10.3. Сучасний CSS. Основні поняття, селектори, блокова модель

Особливості застосування та основні можливості сучасного CSS. Принципи каскадних таблиць стилів та типи селекторів. Поняття блокової та альтернативної моделей, опис їх властивостей.

1. Каскадні таблиці стилів: визначення та можливості

2. Основні категорії CSS

3. Способи підключення стилів

- Вбудовані стилі (inline styles)
- Внутрішні стилі (internal styles)
- Зовнішні стилі, що підключаються (external styles)

4. Види селекторів у CSS

- Глобальний селектор
- Селектор за елементом
- Селектор класу
- Селектор з ідентифікатора
- Селектор з атрибуту
- Селектор із псевдокласу
- Селектор із псевдоелементу
- Групові селектори
- Комбіновані селектори

5. Принципи роботи CSS

- Каскадність (Cascade)
- Специфіка (Specificity)
- Наслідування (Inheritance)

6. Блокові (Block) та малі (inline) елементи

7. Стандартна та альтернативна блокові моделі (Box models)

- Стандартна модель
- Альтернативна модель

Каскадні таблиці стилів: визначення та можливості

Стилізація елементів веб-сторінки та завдання зовнішнього вигляду документа досягається за допомогою мови CSS (Cascading Style Sheets, Каскадні таблиці стилів). Він дозволяє застосовувати всі сучасні можливості браузерів для відображення сайтів.

Каскадні таблиці стилів – інструмент опису зовнішнього вигляду сторінок веб-ресурсу, які використовують як конструктор мову розмітки HTML або XML. Поєднання CSS та HTML – невід'ємна частина веб-розробки. Таблиці стилів доповнюють мову розмітки, опосередковано розширюють його функціонал.

Недоліки сайтів на «чистому» HTML:

- Статичність (сторінки не змінюються, відсутня динаміка та анімації).
- Непередбачуваність поведінки в різних браузерах (хоч теги і розуміються практично всіма ними, проте візуально може модифікуватися їх відображення у користувача з врахуванням його налаштувань).
- Мінімалістичність, непривабливість – сам собою HTML практично не дозволяє добре налаштовувати зовнішній вигляд елементів (отже, приємних і ергономічних сайтів на ньому не побудуєш).

CSS першої версії з'явилися в далекому 1996 р. і містили не такий великий набір правил, як сьогодні. Щоправда, вже тоді було закладено основні категорії та поняття: селектори, блочно-рядкове форматування, псевдоелементи тощо.

Друга версія каскадних таблиць стилів вийшла у 1998 р і принесла із собою комбінатори, табличну модель, голосові стилі.

CSS3 функціонує з 2007 р. Він постійно розвивається та доповнюється, хоч і не змінює версію. До сьогодні CSS підтримує практично всі «фішки» сучасних браузерів і пристроїв і здатний задовольнити потреби різних груп користувачів, аж до тих, хто має обмежені можливості контакту з комп'ютером.

Для наочності розуміння стеку веб-розробки HTML-CSS-JavaScript наведемо аналогію. Уявіть будівлю, що будується:

- HTML - це каркас будівлі, дах, підлоги, стіни.
- CSS – елементи декоративної якості, прикраси (від шпалер, лінолеуму до меблів та картин у приміщенні). Вони створені спеціально для людини: зручності, естетичності. Адже комп'ютер все одно бачить лише байти.
- JavaScript – функціональні елементи (робота з освітленням, водопроводом, регулювання температури тощо).

Основні категорії CSS

Для роботи з CSS на сторінках сайту необхідно розібратися з термінологією. Структура будь-якого елемента каскадних таблиць представлена нижче.

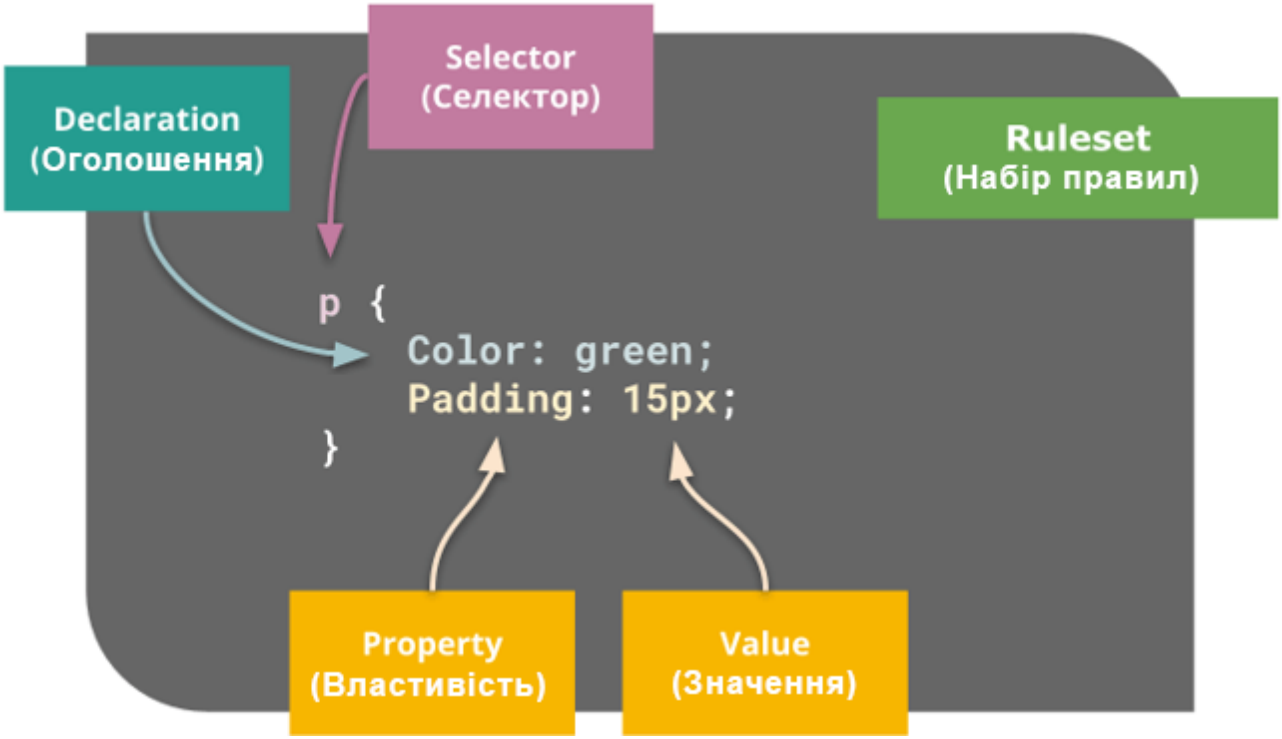

Рис Структура CSS-блоку

Завдання CSS-властивостей для HTML-елементів передбачає оголошення селектора, а всередині фігурних дужок через точку з комою перелічуються властивості та їх значення.

Охарактеризуємо кожну категорію:

- Ruleset (набір правил) – весь блок конкретного селектора з усіма параметрами, що змінюються. Наприклад, тег <p> та його властивості: колір та внутрішні відступи.
- Selector (селектор) - категорія, що модифікується (тег <p>). Як селектор можуть виступати не тільки теги, а й класи, ідентифікатори, псевдоелементи і псевдокласи, конкретні атрибути. Селектори бувають складними (з врахуванням успадкування).
- Declaration (декларація, оголошення) – конкретна пара *властивість: значення*. Кількість таких пар не обмежена. Пара обов'язково має закінчуватися крапкою з комою.
- Property (властивість) – атрибут елемента, якому задають певне значення. На рис. як такі представлено колір тексту (color), внутрішні відступи від меж блоку з усіх боків (padding).
- Value (значення) – конкретна величина атрибута. У прикладі колір тексту зроблено зеленим, а відступи – 15 пікселів. Варіанти значень властивостей визначаються документацією.

При порушенні синтаксису набору правил у будь-якому місці помилок на сторінці не видно, але самі правила не будуть застосовані до обраного селектора. Якщо щось працює не так як задумано, потрібно перевірити синтаксис, валідність властивостей і селекторів, допустимість значень.

Способи підключення стилів

Щоб каскадні таблиці стилів відображалися на сторінці, необхідно не лише правильно їх задати, але й підключити. Для цього є 3 варіанти.

Вбудовані стилі (inline styles)

Кожному тегу в HTML-документі можна задати атрибут style, всередині якого описуються властивості та значення.

Приклад - HTML

```
<article style="color:darkgoldenrod; font-size:xx-large;">Параграф тексту
</article>
```

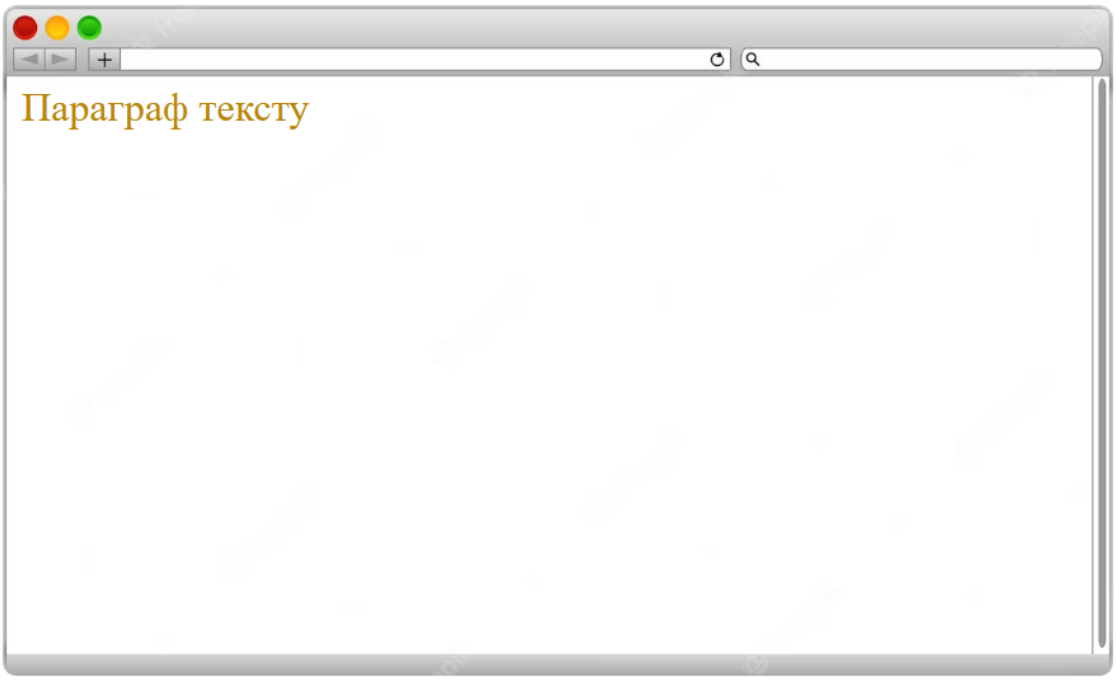

Рис. Сторінка браузера

Насправді прийнято ставити значення атрибутів всередині подвійних лапок (хоча можна використовувати і одинарні). У наведеному прикладі для тегу <article> визначено темно-золотистий колір шрифту та збільшений розмір.

Таке застосування стилізації елементів не рекомендується, оскільки втрачається сенс CSS, але в деяких випадках допустимо (для отримання максимального пріоритету стилю).

Внутрішні стилі (Internal styles)

Використовуються всередині HTML-сторінки без винесення окремий файл. Виправдано у разі невеликої кількості елементів, що модифікуються, на компактних сторінках. Визначаються у тезі <style>.

Приклад - HTML

```
<!DOCTYPE html>
<html lang="uk">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Практика CSS</title>
   <link rel="stylesheet" href="style.css">
   <style>
```

```
      article {
            color : darkgoldenrod;
            font-size:xx-large;
      }
   </style>
</head>
<body>
   <article>Параграф тексту</article>
</body>
</html>
```

Цей спосіб завдання стилів декларується, як правило, у тезі <head>.

Зовнішні стилі, що підключаються (external styles)

Найпоширеніший спосіб завдання стилів на сторінках. Для цього створюється один або кілька файлів з розширенням .css , шлях до яких вказується в тегу <link>, розташованому в заголовку HTML-документа.

Приклад - HTML

```
<!DOCTYPE html>
<html lang="uk">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Практика CSS</title>
   <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
   <article>Параграф тексту</article>
</body>
</html>
```

Приклад - CSS

```
/* Файл style.css */
article {
    color:darkgoldenrod;
    font-size:xx-large;
}
```

До документу HTML підключили зовнішній файл із таблицями стилів style.css. В тезі <link> атрибут type в даному випадку можна не вказувати, оскільки CSS на сьогодні єдина мова таблиць стилів, що використовується в мережі.

То який варіант краще?

Три наведені способи дають однаковий результат. У сучасній веб-розробці, коли стилів для елементів може бути величезна кількість, найбільш оптимальним є третій спосіб. Для цього зазвичай в папці проекту створюється папка css, всередині якої розміщуються таблиці стилів.

У подальших прикладах використовуватиметься HTML-сторінка index.html, поруч із якою розташується папка css , де створимо текстовий документ style.css.

