10.4. CSS. Типи даних та робота з HTML-елементами

Типи даних, що використовуються в CSS. Способи завдання розмірів елементів. Особливості роботи з різними HTML-елементами (текстовими, зображеннями тощо) та завдання їм загальних властивостей.

1. Типи даних у CSS
2. Типи даних у CSS. Цілі числа (integer)
3. Типи даних у CSS. Десяткові дроби (number)
4. Типи даних у CSS. Вимірювання (dimension)
    a. Довжина
    b. Кути
    c. Час
    d. Дозволи
5. Типи даних у CSS. Відсотки
6. Типи даних у CSS. Колір
    a. Ключові слова
    b. Шістнадцяткові RGB-кольори
    c. Застосування функцій rgb() та rgba()
    d. Застосування функцій hsl() та hsla()
7. Типи даних у CSS.
8. Типи даних у CSS. Позиція (position)
    a. Одне значення
    b. Два значення
    c. Три значення
    d. Чотири значення
9. Типи даних у CSS. Рядок
10. Типи даних у CSS. Функції11. Підходи до завдання розмірів елементів у HTML-документах
    a. Природний розмір
    b. Заданий розмір
    c. Розміри у відсотках
    d. Мінімальні та максимальні розміри
    e. Розміри в залежності від величини вікна браузера
11. Працюємо з HTML-елементами: загальні властивості
    a. Фон
    b. Межі
    c. Перекриття вмісту (overflow)

Типи даних у CSS

При написанні правил каскадних таблиць стилів можна використовувати різні типи даних. Залежно від властивостей та функцій їх значення різняться. Деякі є специфічними для конкретних атрибутів, інші більш універсальні.

У наведених прикладах буде повторюватися базовий шаблон HTML-сторінки (його необхідно один раз створити і всередині нього розміщувати наведені коду). Вигляд такого шаблону показаний нижче (index.html).

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="css/style.css">
      </head>
      <body>
            ….
      </body>
</html>
```

Для відображення CSS разом з html-документом створюємо папку css, всередині якої розташується зовнішній файл зі стилями style.css. Додатково створити папку img, в яку кладуться потрібні зображення. При написанні коду варто перевірити правильність шляху до файлів з зображеннями та стилями.

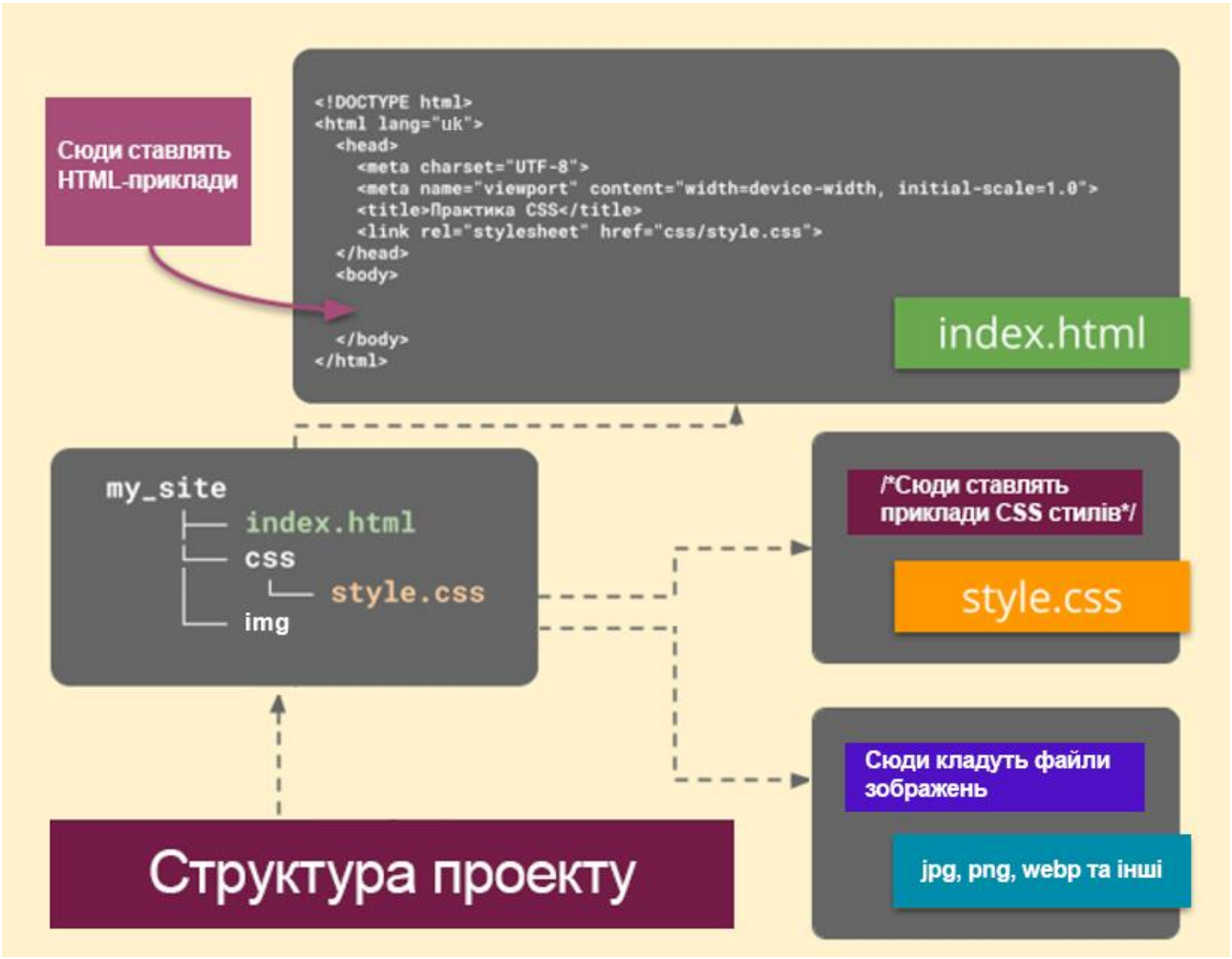

Рис.1 Структура простого проекту

Типи даних у CSS. Цілі числа (integer).

Є окремим випадком типу number. Сюди включають як додатні, і від'ємні числа. Застосовуються для обмеженого кола CSS-властивостей (z-index, column-count, counter-increment, grid-row тощо).

Для завдання значень для перелічених атрибутів оперують десятковими числами (інші системи обчислення CSS не передбачено). Обмежень у числовому діапазоні немає, хоч і не рекомендується використовувати величезні величини (щоб максимально охопити всі браузери не потрібно переходити межу у 100 000).

Допустимі варіанти цілого типу даних:

приклад

