:
  Text або
  Text - порядок запису не має значення.

Назви наших класів ми також повинні писати осмислено. Декілька рекомендацій з неймінгу класів, поки що тільки для елементів з текстом для закріплення матеріалу. Наприклад ми маємо декілька елементів тексту червоного кольору, причому частина з них виділена жирним шрифтом. Давайте створимо 2 класи виходячи з цих умов. Перший можливо написати таким чином class="text-red". Другий ми можемо назвати просто **bold** так як ця властивість

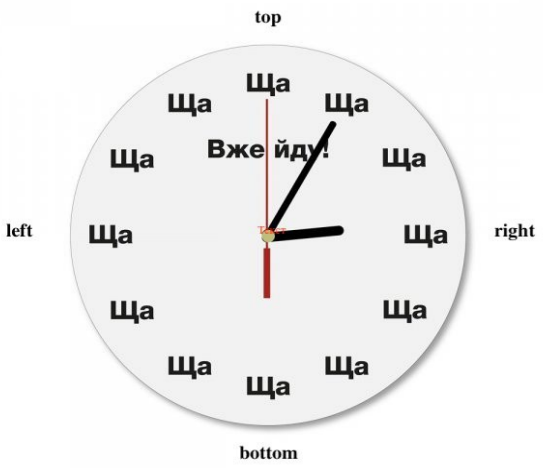
використовується тільки для тексту - class="bold". Таким чином для елементів тексту червоного кольору ми будемо використовувати class="text-red", а для жирного червоного тексту - class="text-red bold". Ще один приклад. Допустимо, що ми маємо сторінку з кількома заголовками та підзаголовками з різними стилями. Назві класу для заголовків можно дати ім'я class="title", а для підзаголовків — class="subtitle". Поки досить цієї інформації, продовжимо нашу тему.

Наступні важливі властивості CSS — ***padding***(внутрішній відступ), ***margin***(зовнішній відступ) та ***border***(універсальний атрибут, який дозволяє одночасно встановити товщину, стиль і колір межі навколо елемента). Давайте розглянемо ці властивості на прикладі 1-го домашнього завдання. На вкладці праворуч вибираємо меню **Computed** і вибираємо перший параграф. Ми бачимо властивості за замовчуванням для цього елемента сторінки.



Ми бачимо, що наш параграф має розмір 777*63px, внутрішній відступ(padding: 0px;), border: 0px; і зовнішній відступ(margin: 18px;)