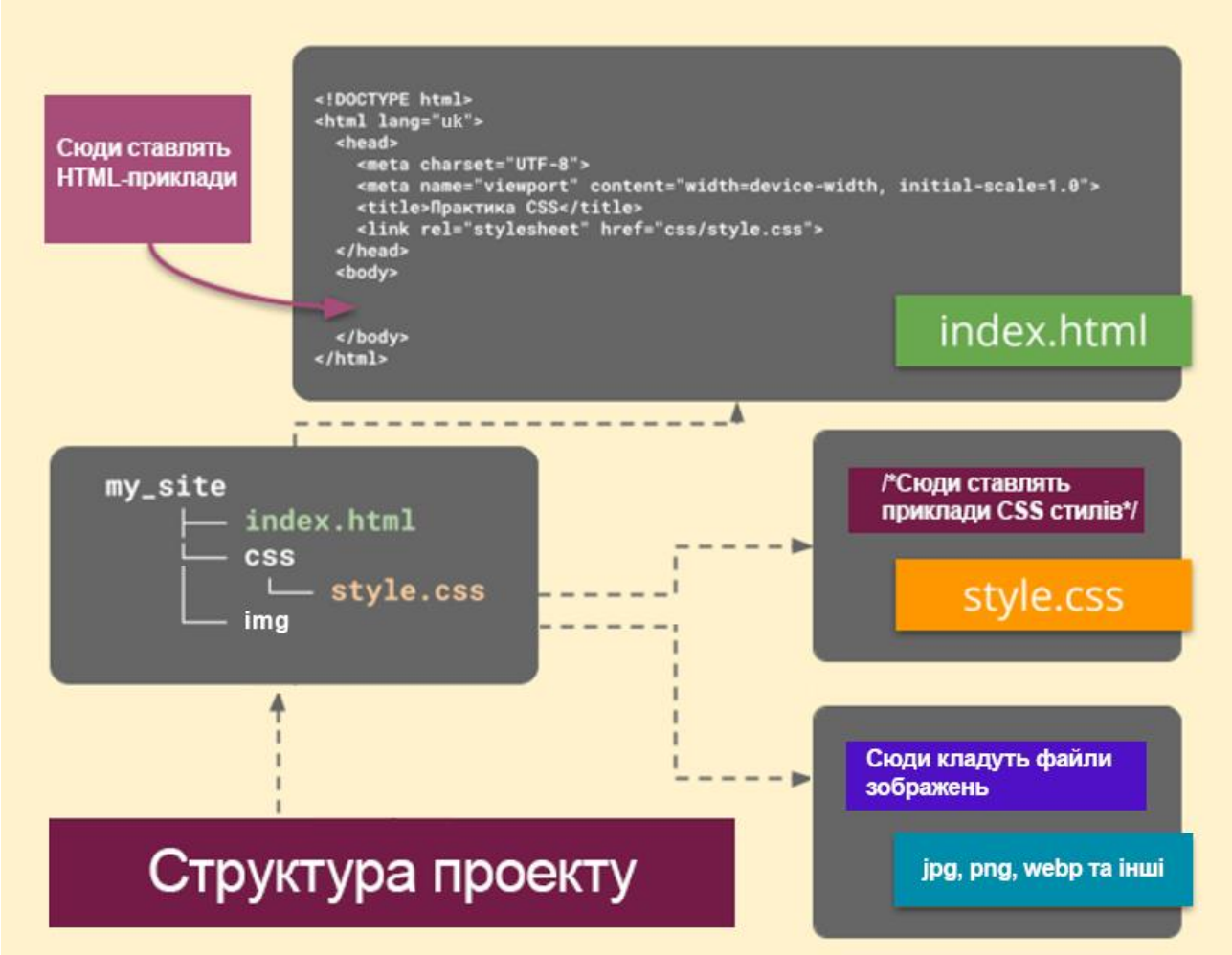

Рис.1 Структура простого проекту

Базова структура кореневого HTML-файлу:

Приклад - HTML

```
<!DOCTYPE html>
<html lang="uk">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Практика CSS</title>
   <link rel="stylesheet" type="text/css" href="css/style.css">
</head>
<body>
</body>
```

```
</html>
```

Надалі вона наводитися не буде. Всередині тега <body>за необхідності розміститься інші елементи (тобто, у прикладах покарfуj лише вміст <body>).

Види селекторів у CSS

Завдання властивостей елементам сторінки через каскадні таблиці стилів передбачає початкову вибірку селекторів. Варіантів здійснення такої операції багато:

1. Глобальний селектор
2. Селектор за елементом
3. Селектор за класом
4. Селектор з ідентифікатора
5. Селектор за атрибутом
6. Селектор із псевдокласу
7. Селектор із псевдоелементу
8. Групові селектори
9. Комбіновані селектори
    a. Селектор нащадків
    b. Селектор дочірніх елементів
    c. Селектор "першого сусіда"
    d. Селектор "усіх сусідів"

Глобальний селектор

Оголошується зірочкою і має вплив на всі елементи сторінки, де є ця властивість.

Приклад - HTML

```
<h1>Головна сторінка сайту</h1>
<article>Як працювати з CSS</article>
```

Приклад - CSS

```
* {
    background-color:lightsalmon;
    font-size:30px;
}
```

Кожному тегу на сторінці сайту буде надано колір фону світло-рожевий, а також розмір шрифту в 30 пікселів.

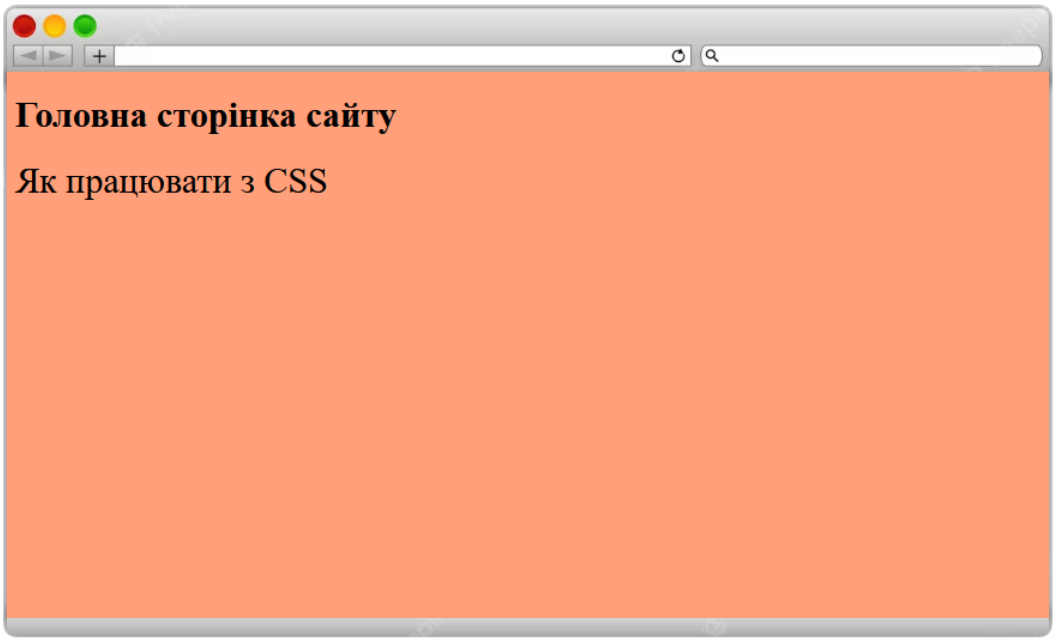

Рис. Глобальний селектор

Цей селектор використовується не часто, але іноді потрібний для скидання всіх налаштувань (оскільки браузери часто самі задають відступи у блокових тегів, наприклад, а ми хочемо їх відключити). Дозволяє досягти максимальної ідентичності зовнішнього вигляду ресурсу на різних браузерах та пристроях.

Селектор елемента

Як селектор виступає конкретний тег HTML. Нові налаштування поширяться на всі ці об'єкти, що зустрічаються в коді, незалежно від вкладеності.

Приклад - HTML

```
<h1>Головна сторінка сайту</h1>
<q>Практика – основа майстерності</q>
<p>
   <q>Роби - і навчишся</q>
</p>
```

Приклад - CSS

```
q {
    color:maroon;
    font-size:15px;
}
```

Всі цитати в документі набули бордовий відтінок і відображаються шрифтом розміром 15 пікселів.

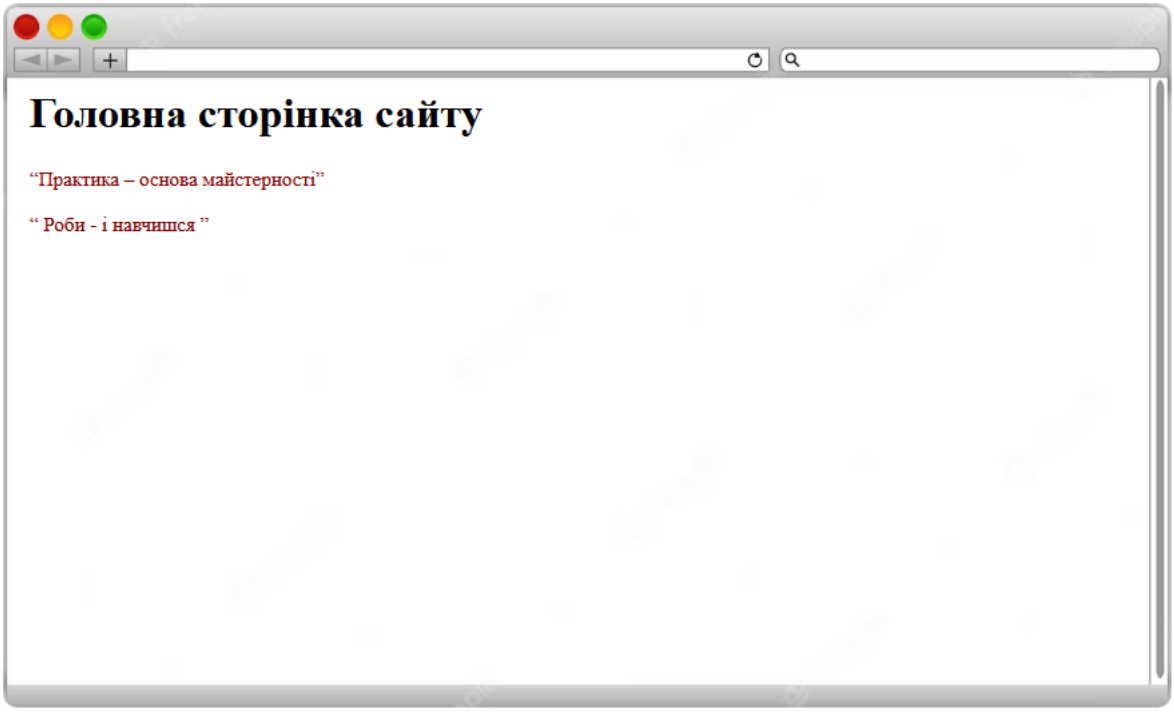

Рис. Селектор елемента

Селектор класу

Кожному HTML-елементу можна привласнити клас, задавши йому певне ім'я. Це дозволяє групувати теги та виділяти їх особливим чином у документі. У CSS до класу звертаються із селектором, що починається з точки .

Приклад - HTML

```
<h1 class="blue-wave">Головна сторінка сайту</h1>
<q class="blue-wave">Практика – основа майстерності</q>
<p>
   <q>Роби - і навчишся</q>
</p>
```

Приклад - CSS

```
.blue-wave {
    text-decoration-style:wavy;
    text-decoration-color:blue;
    text-decoration-line:overline;
}
```

Створено клас blue-wave, який заданий для заголовка <h1> та першої цитати. Всім елементам цього класу надано верхнє підкреслення синього кольору у вигляді хвилястої лінії.

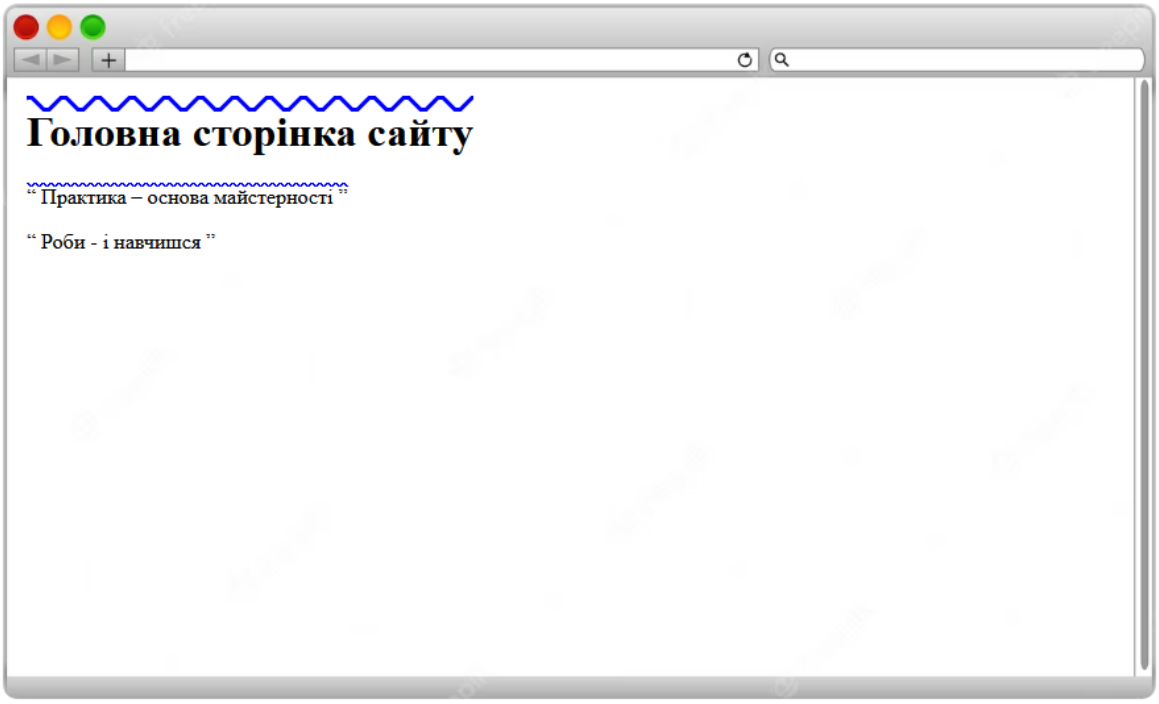

Рис. Селектор класа

Селектор ідентифікатора

ID надає змогу зробити унікальним конкретний елемент документа. Ідентифікатор певного імені може бути присутнім на сторінці в єдиному екземплярі (на відміну від імені класу). Найчастіше застосовується при обробці тегів та їх вмісту через JavaScript .