```
10
+30
-22
+0
```

Знак плюса не заборонено опускати. Декілька прикладів неправильного завдання цілих чисел:

приклад

```
/* Це вже варіант типу number, а не integer */
-20.0
/* Допустимо лише один символ + або - */
+-75
/* Наукові позначення таблиці стилів не вміють інтерпретувати */
2e7
```

Як приклади роботи з цілим типом даних розглянемо такі характеристики:

- Властивість column-count
- Властивість z-index

Властивість column-count

Ця властивість розділяє текстовий вміст блоку на кілька стовпців. Воно успадковане від друкованих газет, де текст йде не суцільним полотном на всю ширину сторінки, а поділяється на вузькі стовпчики, що зручніше при читанні. Може приймати тільки додатні значення та auto (у разі кількість стовпчиків обчислюється виходячи з властивості column-width).

Приклад - HTML

```
<article>
    Стаття розділена на кілька колонок, що спрощує її візуальне
    сприйняття. Незважаючи на те, що кількість тексту не змінюється, він
    сприймається значно легше. Такою є психологія людського мозку. І хоча
    друкованих газет сьогодні ми читаємо не так багато, на підсвідомому
    рівні ми звикли до такого подання текстової інформації. Правила такі:
    дані всередині блоку рівномірно поділяються на стовпці з відступами,
    розподіляючись за ними максимально збалансовано.
</article>
```

Приклад - CSS

```
article {
      column-count: 5;
}
```

Якщо на екрані смартфона читати текст, що займає всю ширину екрана, не складно, то на великих моніторах зручніше сприймати багатоколонковий контент.

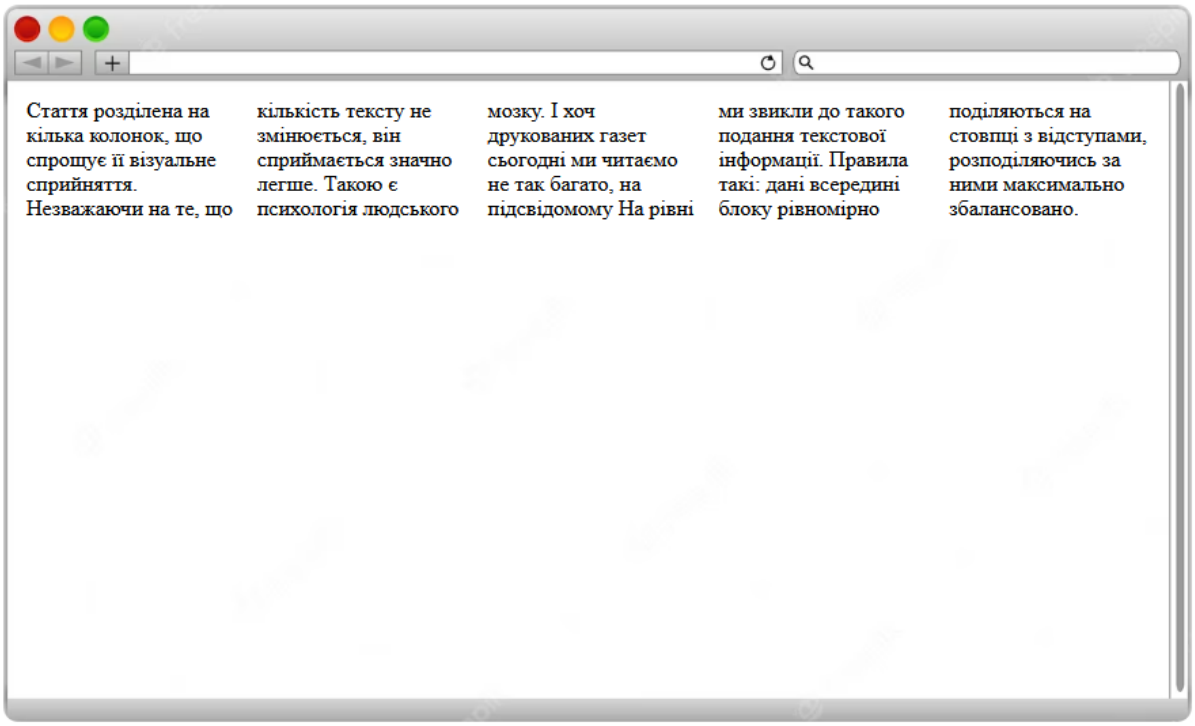

Рис.2. Подання тексту у кількох колонках

Властивість z-index

Ця властивість може мати як додатні, так і від'ємні значення. З його допомогою позиціонують елементи по глибині. Це певна імітація третього виміру (об'єкти перекривають одне одного, частина з яких переноситься на передній план, інші – на задній).

Вища величина якості наближає елемент до верху, висуває його на передній план. Застосовується лише для об'єктів, яким задано позиціонування.

Приклад - HTML