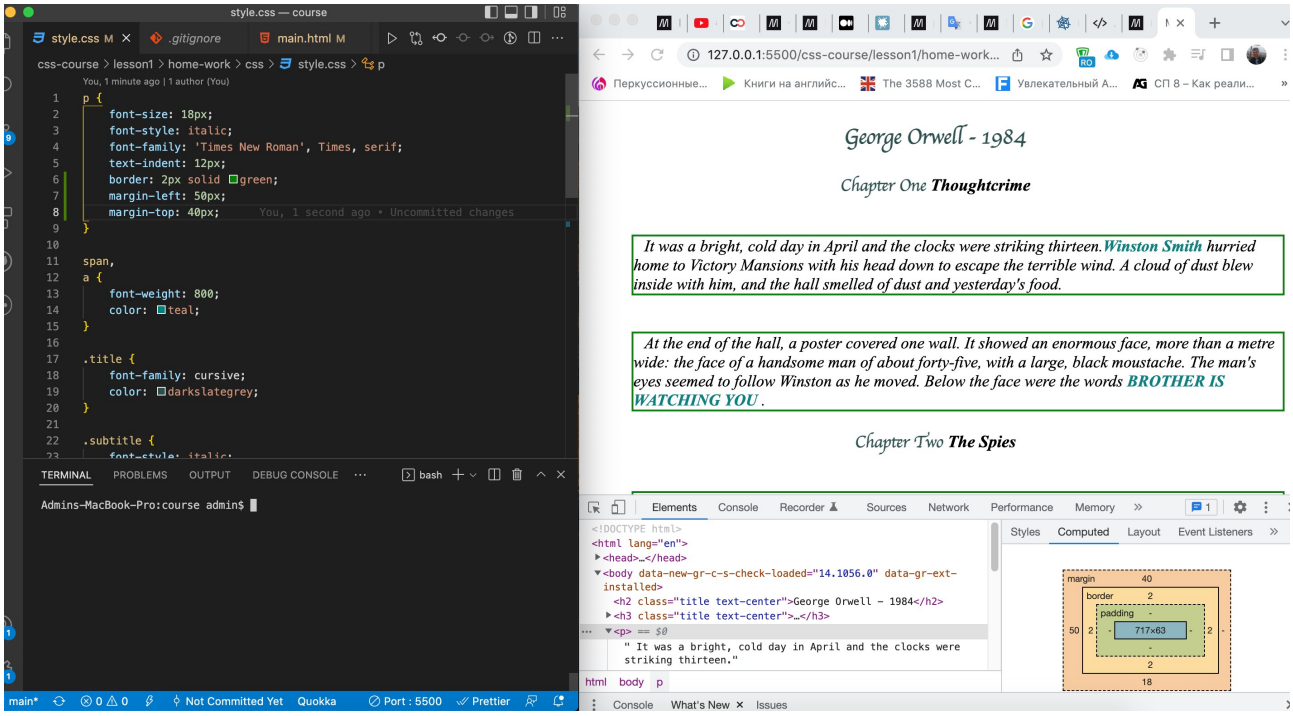
Всі ці властовості можуть мати окремі відступи: зверху(top), від правого краю(right), знизу(bottom), від лівого краю(left). Порядок запису відступів легко запам'ятати за допомогою малюнка:



Запис властивостей буде виглядати таким чином(зверху за годинниковою стрілкою):

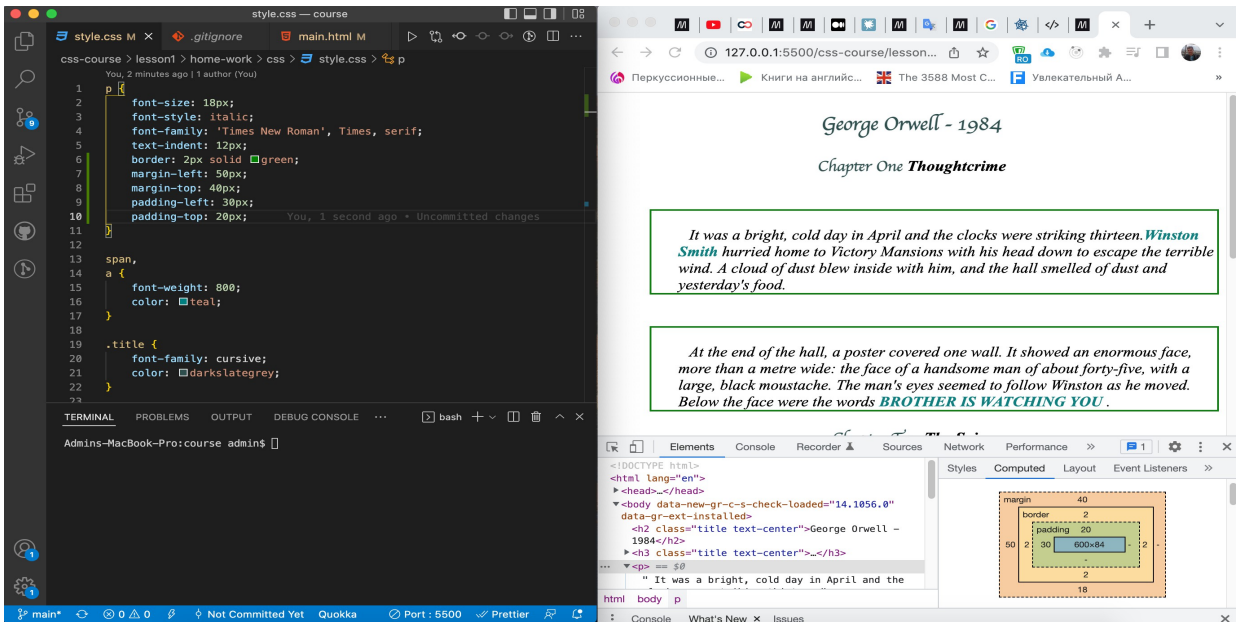
margin: top right bottom left;
padding: top right bottom left;
border: 1px solid black;