Приклад - HTML

```
<h1 id="red-dots">Головна сторінка сайту</h1>
   <q class="blue-wave">Практика – основа майстерності</q>
   <p>
       <q>Роби - і навчишся</q>
   </p>
```

Приклад - CSS

```
.blue-wave {
    text-decoration-style:wavy;
    text-decoration-color:blue;
    text-decoration-line:overline;
}
#red-dots {
    text-decoration-style:dotted;
    text-decoration-color:red;
    text-decoration-line:underline;
}
```

Селектор ідентифікатора починається зі знаку решітки. Тепер заголовок першого рівня підкреслено знизу червоною лінією у вигляді крапок.

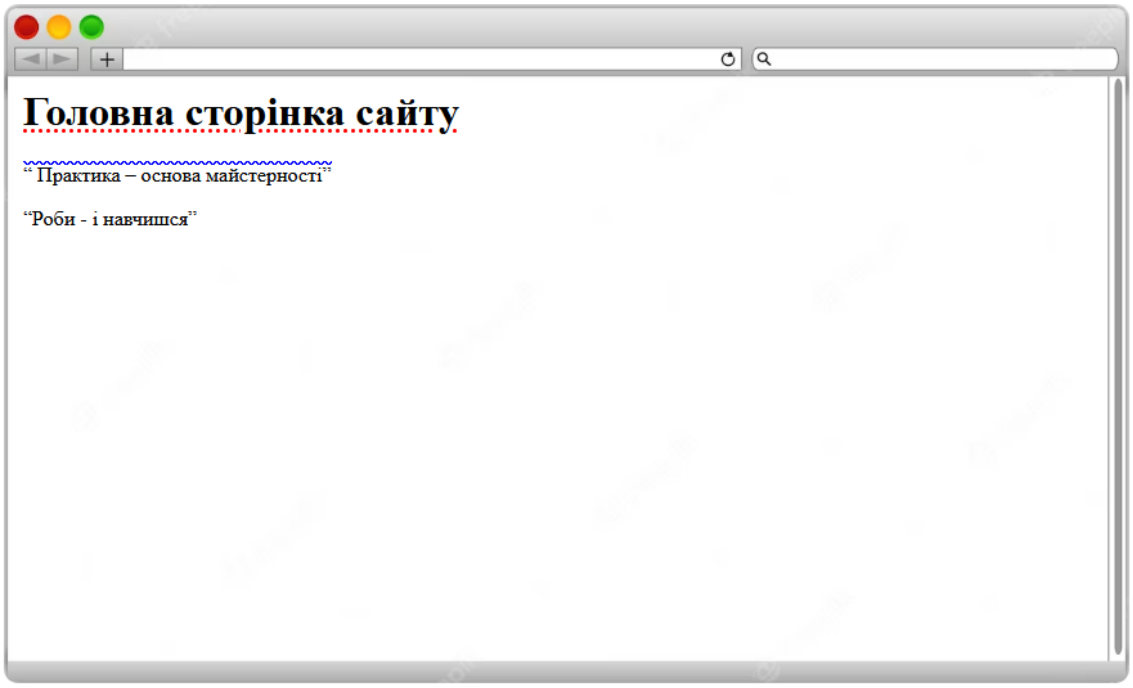

Рис. Селектор ідентифікатора

Селектор з атрибутом

Вибрати елемент документа можна і за наявності у нього певної властивості та значення . Для цього в квадратних дужках вказується атрибут з або без значення.

Приклад - HTML

```
<a href="https://bing.com/">Bing</a>
<br>
<a href="https://www.google.com/">Google</a>
```

Приклад - CSS

```
a[href="https://bing.com/"] {
font-size: 40px;
text-decoration-line: line-through;
}
```

Посилання, що веде на Bing, суттєво збільшилося у розмірі і стало перекресленим, тоді як з Google нічого не сталося.

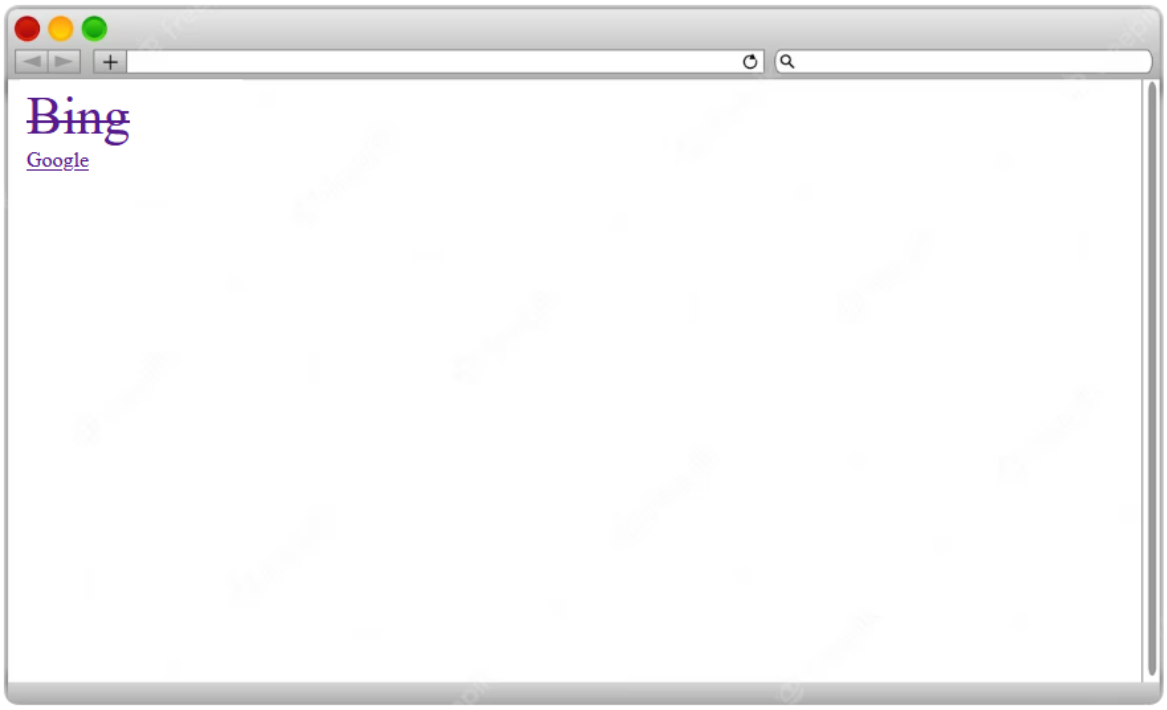

Рис. Селектор з атрибутом

Селектор з псевдокласом

Псевдоклас визначає особливий стан HTML-елемента. Це дозволяє стилізувати його не лише на підставі знаходження в DOM-дереві, але й з врахуванням низки зовнішніх факторів (історія відвідування, положення курсору миші тощо). На сьогодні їх налічується близько 40 псевдокласів, але вони застосовні не до всіх елементів. Для правильної роботи слід ознайомитись із документацією, оскільки можливі непередбачені ефекти.

Задаються через двокрапку:

Приклад - HTML

```
<a href="https://bing.com/">Bing</a>
<br>
<a href="https://www.google.com/">Google</a>
```

Приклад - CSS

```
a:link {
    font-size:15px;
    color:tomato;
}
a:visited {
    font-size:20px;
    color:saddlebrown;
}
a:hover {
    font-size:25px;
    color:seagreen;
}
```

```
a:active {
    font-size:30px;
    color:violet;
}
```

Спочатку всі посилання мають невеликий розмір шрифту та червоний відтінок. Відвідані посилання стають коричневими. При наведенні миші отримуємо зелений колір, а активне посилання (при натисканні та утриманні лівої кнопки миші) набуває фіолетового відтінку.

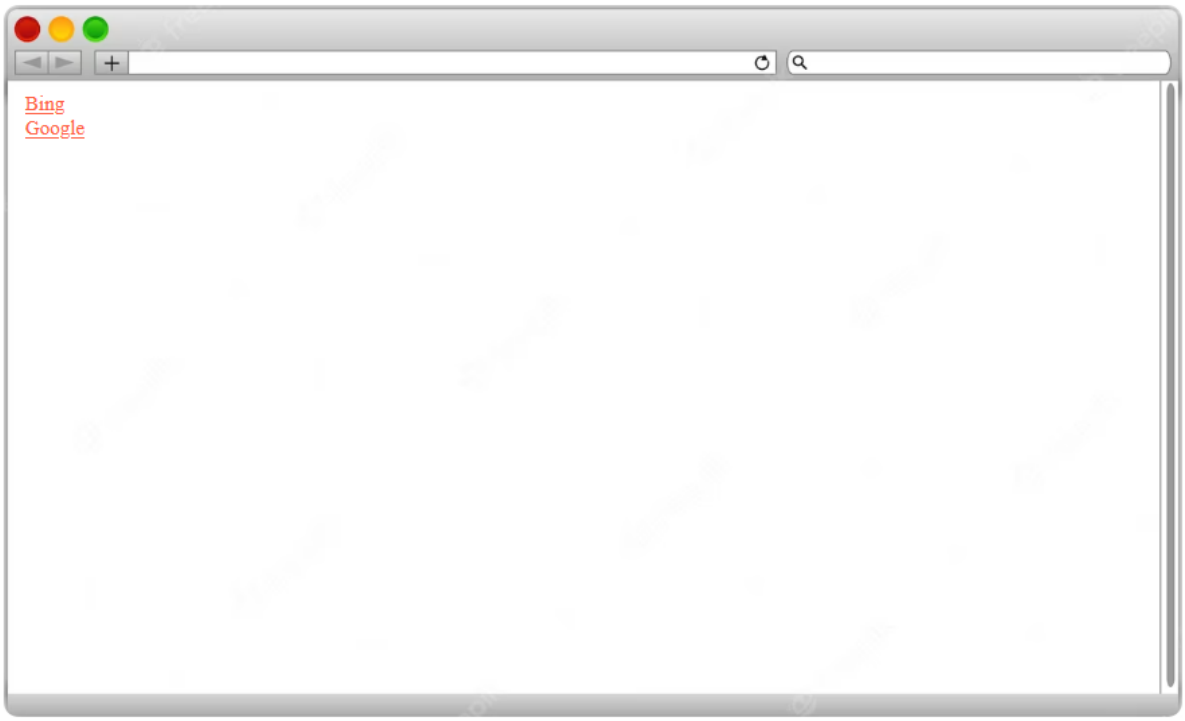

Рис. Сторінка браузера до переходу за посиланнями

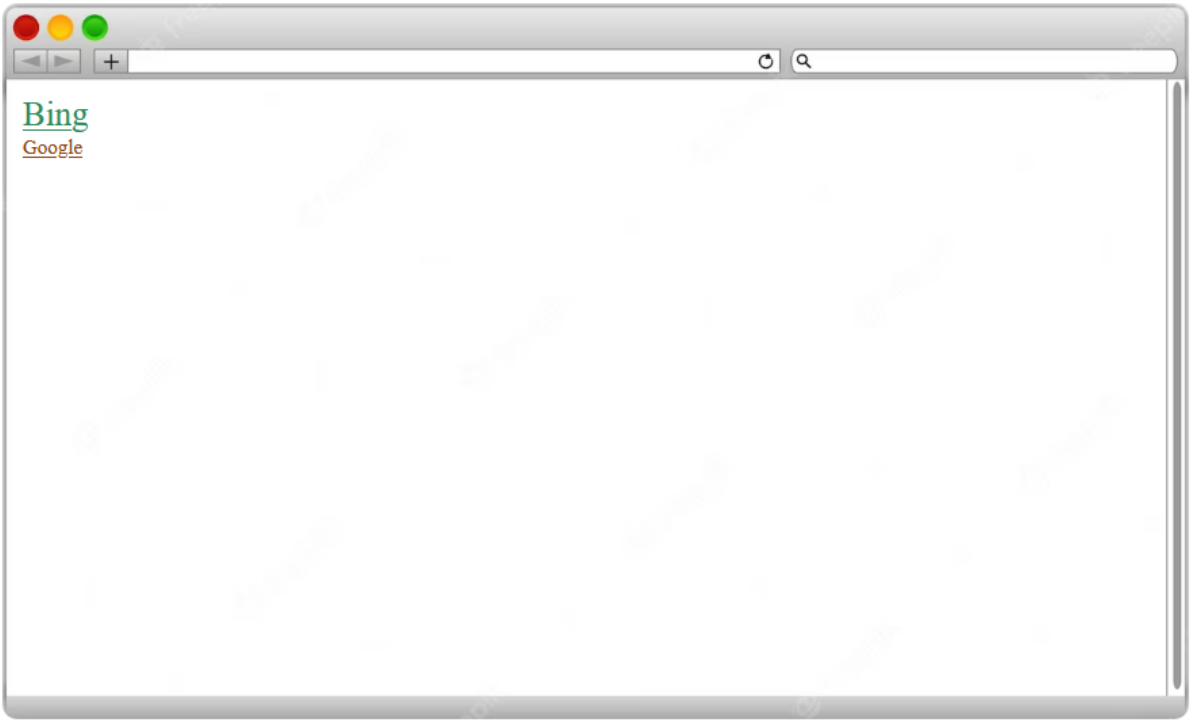

Рис. Сторінка браузера після наведення курсору на посилання (`a:hover`)

Відповідно до документації, наведений порядок оголошення псевдоелементів (a:link , a:visited, a:hover , a:active) є обов'язковим, інакше вони не коректно працюватимуть.

Селектор з псевдоелементом

Псевдоелементи стилізують деяку частину елемента. Задаються через дві двокрапки. Їх поки що не так багато, проте вони досить часто зустрічаються в коді верстальників.

Приклад - HTML

```
<h1>Вивчаємо псевдоелементи</h1>
<section>
    За допомогою ::first-line перетворимо перший рядок даного тегу до верхнього регістру.
    <br>
    Інші рядки при цьому залишаються незмінними, такими, якими ми їх поставили спочатку.
</section>
```

Приклад - CSS

```
section::first-line {
    text-transform:uppercase;
}
```

Як бачимо, перший рядок повністю переведено у верхній регістр.