```
<section>
      Фіолетовий блок
      <p class="red-box">Червоний блок</p>
      <p class="orange-box">Оранжевий блок</p>
</section>
```

Приклад - CSS

```
section {
      position: relative;
      z-index: - 1;
      border: brown dotted;
```

```
      height: 200px;
      margin: 30px 10px;
      background-color: cornflowerblue;
      padding: 25px;
      width: 400px;
}
.red-box {
      position: absolute;
      z-index: 3;
      border: grey solid;
      background-color: firebrick;
      width: 60%;
      left: 100px;
      top: 40px;
      color: ghostwhite;
      height: 150px;
      padding: 12px;
}
.orange-box {
      position: absolute;
      z-index: 5;
      border: navy double;
      background-color: orange;
      padding: 10px;
      width: 40%;
      left: 300px;
      top: - 15px;
      height: 90px;
      opacity: 0.6;
}
```

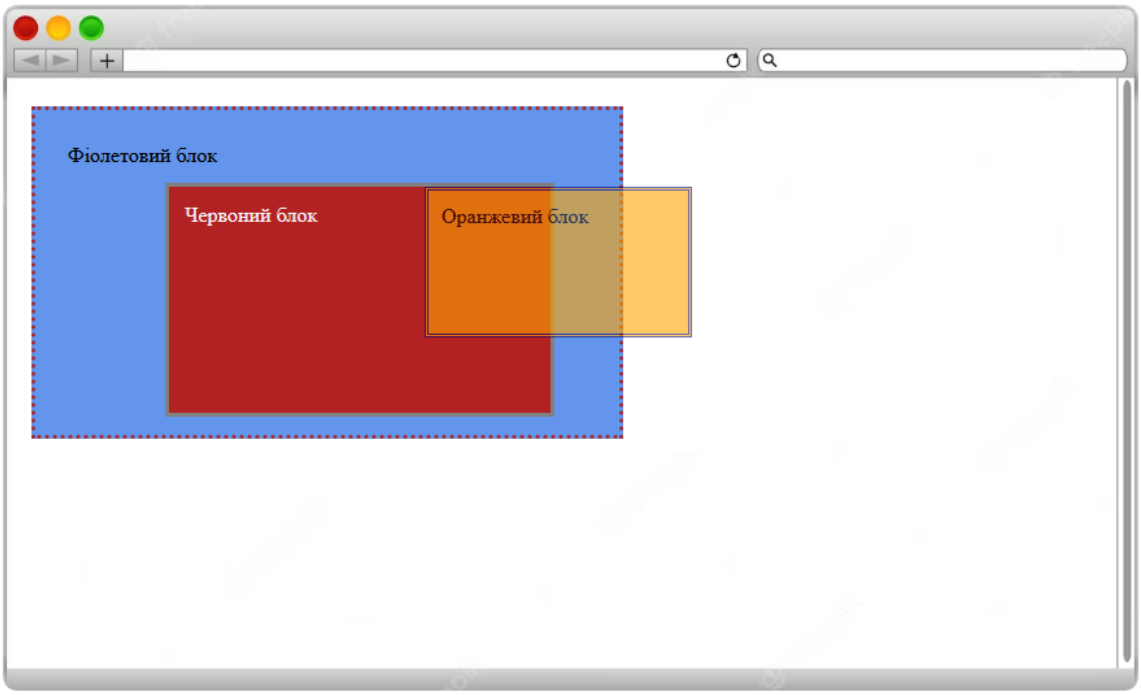

Рис.3. Результат роботи властивості z-index із застосуванням позиціонування блоків

1. Елементу section привласнено найменше значення z-index, що відсунуло його на задній план. Абзац із класом orange-box знаходиться найближче до верху і перекриває інші об'єкти (для наочності йому задана прозорість за допомогою opacity). Блок, що залишився, розташований посередині (див. рис).
2. Відносне позиціонування (position: relative) тега <section> дозволяє розмістити всередині нього нащадків (у даному прикладі – абсолютно) за допомогою відступів (left, top, bottom, right).
3. Величина атрибуту z-index може бути будь-якою, а порядок накладання блоків визначатиметься таким чином: чим менше число, то далі від «спостерігача» знаходиться цей об'єкт. У деяких випадках для впевненості того, що елемент завжди розташується попереду, йому надають велике значення (наприклад, 1000).

Типи даних у CSS. Десяткові дроби (number)

Це значно ширший діапазон значень, що містить також й цілі числа. Як роздільник вживають крапку. Після неї може бути будь-яка кількість цифр (але не менше однієї).

Допустимих варіантів написання тут більше, ніж у разі завдання цілих чисел:

приклад

```
/* Є і цілим, і десятковим числом */
101
/* Додатнє */
2.12
/* Від'ємне */
-7.1
/* Нуль (можна писати як з плюсом, так і з мінусом, але простіше взагалі без знаку) */
+0.0
```