Давайте трохи пограємось і додамо border, margin-left та margin-top

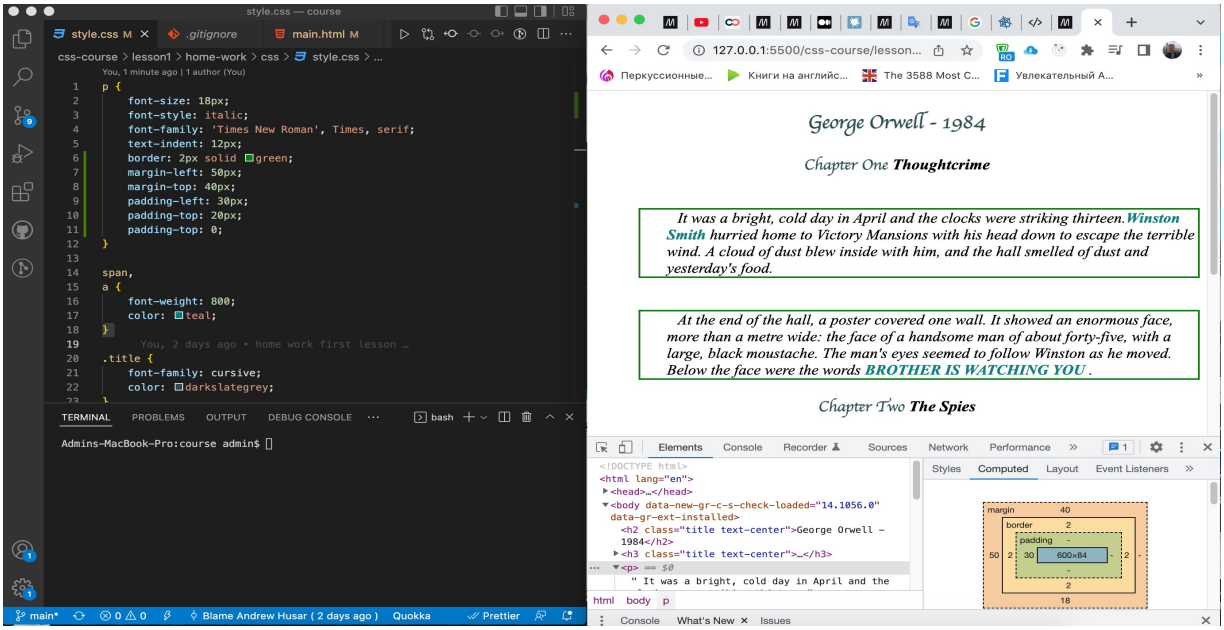


Важливо розуміти, що всі зовнішні відступи параграфів виконуються відносно батьківського елемента, в нашому випадку — це html сторінка.

Давайте додамо внутрішні відступи padding-left та padding-top — батьківським елементом для внутрішніх відступів є наш параграф **p**.



Тепер давайте зробимо наш внутрішній відступ padding-top: 0;



Як ви побачили - виконався останній рядок коду.