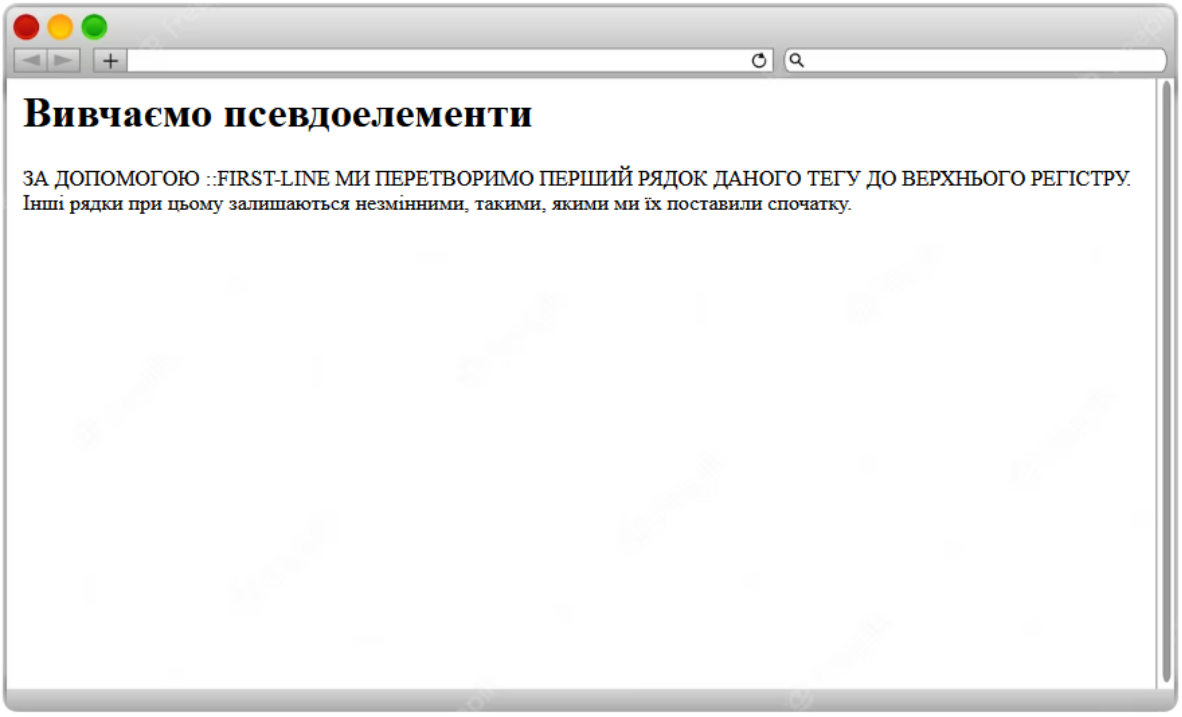

Рис. Селектор з псевдоелементом

Групові селектори

При перерахуванні будь-яких селекторів через кому їм можна задати однакові властивості.

Приклад - HTML

```
<h1>Фанати зеленого кольору</h1>
```

```
<article>
   <header>
          Говорять, зелений колір заспокоює очі.
   </header>
   <section>
          Дизайнери вважають, що застосування зеленого кольору сприяє відпочинку очей. Але це не доведено вченими, тож повіримо на слово.
   </section>
   <footer>
          Дослідження планується до проведення у Швеції, тоді й перевіримо.
   </footer>
</article>
```

Приклад - CSS

```
h1, header, section {
    color:seagreen;
}
```

Для трьох тегів (h1, header, section) задано колір шрифту світло-зеленого відтінку, що не призвело до дублювання коду.

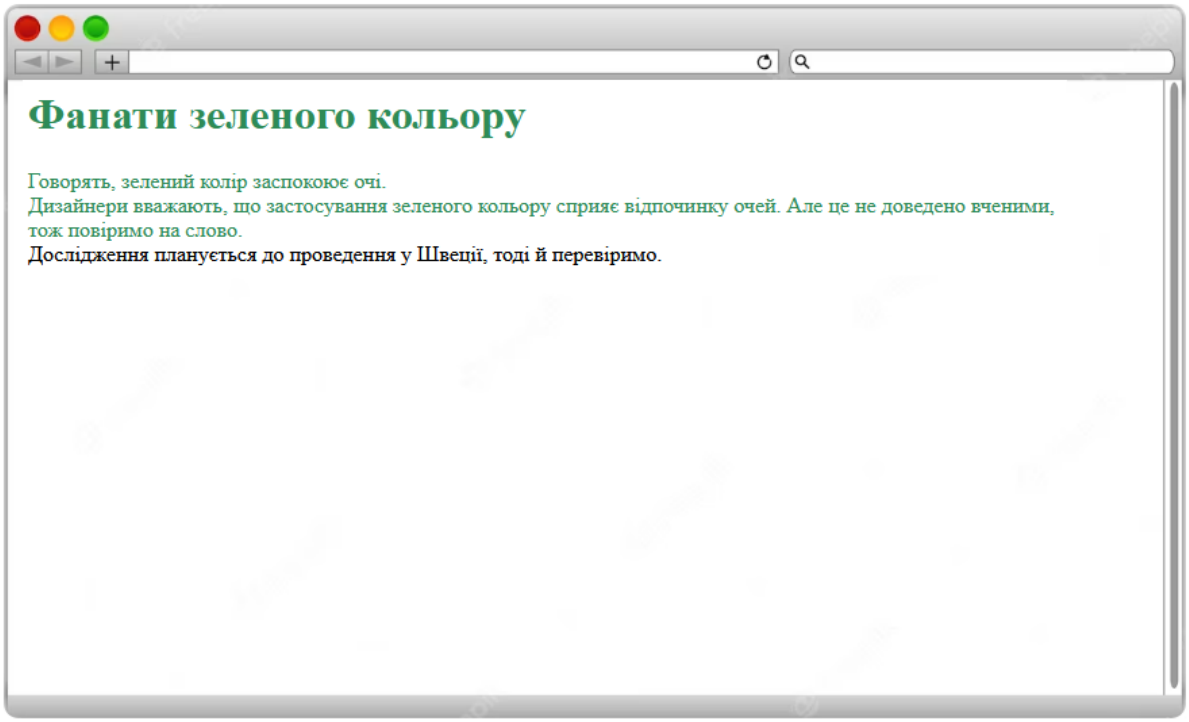

Рис. Групові селектори

Комбіновані селектори

Використання комбінаторів дозволяє вибирати елементи виходячи з відносини з-поміж них. Виділяють 4 типи комбінаторів:

Селектор нащадків

Задається пробілом. Будуть задіяні всі елементи, що розташовані всередині батьківського, незалежно від рівня вкладеності.

Приклад - HTML

```
<h1>Комбінатори</h1>
<p>Що це і навіщо</p>
<article>
   <header>
       <p>Перший параграф</p>
       <p>Другий параграф</p>
   </header>
   <section>
       <p>Третій параграф</p>
       <p>Четвертий параграф</p>
   </section>
   <footer>
       <p>П'ятий параграф</p>
       <p>Шостий параграф</p>
   </footer>
</article>
```

Приклад - CSS

```
article p {
    font-size:25px;
    background-color:bisque;
}
```

Всередині тегу <article> є теги. Незважаючи на те, що вони не дочірні елементи, селектор вплинув на них відповідно до правила.

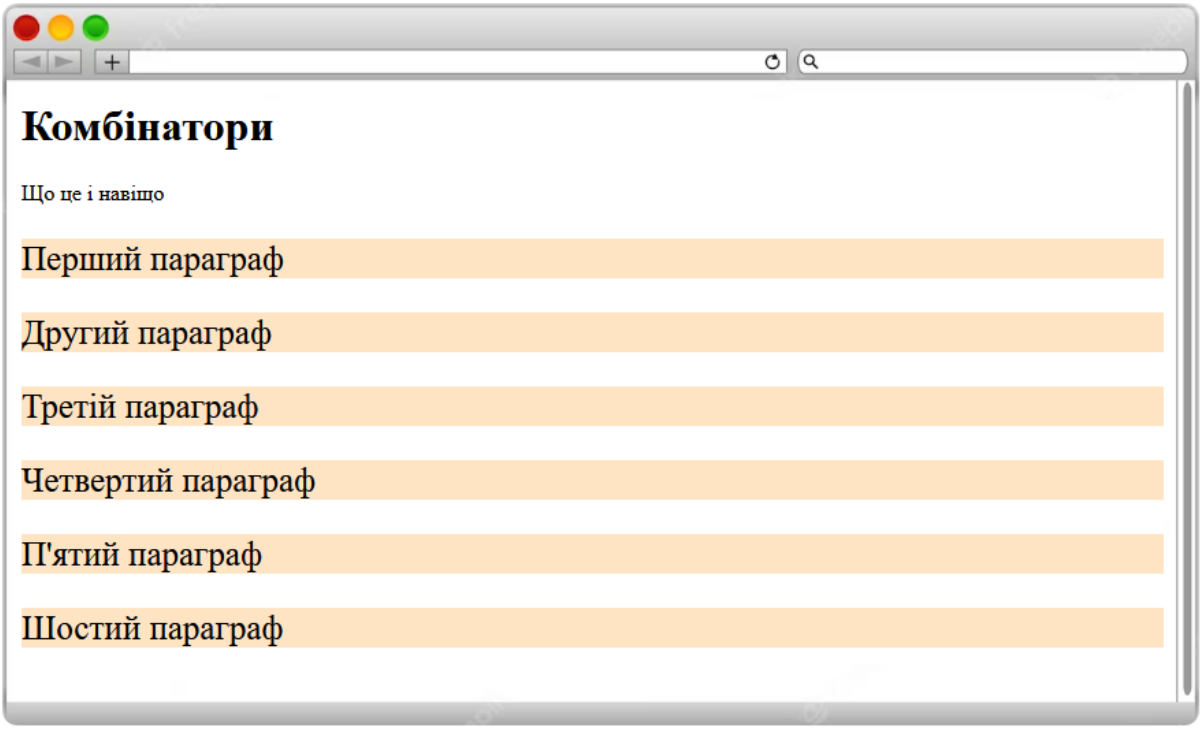

Рис. Селектор нащадків

Селектор дочірніх елементів

Вибирає лише перших нащадків, ігноруючи інших на більш глибоких рівнях вкладеності. Визначається знаком >.

Приклад - HTML

```
<body>
   <h1>Комбінатори</h1>
   <p>Що це і навіщо</p>
   <article>
       <header>
           <p>Перший параграф</p>
           <p>Другий параграф</p>
       </header>
       <section>
           <p>Третій параграф</p>
           <p>Четвертий параграф</p>
       </section>
       <footer>
           <p>П'ятий параграф</p>
           <p>Шостий параграф</p>
       </footer>
   </article>
   <p>Кінець</p>
</body>
```

Приклад - CSS

```
body > p {
    font-size:25px;
    background-color:bisque;
}
```

Вплив селектора спостерігається лише на ті параграфи, які безпосередньо успадковуються від <body> (з текстом Що це навіщо і Кінець).

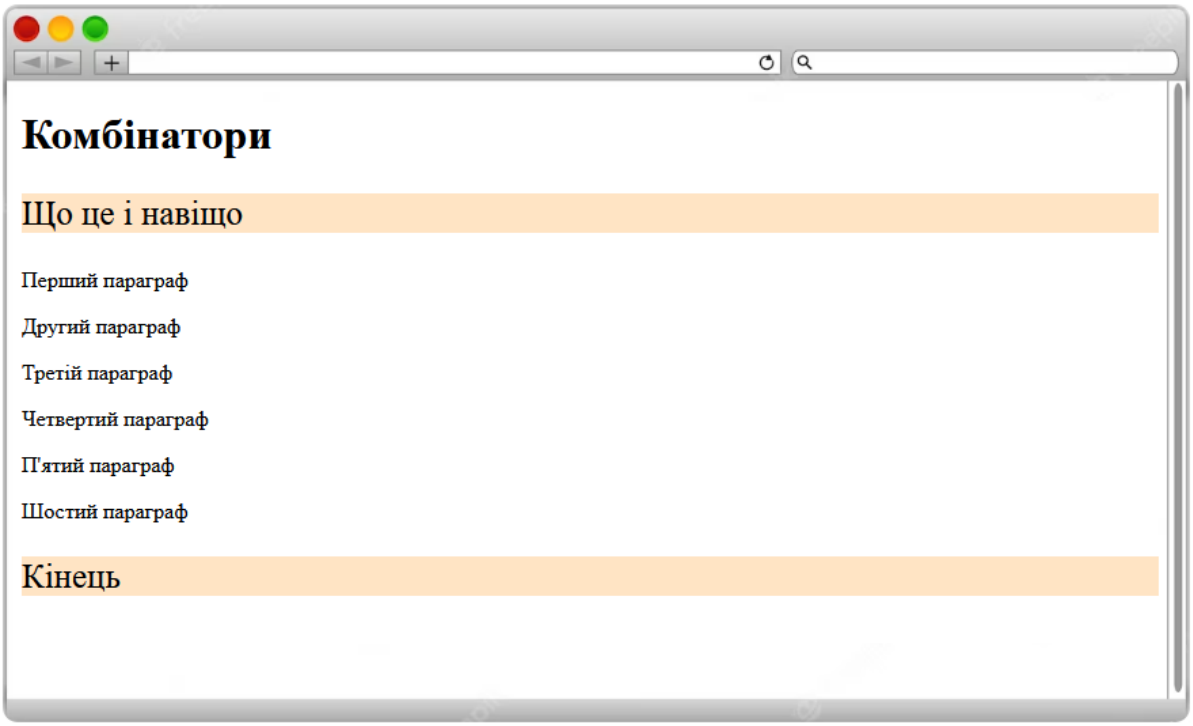

Рис. Селектор дочірніх елементів

Селектор першого сусіда

Коли 2 елементи на сторінці йдуть один за одним на одному рівні DOM-дерева, можна застосувати комбінатор першого сусіда. Для цього використовується знак +.