```
/* Наукове позначення чисел зі ступенями */
3.1e7
/* Допускається пропуск нуля на початку */
.35
```

Конкретних властивостей, які у чистому вигляді застосовують дроби, нині у CSS практично немає (крім ступеня прозорості – opacity). Тим не менш, такі числа зустрічаються у функціях (наприклад scale()). Не заборонено задавати ширину чи висоту за допомогою чисел із плаваючою точкою (хоч це й не прийнято).

Приклад - HTML

```
<div>Стандартний блок</div>
<div class="scaled">Масштабований блок</div>
```

Приклад - CSS

```
div {
      width: 500px;
      height: 200px;
      background-color: hotpink;
      margin-left: 100px;
}
.scaled{
      transform: scale(1.3 - 0.6);
      background-color: mediumturquoise;
}
```

Кожен тег спочатку має однакові ширину та довжину. До останнього блоку застосовано властивість transform з функцією scale(). Як перший параметр вона приймає величину розтягування по осі X, а як другий – по осі Y. Якщо абсолютне значення будь-якого аргументу більше за одиницю, то елемент збільшиться, інакше – зменшиться. Від’ємні числа не лише змінюють об'єкт, а й відбивають його щодо заданої осі (див. рис).

Рис.4. Використання дробових чисел у функції scale()

Типи даних у CSS. Вимірювання (dimension)

У CSS до цього типу даних відносять число, до якого прикріплена одиниця виміру. Це цілий клас, що містить довжину (length), кути (angle), час (time), роздільність (resolution).

Довжина

Без цього числового типу не обходиться жоден верстальник, оскільки завдання розмірів – операція, що часто зустрічається.

Виділяють абсолютні та відносні одиниці визначення довжини.

Абсолютні включають:

- сантиметри (11cm);
- міліметри (108mm);
- чверть-міліметри (200Q);
- дюйми (4in);
- піки (17pc);
- точки (212pt);
- пікселі (310px).

Піксельний еквівалент кожної одиниці можна знайти в офіційній документації. Вони часто використовуються в друку, тому враховуються при верстці.

До відносних величин вдаються у разі необхідності масштабування елементів на різних пристроях (щоб і на смартфоні, і на моніторі було зручно переглядати веб-ресурс).

Відносні одиниці довжини:

Одиниця	Опис
em	Враховують розмір шрифту батьківського елемента, в якого за замовчуванням значення 1em.
ex	Розтягують шрифт щодо висоти символу «x» у нижньому регістрі.
ch	Як зразок використовується символ "0", нуль. Якщо браузеру не вдається виявити його розмір, базово задаються йому такі розміри: ширина 0.5em, а висота 1em.
rem	Є величиною базового значення шрифту кореневого елемента сторінки – <html>. Щодо нього задаються розміри для інших елементів. За замовчуванням бере розмір, що встановлений у браузері за замовченням (зазвичай, 1rem=16px).
lh	Розраховується щодо висоти рядка блоку.

Одиниця	Опис
vw	Це 1% від ширини області перегляду браузера, де 100vw – повна ширина. При зміні розмірів вікна браузера автоматично міняються відповідні блоки
vh	Аналог vw, але щодо висоти області перегляду браузера.
vmin	Відбувається порівняння величин vw і vh, після чого вибирається мінімальне їх значення як базове значення.
vmax	Також порівнює vw та vh, але бере максимум з них.
rlh, ic, vi, vb, cap	Відносні одиниці довжини, які використовуються у вузькому колу завдань.

Приклад - HTML

```
<h1>Експериментуємо зі шрифтами</h1>
<span>успадковується від html</span>
<p>Текст в абзацах дещо зменшений порівняно з іншими блоками</p>
```

Приклад - CSS

```
html {
      font-size: 25px;
}
h1 {
      font-size: 1.5rem;
}
p {
      font-size: 0.8rem;
}
```

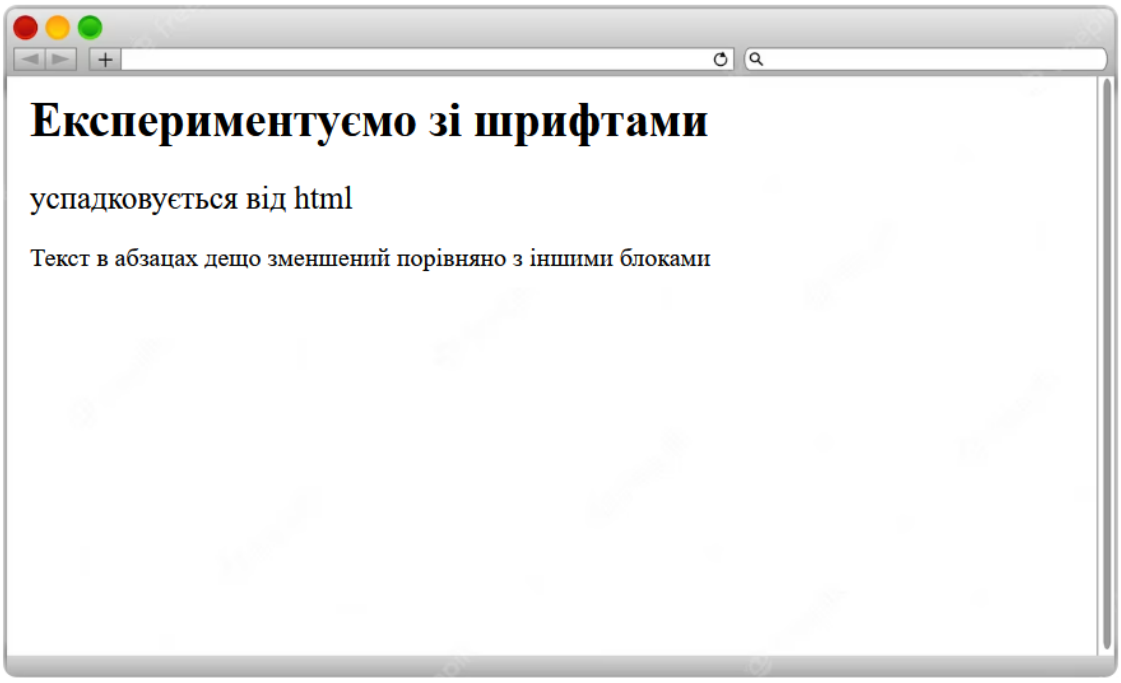

Рис.5. Абсолютні та відносні розміри шрифтів

Базовий шрифт для сторінки визначено у розмірі 25px. Усі заголовки першого рівня збільшені щодо нього на 50 %, а текст всередині абзаців зроблений дрібнішим.

Приклад - HTML