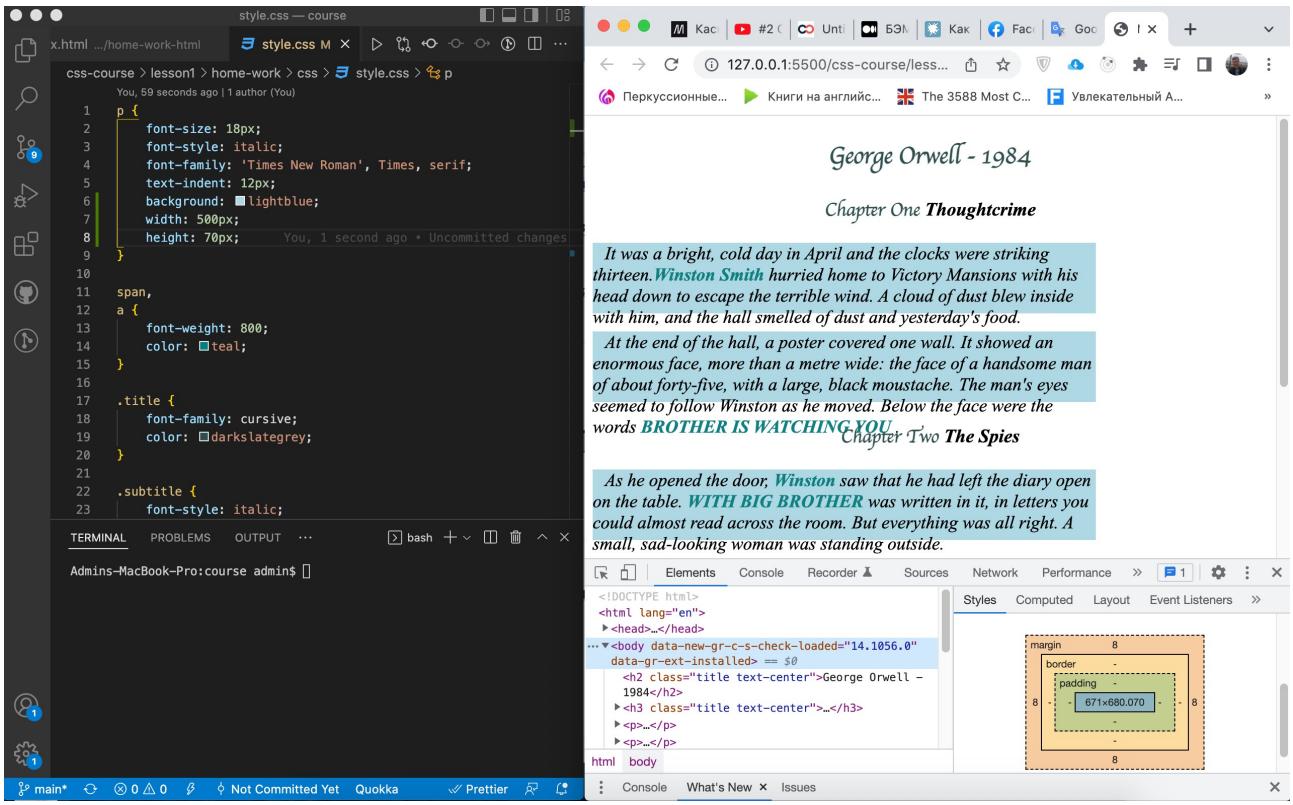
Основне правило для однакових властивостей - виконується те, що знаходиться в коді нижче.

Матеріали для самостійного вивчання:

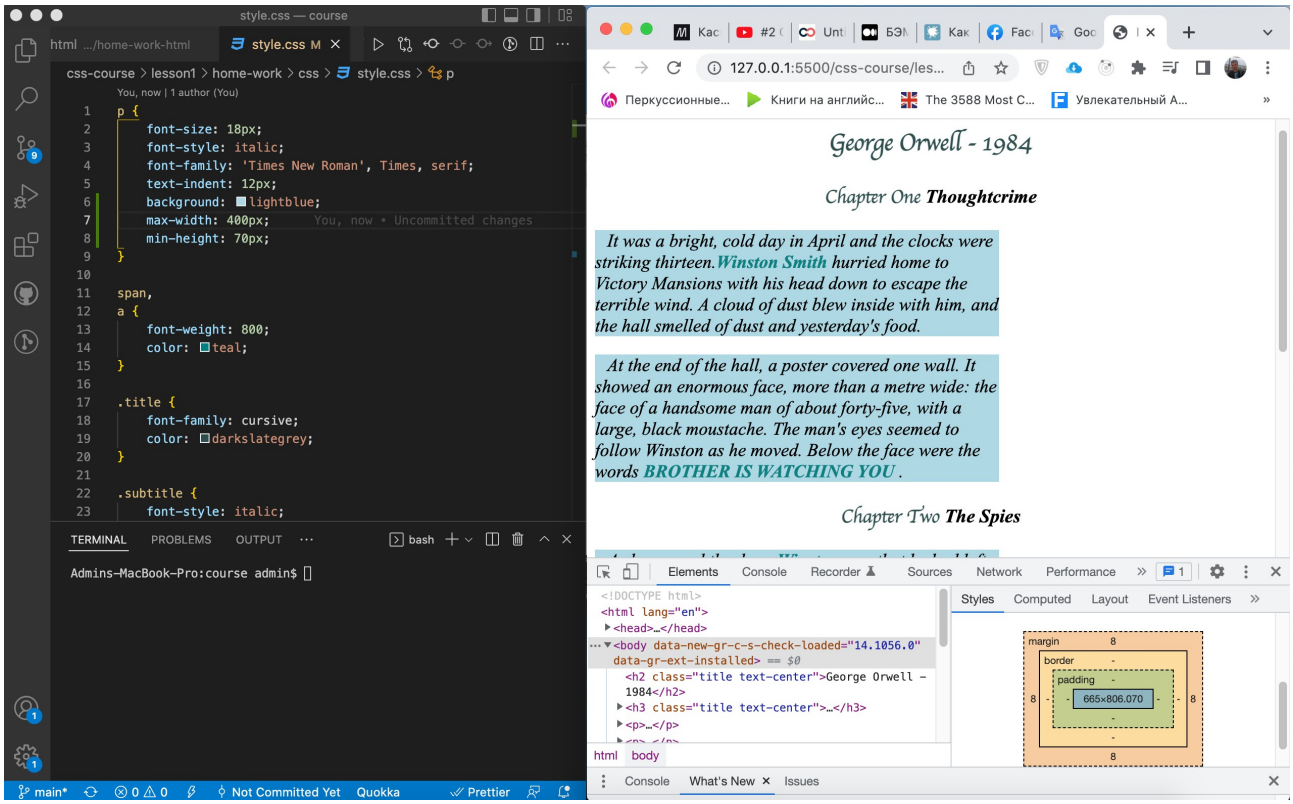
- https://developer.mozilla.org/en-US/docs/Web/CSS/margin
- https://developer.mozilla.org/en-US/docs/Web/CSS/padding
- https://developer.mozilla.org/en-US/docs/Web/CSS/border

Давайте розберемо ще кілька властивостей CSS, таких як ***width***(ширина), ***height***(висота) та ***background***(задній фон)

Видалимо код, що ми написали, щоб все було, як на початку уроку і напишемо новий для наших параграфів <p>.



Ми бачимо, що наші параграфи стали меншими за шириною та висотою, причому текст по висоті не вміщується в розміри нашого параграфу. Щоб виправляти такі помилки, ці властивості використовуються з префіксом **min-** або **max-** . Давайте зменшимо нашу ширину і зробимо її максимальною, а нашу задану висоту зробимо мінімальною.



Як бачимо, наша картинка стала менше шириною і текст заповнив весь параграф по висоті.

Матеріали для самостійного вивчання:

- https://developer.mozilla.org/en-US/docs/Web/CSS/width
- https://developer.mozilla.org/en-US/docs/Web/CSS/height
- https://developer.mozilla.org/en-US/docs/Web/CSS/background

До зустрічі на наступній лекції.