Приклад - HTML

```
<h1>Комбінатори</h1>
<p>Що це і навіщо</p>
<article>
   <header>
       <p>Перший параграф</p>
       <p>Другий параграф</p>
   </header>
   <section>
       <p>Третій параграф</p>
       <p>Четвертий параграф</p>
   </section>
   <footer>
```

```
        <p>П'ятий параграф</p>
        <p>Шостий параграф</p>
    </footer>
</article>
<p>Кінець</p>
```

Приклад - CSS

```
h1 + p {
    font-size:25px;
    background-color:burlywood;
}
```

Колір і шрифт змінився лише в першого параграфа, що йде після тегу <h1> і з ним одному рівні ієрархії.

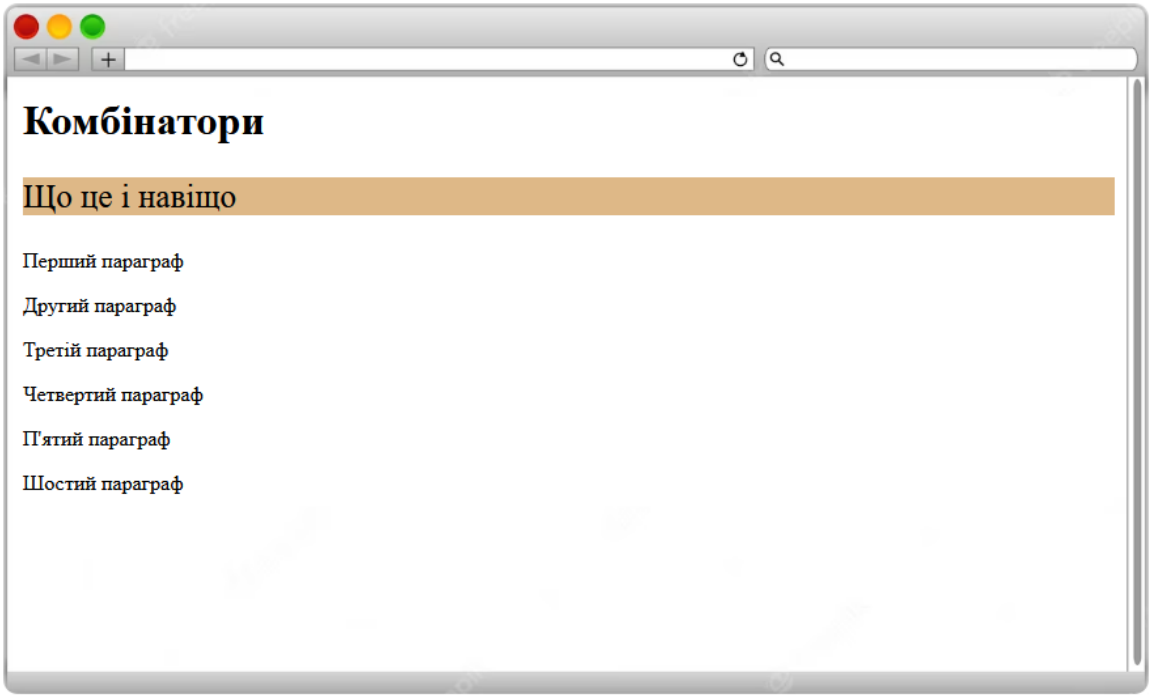

Рис. Селектор першого сусіда

Селектор всіх сусідів

Вибираються всі сусіди, що йдуть далі (тобто, що знаходяться на одному рівні ієрархії елементи). Задаються символом ~.

Приклад - HTML

```
<h1>Комбінатори</h1>
<p>Що це і навіщо</p>
<article>
    <q>Якась цитата</q>
    <p>Перший параграф</p>
    <p>Другий параграф</p>
</article>
<p>Кінець</p>
```

Приклад - CSS

```
article ~ p,
q ~ p {
    font-size:25px;
    background-color:burlywood;
}
```

Як видно, змінено лише параграфи з текстом Перший параграф, Другий параграф та Кінець як сусідні і наступні після початкових (q і article відповідно).

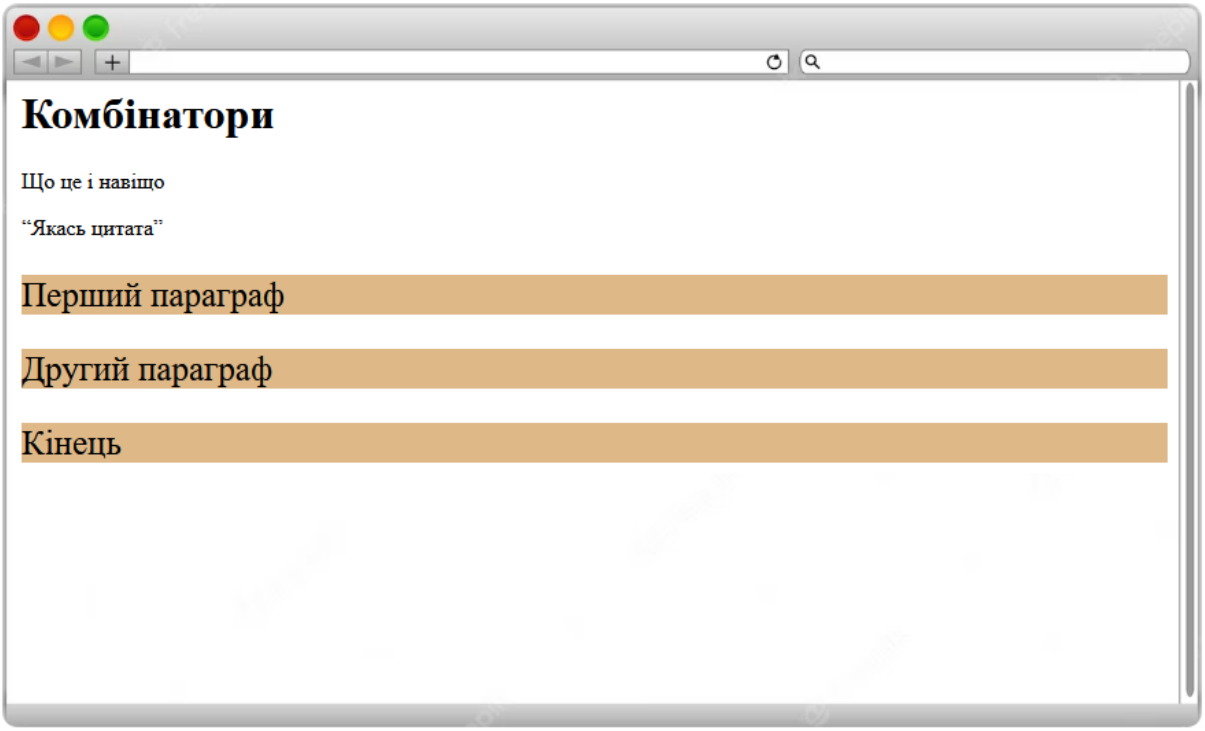

Рис. Селектор всіх сусідів

Застосовуючи різні техніки комбінування селекторів, можна задавати унікальні властивості для різних частин веб-сторінки при мінімізації обсягів коду та виключення дублювання.

Принципи роботи CSS

Працюючи з каскадними таблицями стилів виникають ситуації, коли не зрозуміло, яка буде у результаті властивість того чи іншого елемента. Наприклад, ми задали розмір тексту для тегу <body>, а потім і для тегу <p>. Виникає питання: якого розміру буде вміст параграфа? Таких ситуацій для формування складних таблиць стилів можливе багато.

Для розуміння пріоритетів успадкування стилів визначено принципи роботи CSS .

Каскадність (Cascade)

Каскадність це можливість застосування стилів з різних джерел до того самого об'єкта. Браузеру необхідно розібратися з тим, як відображати елемент.

Для цього вибудовується черга пріоритетів у наступному порядку (від найнижчого до максимально значущого):

1. Таблиці стилів браузера (у багатьох браузерах для різних елементів визначено стандартні стилі – шрифти, відступи, межі, розміри).
2. Таблиці стилів користувача (відвідувач сайту може заздалегідь налаштувати відображення тегів у браузері. Якщо вони не перевизначені розробниками, то будуть використані).
3. Таблиці стилів розробника (розробник задає параметри елементам, які за пріоритетом вищі, ніж два попередні типи. Це розраховано на максимальну схожість зовнішнього вигляду ресурсу у користувачів).
4. Стилі браузера !important (значення !important у будь-якої якості має підвищений пріоритет).
5. Стилі користувача !important (користувач також може примусово викликати певний стиль, встановивши значення !important).
6. Стилі розробника ! Important (максимально високий пріоритет).

Рис. Черга пріоритетів стилів

Пріоритетність застосування стилів

Значення !important не рекомендується встановлювати, але в деяких випадках можуть використовуватися для впевненості спрацьовування стилізації.

Як бачимо, стилі розробника вищі за пріоритетом, ніж усі інші. Якщо вони встановлені, то відображатимуться. В іншому випадку «рішення» приймає браузер або налаштування користувача.

Приклад - HTML

```
<h1>Основна сторінка</h1>
<article>
   <p>Важливий вміст статті</p>
```

```
</article>
<footer>Сайт розроблений site.com</footer>
```

Приклад - CSS

```
p {
    font-size:24px;
    color:chocolate;
    background-color:lightblue;
}
```

Отже, для параграфа змінили фон, колір та розмір шрифту. Тим не менш, він має й інші властивості, успадковані від браузера. Їх можна переглянути в консолі розробника (Клавіша F12 - >Стилі):

Приклад - Консоль розробника

```
p {
    display:block;
    margin-block-start:1em;
    margin-block-end:1em;
    margin-inline-start:0px;
    margin-inline-end:0px;
}
```

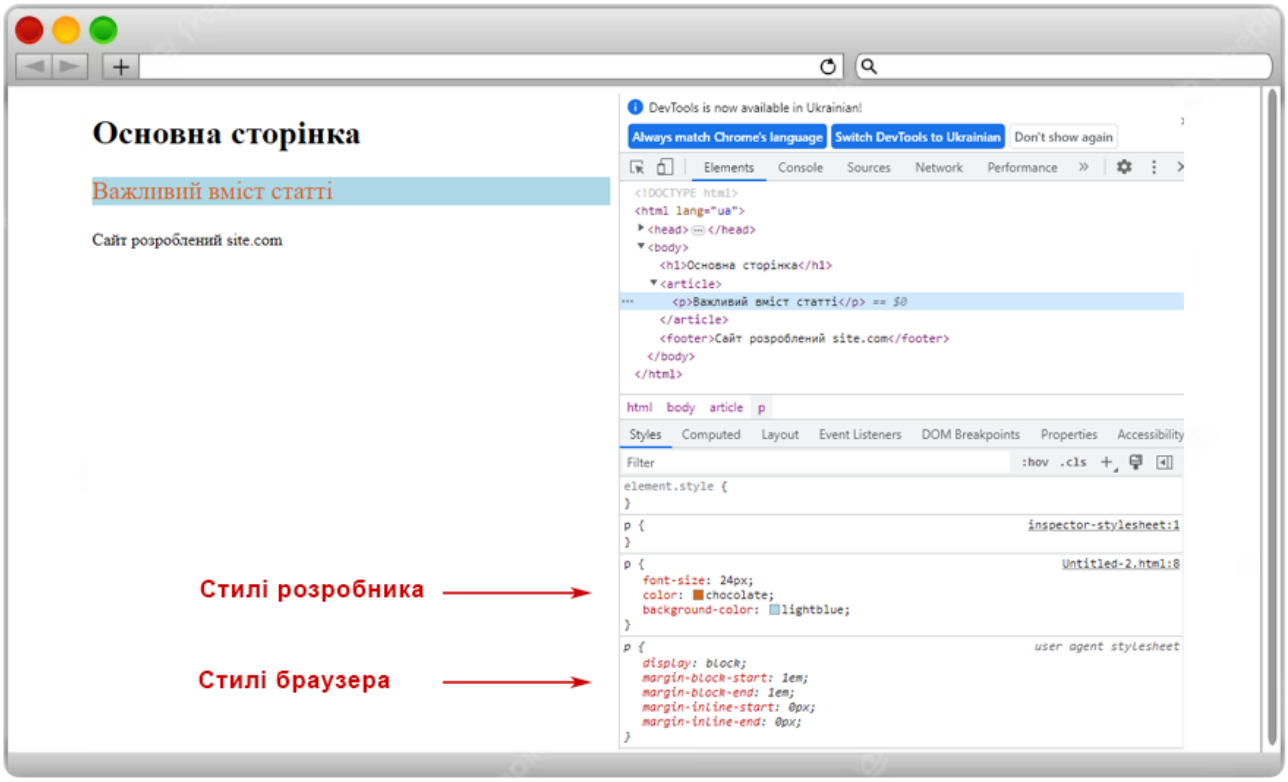

Рис. Інструменти розробника - Стилі

Якщо одне з них змінити, то спрацює пріоритет іміджу розробника.

Приклад - CSS

```
p {
    font-size:24px;
```

```
    color:chocolate;
    background-color:lightblue;
    margin-inline-start:25px;
}
```

Приклад - Консоль розробника

```
p {
    display:block;
    margin-block-start:1em;
    margin-block-end:1em;
    margin-inline-start:0px;
    margin-inline-end:0px;
}
```

Тепер абзац отримав відступ ліворуч на 25 пікселів на першому рядку.