```
<section>
      Преамбула
      <header>Заголовна частина</header>
      <article>Основний вміст</article>
      <footer>Підсумкова частина</footer>
</section>
```

Приклад - CSS

```
section {
      font-size: 3vw;
}
header {
      font-size: 5vw;
}
article {
      font-size: 4vw;
}
footer {
      font-size: 2vw;
}
```

На сторінці з представленим форматуванням тексту на будь-якому пристрої буде зручно читати його, оскільки він обчислюється відносно ширини вікна браузера. Якби ми задали його величину в пікселях, то за різних дозволів або розмірів пристроїв ергономічність сайту постраждала б.

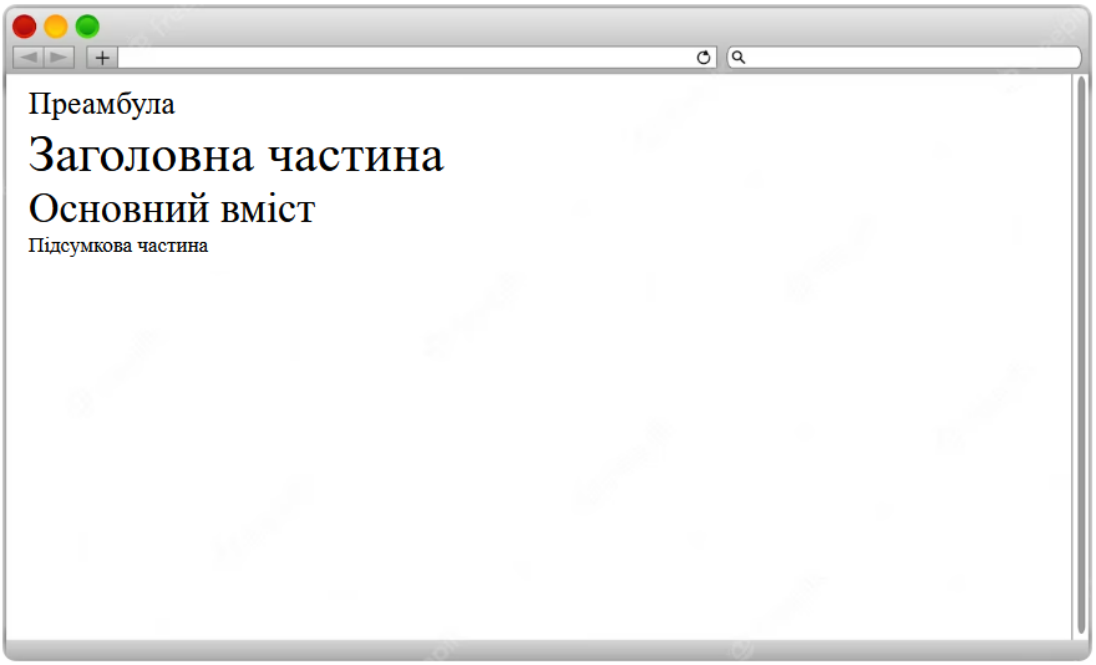

Рис.6. Менше ширина екрану – менше текст

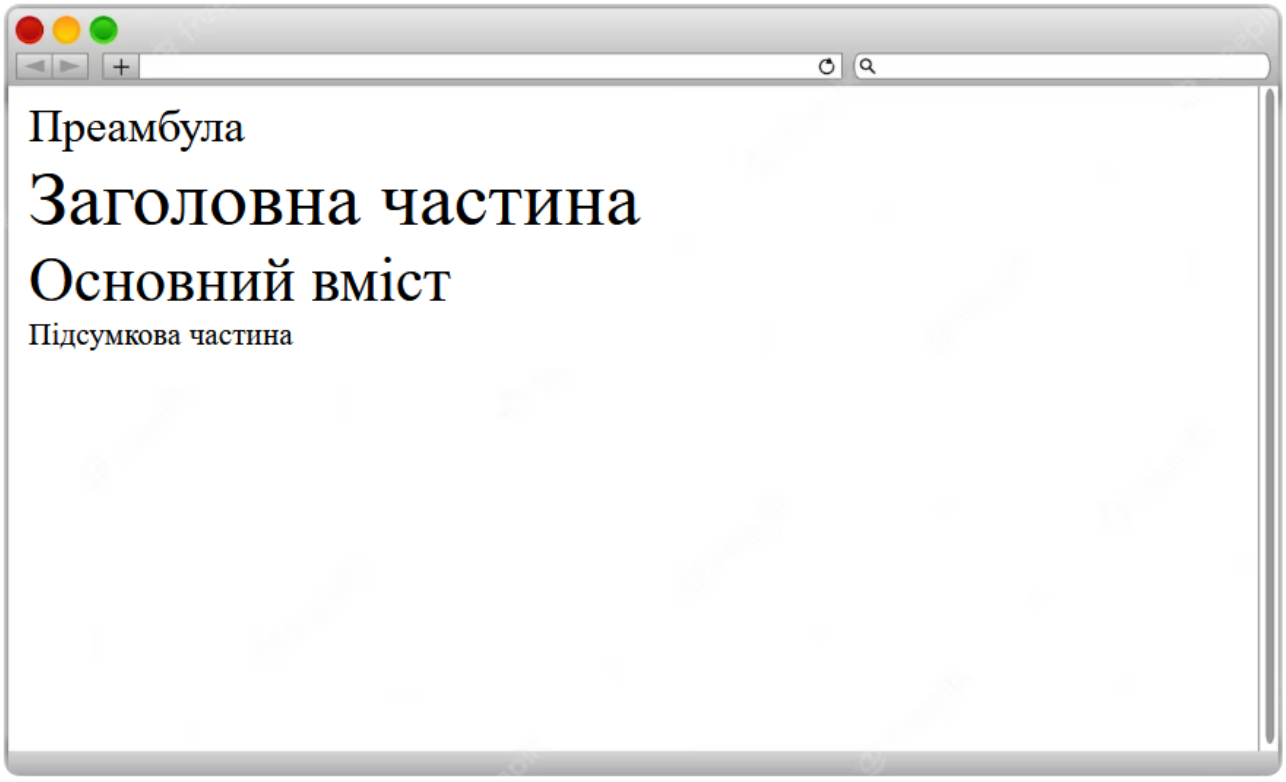

Рис.7. Збільшили ширину екрану – збільшився текст

Кути

У CSS передбачені такі заходи кутів: градуси (deg), радіани (rad), гради (grad) і повороти (turn).

Співвідношення одиниць

180deg = 200grad = 0.5turn = 3.142rad.

Донедавна їх практично не використовували, оскільки не всі браузери підтримували властивості, в яких вони використовуються. Сьогодні таких проблем немає, але застосування операцій з градусами досить обмежене: властивість transform та побудова градієнтів.

Приклад 1:

Розгортаємо градієнт на 90 градусів

Приклад - HTML

```
<div></div>
<div class="deg"></div>
```

Приклад - CSS

```
div {
	margin: 2vw;
	width: 80vw;
	height: 20vh;
	background: linear-gradient(blue, green);
}
.deg {
	background: linear-gradient(90deg, blue, green);
}
```

Для другого блоку задано поворот на 90 градусів, що змінило напрямок градієнта.

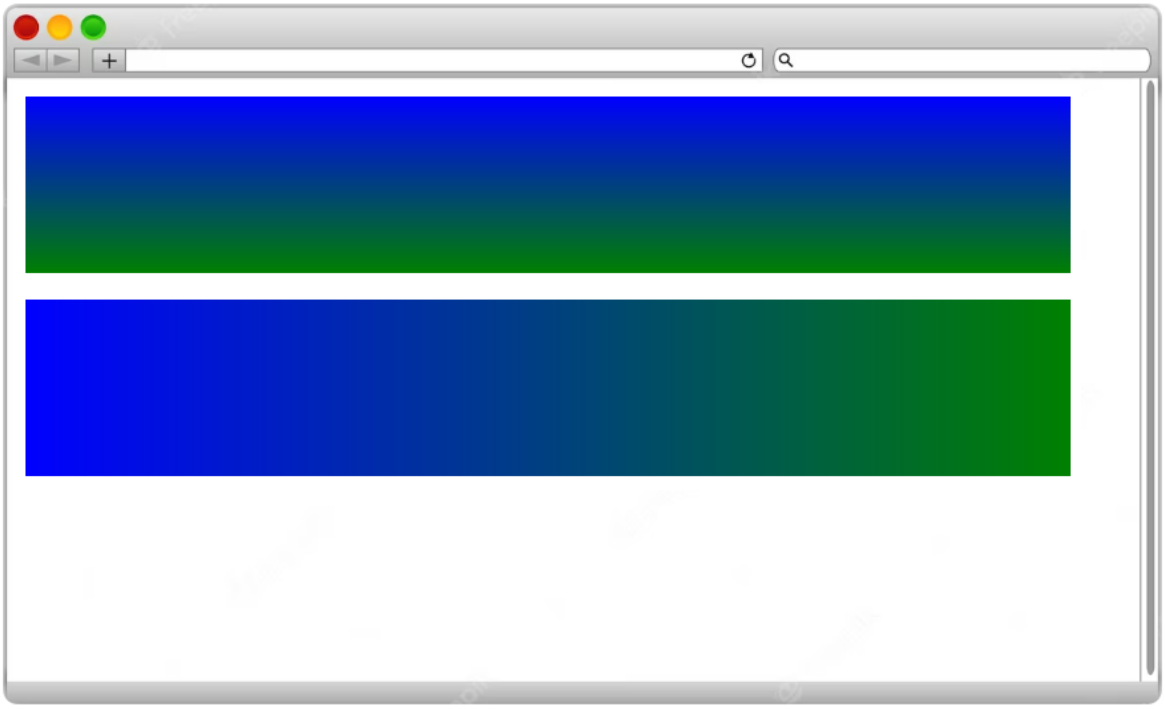

Рис.8. У другого прямокутника градієнт повернутий на 90 градусів

Приклад 2. Повернути блок на чверть повороту за годинниковою стрілкою.

Приклад - HTML