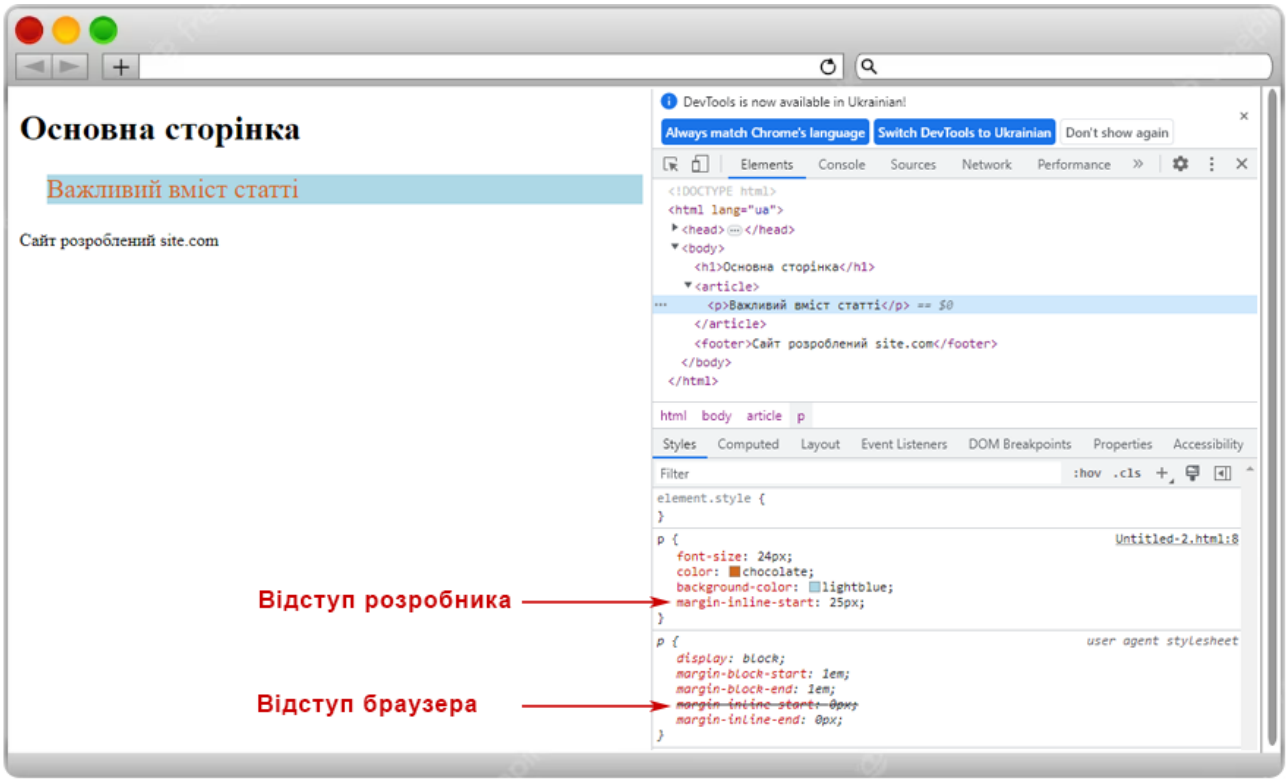

Рис. Інструменти розробника - Стилі

Специфіка (Specificity)

Відповідає за пріоритет застосування стилів до елементів залежно від типу селектора. Специфіка передбачає завдання ваг конкретним правилам. Вища його величина визначає підсумкове значення відображуваного властивості.

Ланцюжок пріоритетів виглядає так (у порядку збільшення значущості):

1. Селектори елементів та псевдоелементів (вибирається конкретний тег або його частина).
2. Селектори класів, атрибутів та псевдокласів (вибір на підставі класу, специфічного атрибуту або стану класу).

3. Селектори ідентифікаторів (оскільки задаються в єдиному екземплярі на сторінці, то мають максимальну вагу).

Рис. Пріоритетність застосування стилів до елементів

Комбінатори, глобальний селектор (зірочка), псевдоклас заперечення :not() – не впливають на специфічність. Для завдання максимального пріоритету на будь-якому рівні використовують !important , що знову ж таки не рекомендується (за рідкісними винятками).

Важливо розуміти, що незалежно від порядку оголошення селекторів у файлі стилів їх пріоритет не змінюється (за винятком рівноправних).

Приклад - HTML

```
<h1>Основна сторінка</h1>
<article id="main-content">
   <p class="blue-paragraph">Важливий вміст статті</p>
</article>
```

Приклад - CSS

```
p {
    background-color:cyan;
}
#main-content p {
    background-color:yellow;
}
article p {
    background-color:red;
}
.blue-paragraph {
```

```
    background-color:darkoliegreen;
}
```

Оскільки ідентифікатор є максимально пріоритетним, то колір фону всередині параграфа стане жовтим.

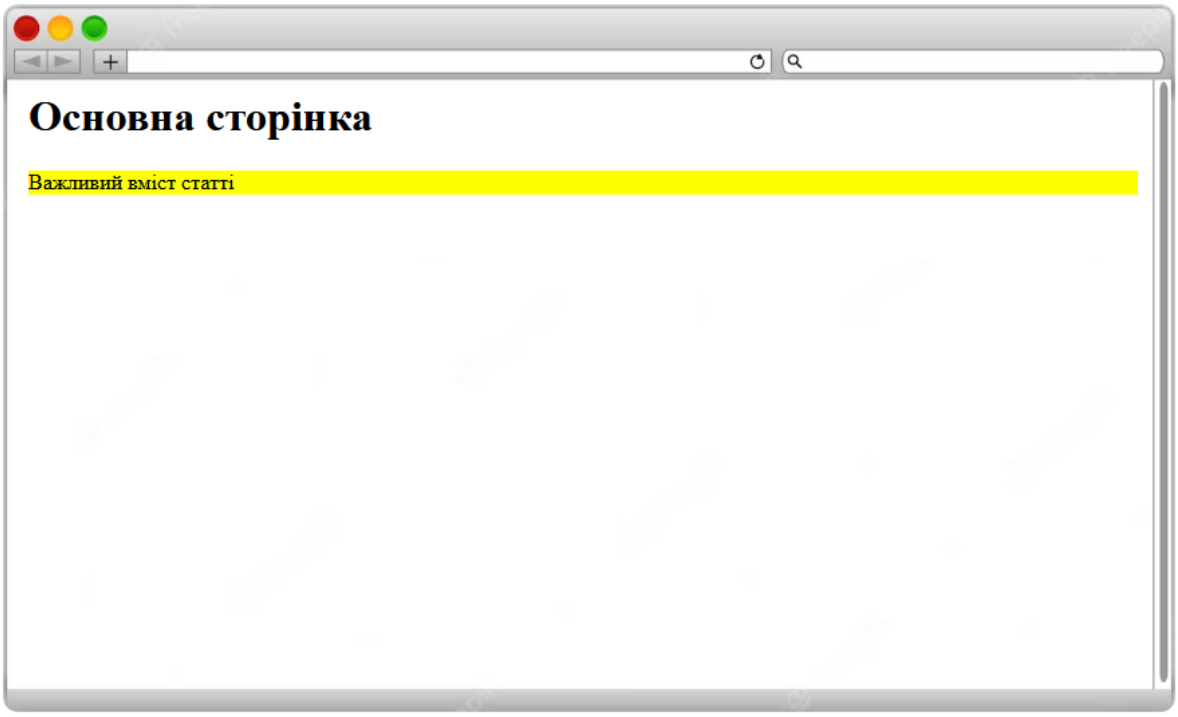

Рис. Пріоритет застосування стилів до елементів

Якщо ж будь-якому іншому селектору привласнити !important, спрацює він.

Приклад - CSS

```
p {
    background-color:cyan;
}
#main-content p {
    background-color :darkoliegreen;
}
article p {
    background-color :red !important;
}
.blue-paragraph {
    background-color :yellow;
}
```

Тепер колір змінився червоним.

Наслідування (Inheritance)

Ще один спосіб визначення кінцевих властивостей елемента - на підставі успадкування. Як відомо, DOM-дерево представлено батьками та нащадками, тому при обчисленні властивостей дочірніх елементів використовуються параметри батьків, якщо вони не вказані для нащадків.

Не всі CSS властивості за замовченням успадковуються, про що можна дізнатися з документації. До таких, наприклад, відносять межі у елементів, їх колір та товщина, відступи. Тим не менш, якщо необхідно, можна примусово задати успадкування через значення inherit.

Приклад - HTML

```
<h1>Основна сторінка</h1>
 <article>
     <p>Параграф перший</p>
     <p>Параграф другий</p>
 </article>
```

Приклад - CSS

```
article {
    border:tomato 10px solid;
}
```

Навколо тега <article> створено суцільну рамку завтовшки 10 пікселів червоного кольору. Теги параграфів, що знаходяться всередині статті, не отримали цю властивість, оскільки вона за замовченням не успадковується.

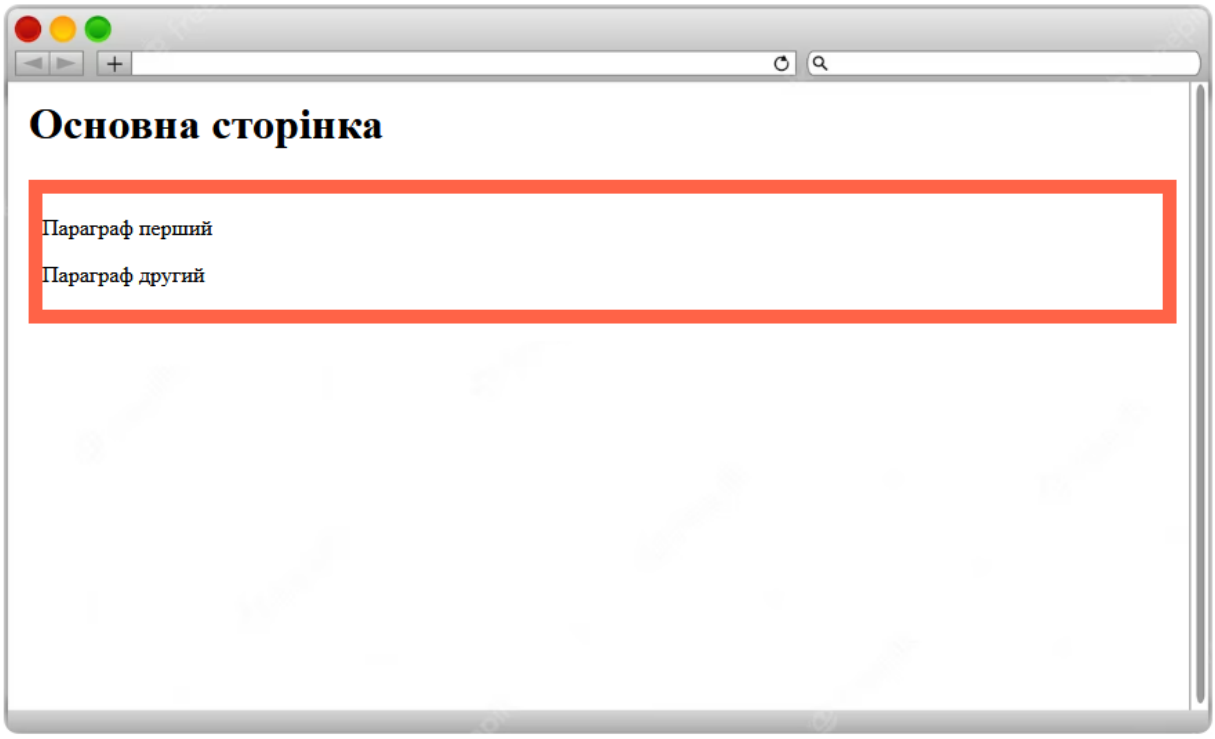

Рис. Наслідування стилів

Можна змусити їх успадковуватися.

Приклад - CSS

```
article {
    border:tomato 10px solid;
}
article p {
    border:inherit;
}
```

Тепер параграфи, що всередині тега <article>, мають самі властивості меж.

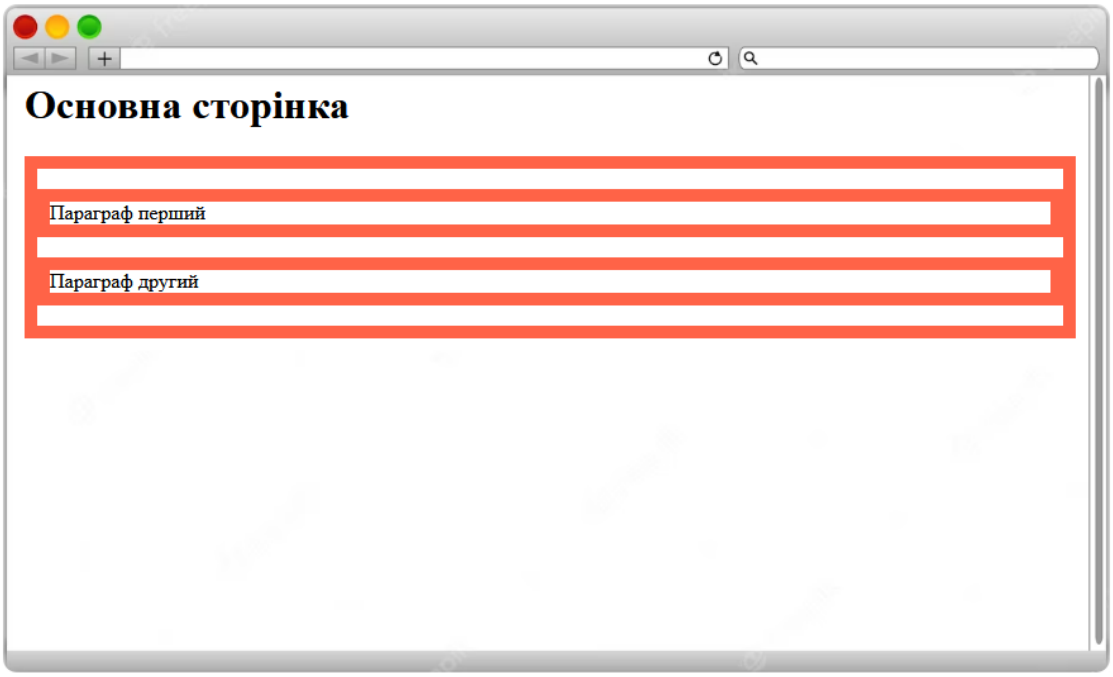

Рис. Спадкування стилів

Таким чином, при формуванні селекторів у CSS необхідно розуміти принципи їх успадкування, специфічності, каскадності. Це дозволить уникнути помилок при стилізації об'єктів і виключити зайві рядки коду (коли не зрозуміло, чому не працює та чи інша властивість, «ледачі розробники» балуються використанням !important).

Блокові (Block) та рядкові (inline) елементи

Перейдемо до особливостей відображення елементів на сторінці. Кожен верстальник розглядає структуру сторінки як набір деяких блоків. Таке бачення сайту дозволяє зрозуміти специфіку відображення об'єктів. Будь-який макет майбутнього ресурсу є набір прямокутників з певним контентом. Якщо це засвоїти, то верстка макета не складе жодних проблем.

Якщо спрощено, то при розробці всі елементи сайту умовно поділяються на 2 типи: блокові та рядкові.

Блокова (block) структура має особливості:

- Кожен елемент такого типу починається з нового рядка (наприклад, теги або <article>. Іншими словами, незалежно від того, наскільки великий вміст об'єктів, кожен з них почнеться з нового рядка).
- Цей блок прагне повного заповнення доступного простору (розширення всю область сторінки чи батьківського елемента).
- Йому можна задавати ширину та висоту (тобто властивості width і height).
- Встановлення параметрів відступів (внутрішніх або зовнішніх) або меж відсунуть інші об'єкти від поточного.

Рядкова (inline) структура характеризується:

- Перенесення на новий рядок не відбудеться, якщо залишається доступний простір на поточному рядку.

- Завдання ширини і висоти не має сенсу, оскільки ці параметри пов'язані тільки з контентом і його обсягом. Якщо рядковому елементу задати ширину в 500 пікселів, то немає жодної гарантії, що це відбудеться, особливо у випадку, якщо всередину того ж тегу помістити мало інформації.
- Застосування відступів і меж виправдане, але вони не відсунуть інші об'єкти, а лише позначаться на вмісті самого рядка.

Рис. Блокові та рядкові веб-елементи

Визначення рядковості або блочності елемента пов'язано, в першу чергу, зі специфікою використаного тегу. Завдяки CSS можна змінити тип об'єкта на той, який потрібний виходячи з цілей. Блоковий елемент легко перетворити на рядковий і навпаки. Для цього застосовується властивість display.

Приклад - HTML

```
<h1>Основна сторінка</h1>
<h2>Частина 1: Мої сайти для пошуку</h2>
<a href="https://google.com/">Головна сторінка Google.com</a>
<a href="https://www.bing.com/">Головна сторінка Bing</a>
```

Якщо не використовувати стилі, то відображення елементів на сторінці матиме дефолтний характер:<h1> і <h2> – це блоки, а посилання – рядки.

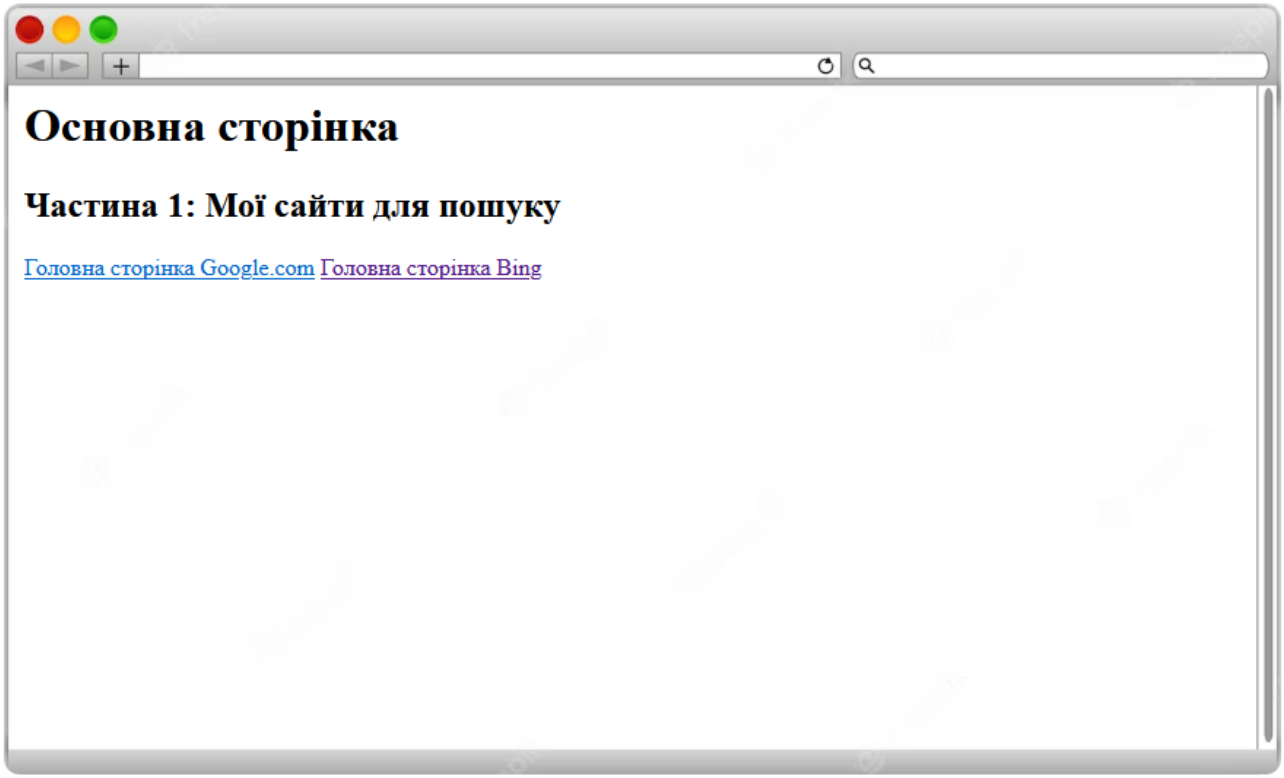

Рис. Сторінка браузера. Дефолтна поведінка елементів.

Змінимо цю поведінку.

Приклад - HTML

```
<h1>Основна сторінка</h1>
<h2>Частина 1: Мої сайти для пошуку</h2>
<a href="https://google.com/">Головна сторінка Google.com</a>
<a href="https://www.bing.com/">Головна сторінка Bing</a>
```

Приклад - CSS

```
h1, h2 {
    display:inline;
}
a {
    display:block;
}
```

Проведені маніпуляції перетворили заголовки на inline-елементи (вони тепер відображаються на одному рядку), а посилання – на блоки.

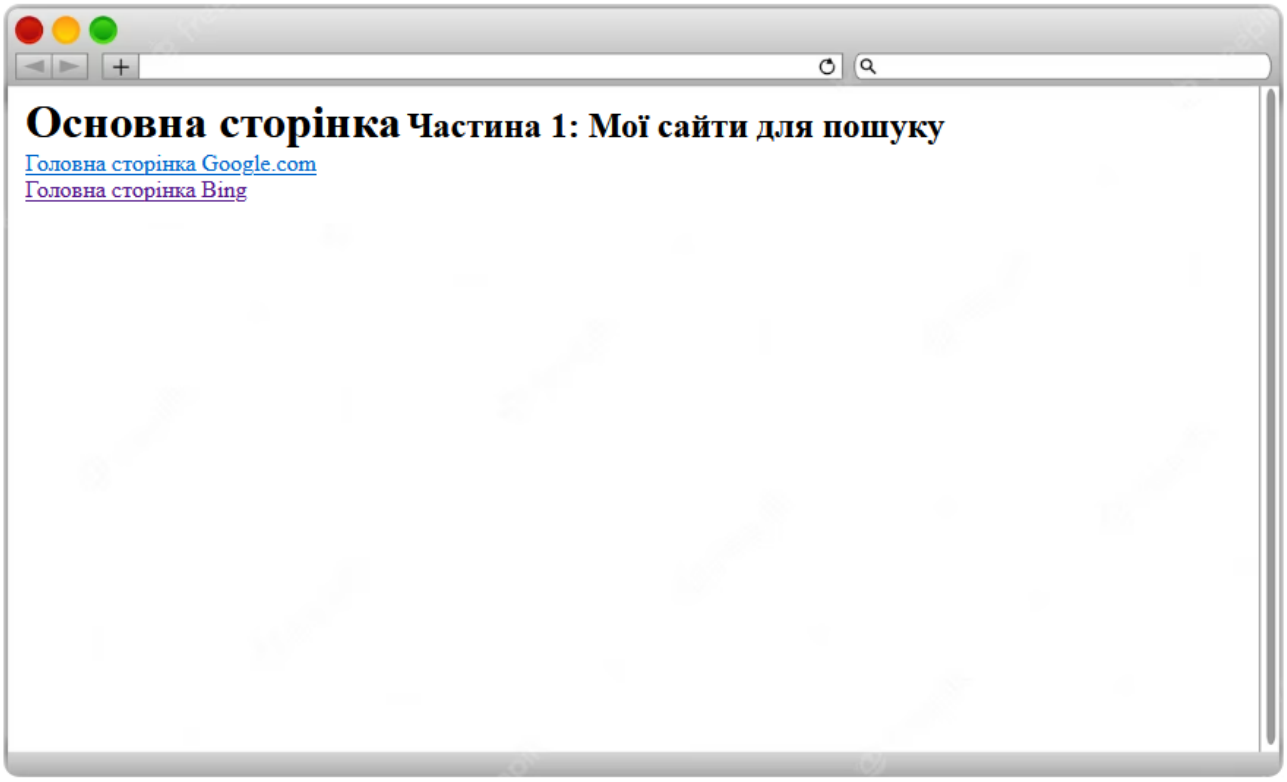

Рис. Сторінка браузера. Змінена поведінка елементів.

Також необхідно відзначити, що крім блокових і рядкових елементів можлива присутність і інших типів: флекси (flex), сітки (grid), таблиці (table), рядкові блоки (inline-block) та інші. Прямі спадкоємці таких елементів можуть набути особливої поведінки (наприклад, всі дочірні об'єкти flex-типу стануть гнучкими і будуть поводитися відповідно до правил.

Стандартна та альтернативна блокові моделі

Як згадувалося вище, всі об'єкти сторінки поводяться або як блок, або як рядок (загалом). Так чи інакше вони характеризуються набором параметрів, про які не слід забувати при формуванні макета сторінки.

Припустимо, потрібно створити блок <article> розміром 450 на 250 пікселів з відступами та межами. Ці додаткові властивості за замовченням враховуватимуться, що може розширити реальні розміри блоку. З іншого боку, CSS не забороняє задати точні розміри елемента із включенням усіх його властивостей.

Спочатку подивимося всі властивості блоку з врахуванням параметрів розмірів (показано на рисунку нижче).

Рис. Області блокового елемента

Обчислення розміру блоку здійснюється за атрибутами:

1. Параметри контенту (властивості width і height).
2. Внутрішні відступи (властивість padding).
3. Кордон, рамка (властивість border).
4. Зовнішні відступи (властивість margin).

Загальний розмір елемента, в результаті, визначатиметься на основі стандартної або альтернативної блокової моделі.

Стандартна модель

Обчислює розмір блоку як суму всіх зазначених вище параметрів. Так, якщо встановити ширину блоку, то до неї додається розмір внутрішніх і зовнішніх відступів по конкретній осі, а також величина кордону.

Приклад - HTML

```
<h1>Основи CSS - стандартна модель</h1>
```

Приклад - CSS

```
h1 {
    width:650px;
    height:300px;
    padding:10px;
    margin:20px;
    border:5px solid black;
}
```

- Властивість padding при передачі одного значення визначає внутрішні відступи від вмісту за всіма чотирма напрямками (внизу, вгорі, ліворуч, праворуч).
- Властивість border характеризує товщину рамки, її вигляд (у разі – суцільна лінія), колір.
- Властивість margin задає відступи від інших блоків сторінки за тими чотирма напрямками.
- На підставі стандартної моделі підсумковий розмір блоку буде не 650 на 300 пікселів, а наступним:
    - Ширина: 650 + 10 +10 + 5 + 5 + 20 + 20 = 720 px.
    - Висота: 300 + 10 +10 + 5 + 5 + 20 + 20 = 370 px .

У ряді випадків це незручно, оскільки замовник вимагає розміру блоку з врахуванням всіх відступів і меж.

Альтернативна модель

У такій ситуації зручна альтернативна блокова модель, яка визначає розмір блоку з врахуванням додаткових властивостей. Для цього визначається box-sizing: border-box;.

Приклад - HTML

```
<h1>Основи CSS - стандартна модель</h1>
```

Приклад - CSS

```
h1 {
    width:650px;
    height:300px;
    padding:10px;
    margin:20px;
    border:5px solid black;
    box-sizing:border-box;
}
```

В результаті параметри заголовка включатимуть внутрішні відступи і межі, але не зовнішні відступи. Їх включення до CSS не передбачено, але навіть за таких обставин обчислювати розміри блоку значно простіше:

- ширина: 650 + 20 + 20 = 690 px.
- Висота: 300 + 20 + 20 = 340 px .