```
<div>Звичайний блок</div>
<div class="rotated">Блок з поворотом на 90 градусів</div>
```

Приклад – CSS

```
div {
	width: 20vw;
	height: 20vh;
```

```
        background-color: palegreen;
        margin: 1vw;
        padding: 1vh;
}
.rotated {
        transform: rotate(0.25turn);
        background-color: mistyrose;
}
```

Розміри обох елементів однакові, але другий повернуто на чверть повороту за годинниковою стрілкою.

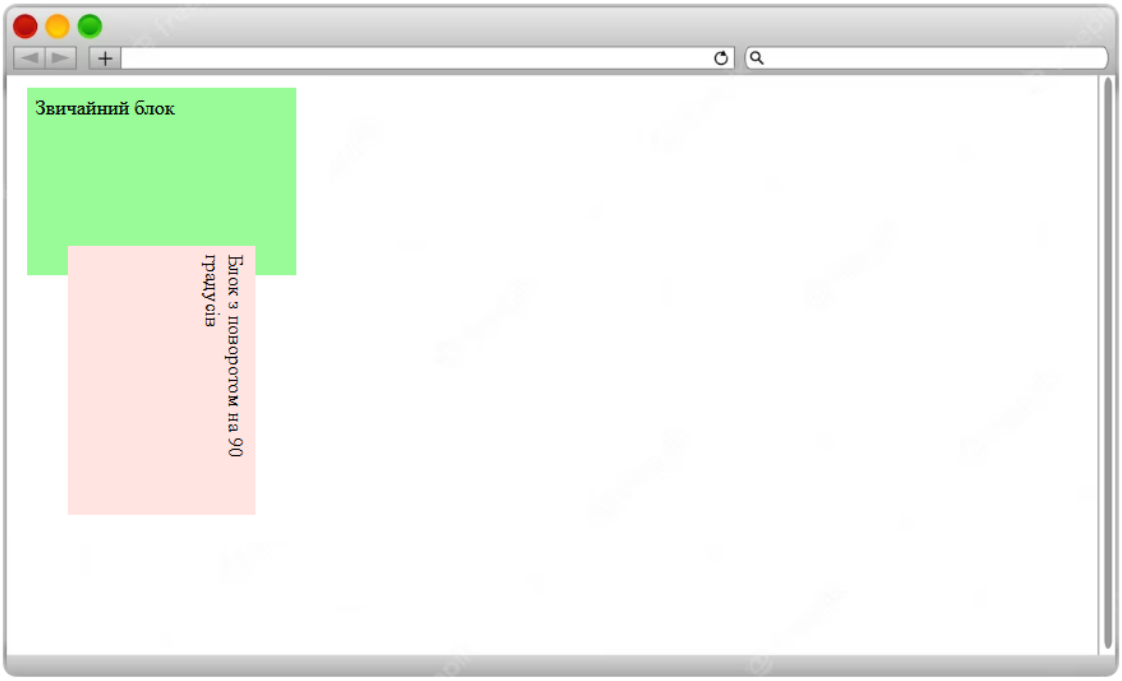

Рис.9. Другий блок повернутий на чверть повороту за годинниковою стрілкою

Час

Використовується лише дві одиниці часу: секунди (s) та мілісекунди (ms). Актуально для анімацій та трансформацій, щоб візуально було видно цей процес.

Приклад - HTML

```
<div>Це блок</div>
```

Приклад - CSS

```
div {
        transition: 0.5s;
        width: 15vw;
        height: 15vh;
        background-color: palegreen;
        margin: 10vw;
        padding: 1vh;
}
```