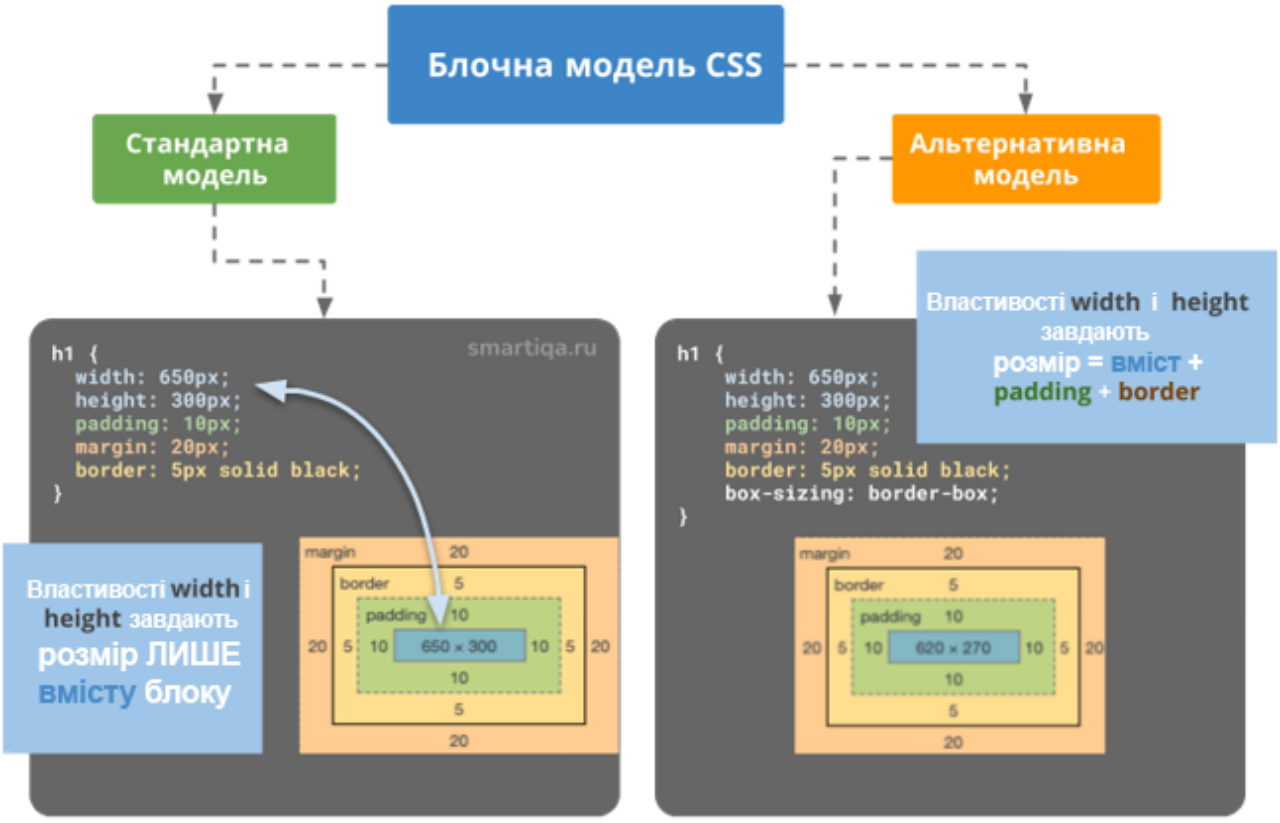

Рис. Блокова модель CSS: Стандартна та альтернативна моделі

Якщо властивість margin строго не визначено, то кінцеві розміри елемента можна вважати такими, якими вони були визначені на основі ширини і висоти.

Працюючи з рядковими елементами в частині властивостей нічого не зміниться. Зокрема, ширина і висота не мають жодного ефекту. Відступи та межі відображаються, але з деякими недоліками: вони перекриватимуть вміст інших об'єктів.

Найпростіше вирішення проблеми - inline-block . Ми показуємо вміст елемента як блок, але він не займатиме всю ширину батька. Виходить мікс рядка та блоку. В цьому випадку жодних проблем з відображенням об'єкта відповідно до задуму не буде.

Приклад - HTML

```
<strong>Відступи та межі відображаються,</strong>
<q>але з деякими недоліками:</q>
<i>вони перекриватимуть вміст інших об'єктів</i>
```

Приклад - CSS

```
q {
    width:300px;
    height:200px;
    padding:20px;
    margin:10px;
    border:4px solid black;
    display:inline-block;
}
```

Елемент <q> має висоту та ширину, відступи внутрішні та зовнішні, а також кордон у 4 пікселі. Хоч і займає всю ширину батька (тега <body>), але цілком відповідає заданим параметрам.

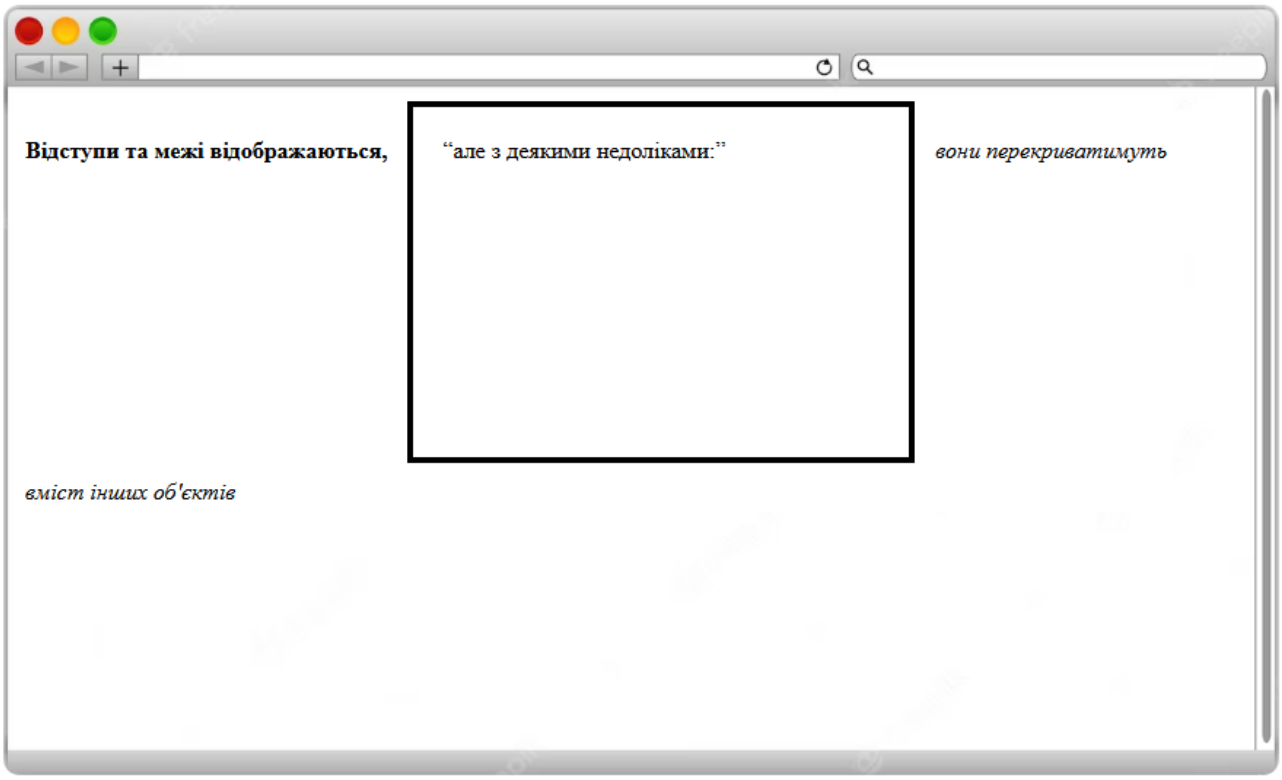

Рис. Елемент <q> із властивістю display: inline-block;

Якщо не використати декларацію display: inline-block;, то поведінка тексту з цитатою буде настільки очевидною, а параметри ширини і висоти повністю ігноруються.

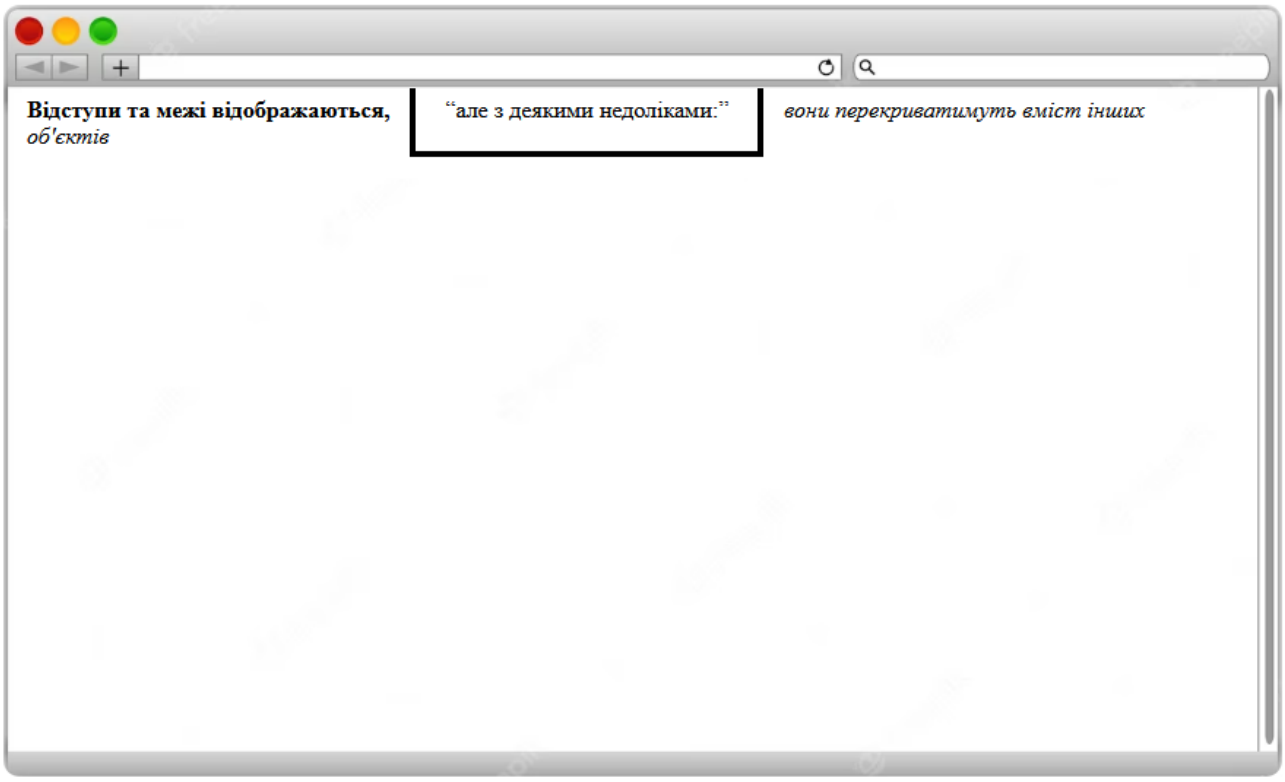

Рис. Елемент <q> БЕЗ якості display: inline-block;

Для більш детального ознайомлення з CSS рекомендується вивчати офіційну документацію (з сайту World Wide Web консорціуму). Детальний опис властивостей та їх можливих значень, а також приклади застосування доступні на сайті Mozilla.

Додаткові ресурси на тему CSS

- Специфікації та їх опис: "https://www.w3.org/Style/CSS/specs.en.html#translations"
- Посібник від Mozilla: "https://developer.mozilla.org/ru/docs/Web/CSS/Reference"

Каскадні таблиці стилів значно спрощують модифікування HTML-елементів, дозволяючи робити веб-сайти сучасними, ергономічними. Щоб ресурс не виглядав як суцільний набір текстового полотна, CSS дозволяє виділяти необхідні блоки, робити відступи, змінювати розміри шрифту і блоків, застосовувати колірне оформлення.

Запитання

1. Для чого застосовується псевдоелемент :: first-letter ?

Псевдоелемент ::first-letter застосовується виділення першого літерного чи числового символу блочного елемента (зокрема ієрогліфів). Потрібно пам'ятати, що будь-який інший символ також буде модифікований, але не буде враховуватися як перший (як і всі спеціальні знаки після першого валідного знака).

Приклад – HTML---

```
<p>Просто абзац якогось тексту</p>
<p>)1..... Модифікується і дужка, і цифра 1, та ще й усі крапки!</p>
```

Приклад – CSS---

```
p::first-letter {
color: red;font-size: 40px;}
```

У першому абзаці червоним кольором і збільшеним шрифтом буде наділена перша літера – П, а у другому, крім цифри 1, зміниться дужка і всі наступні точки до літери М.

2. Для чого застосовуються псевдоелементи :: after , ::before ?

Псевдоелементи ::after та ::before найчастіше дозволяють додати косметичні прикраси до та після батьківського елементу.

Приклад - HTML -

```
<a href="https://yandex.ru/">Яндекс</a>
```

Приклад - CSS---

```
a::after {
      content: "🧡";
}
a::before {
      content: "🩶";
}
```

Тут ми виділили посилання різнокольоровими сердечками.

3. Якого розміру буде шрифт (у пікселях) елемента <p> у наведеному прикладі?

Приклад - HTML----

```
Смеркало, хоч і здавалося, що на вулиці дуже тепло</p>
```

Приклад - CSS---

```
#txt {font-size: 10px;}
.txt {font-size: 22px;}
p {font-size: 60px;}
```

Відповідно до пріоритетів у спадкуванні властивостей CSS вибудовується наступна послідовність (у міру зростання значущості):

- 60 пікселів у тега <p>;
- 22 пікселя у класу txt;
- 10 пікселів у ідентифікатора txt;
- Завдання внутрішнього стилю за допомогою атрибуту style - 42 пікселя.

Виходить, підсумковий розмір шрифту абзацу становитиме 42 px, оскільки інлайн-стиль має максимальний пріоритет.

Завдання

Створіть 3 посилання (на будь-які пошукові системи), кожна з яких буде починатися з нового рядка, мати суцільний зелений кордон завтовшки 7 пікселів та сумарний розмір 300 на 200 пікселів. Вертикальний відступ між посиланнями становитиме 10 пікселів.

Рішення

Наше завдання – мінімізація рядків коду. Тому застосовувати будь-які розриви рядків (за допомогою тега
) буде не логічним.

Оптимальне рішення – надання всім посиланням режиму відображення block . Також слід врахувати, що зовнішні відступи між об'єктами підсумовуються (оскільки в одного об'єкта є, наприклад відступ знизу, а у наступного за ним - відступ зверху).

Приклад HTML

```
<a href="https://www.yahoo.com/">Yahoo</a>_ _
<a href="https://yandex.ru/">Яндекс</a>_ _
<a href="https://www.google.com/">Google</a>_ _
```

Приклад - CSS

```
a {
    width:300px;
    height:200px;
    margin:5px;
    border:7px solid green;
    display:block;
    box-sizing:border-box;
}
```

Оскільки посилання – рядкові елементи, їх слід перетворити на блокові. А щоб не вираховувати із ширини та висоти розміри кордону, зручно скористатися декларацією box-sizing: border-box;

З врахуванням підсумовування зовнішніх відступів потрібно задати їх величину у розмірі 5 пікселів (що призведе до потрібних десяти, якщо підсумувати margin посилання та її сусіда).

Джерела інформації

1. Сучасний CSS. Основні поняття, селектори, блокова модель https://smartiqa.ru/courses/web/lesson-4-css