```
div:hover {
      transform: skew(70.2grad, -0.21rad);
      background-color: mistyrose;
}
```

У даному прикладі при наведенні миші на елемент він дещо розтягується та скручується. Завдання оголошення transition: 0.5s дозволило побачити весь процес не миттєво, а плавно протягом пів-секунди.

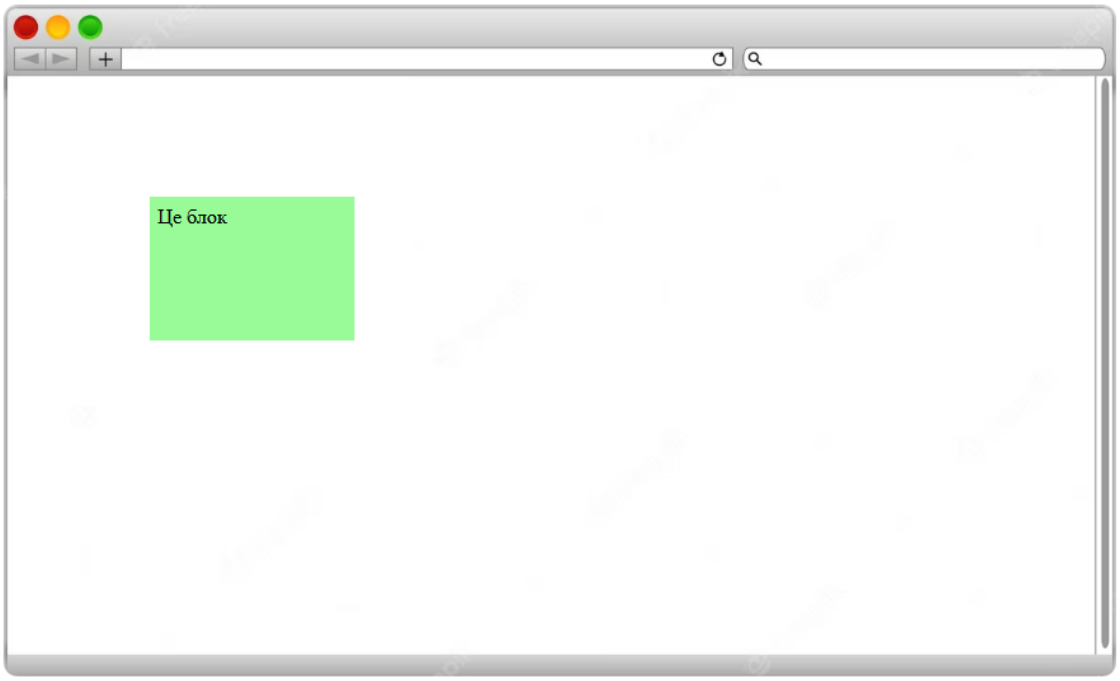

Рис.10. Початковий стан блоку

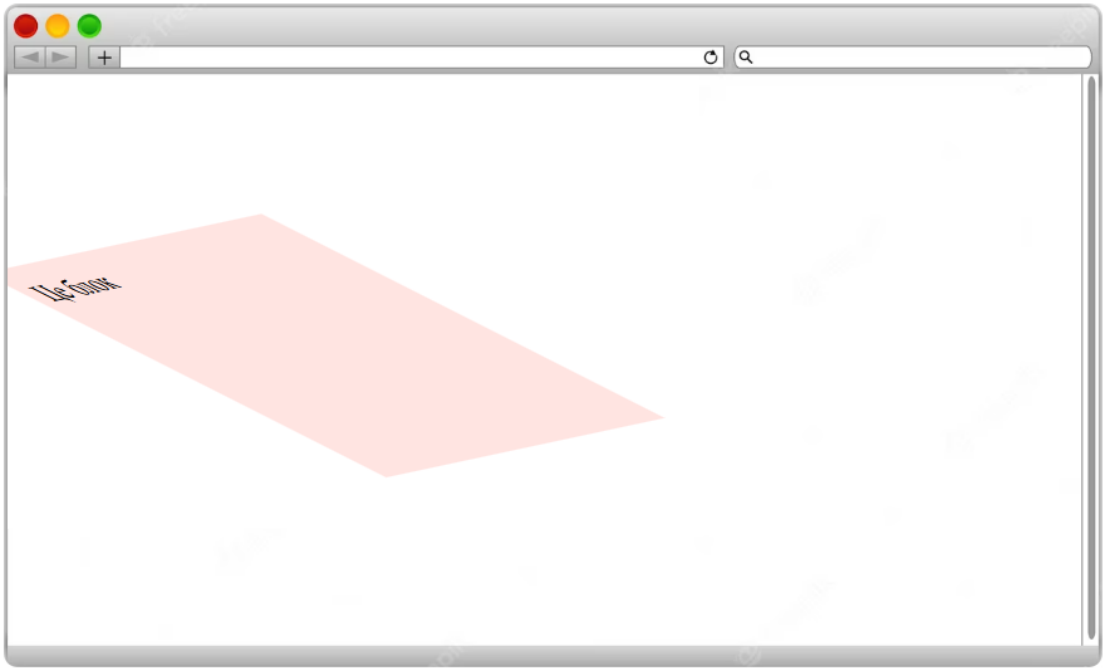

Рис.11. Розтяжка блоку при наведенні

Роздільність

Дані одиниці актуалізувалися з появою якісних екранів у смартфонів та пристроїв високої роздільної здатності (наприклад, дисплеїв retina). Оскільки кількість точок на моніторах різна, розробники хочуть, щоб користувачі сайту отримали максимально чітку картинку незалежно від гаджета.

Виділяють такі одиниці роздільності:

- dpi (кількість точок на дюйм);
- dpcm (кількість точок на сантиметр);
- dppx (кількість точок на піксель).

Приклад - HTML

```
<div>Перевір свій пристрій</div>
```

Приклад - CSS

```
div {
      width: 15vw;
      height: 15vh;
      background-color: teal;
      margin: 10vw;
      padding: 1vh;
      text-align: центр;
      border: tomato 5px double;
      color: whitesmoke;
}
@ media screen and (min-resolution: 200dpi) {
      div {
      background-color: midnightblue;
      }
}
```

На екрані монітора та стандартного смартфона колір блоку буде різним, оскільки щільність точок на екранах телефонів вища, що ми і врахували у цьому умовному прикладі.

Типи даних у CSS. Відсотки.

Застосовні завдання розмірів найчастіше і розраховуються виходячи з батьківського елемента. Дозволяють присвоїти ширину та висоту, відступи або попрацювати зі шрифтами.

Приклад - HTML

```
Опис сайту
<section>
      Блок сайту
      <article>Вміст блоку</article>
</section>
```

Приклад - CSS

```
body {
      height: 100vh;
      background-color: silver;
}
 section {
      margin: 3%;
      font-size: 90%;
      width: 70%;
      height: 60%;
      background-color: moccasin;
}
article {
      font-size: 140%;
      width: 80%;
      height: 40%;
      background-color: thistle;
}
```

Перед нами наочна демонстрація роботи відсотків як із шрифтами, так і з блочними елементами. Тег <section> успадкував розміри від <body>, тоді як тег <article> – від <section>, тобто. свого прямого батька.

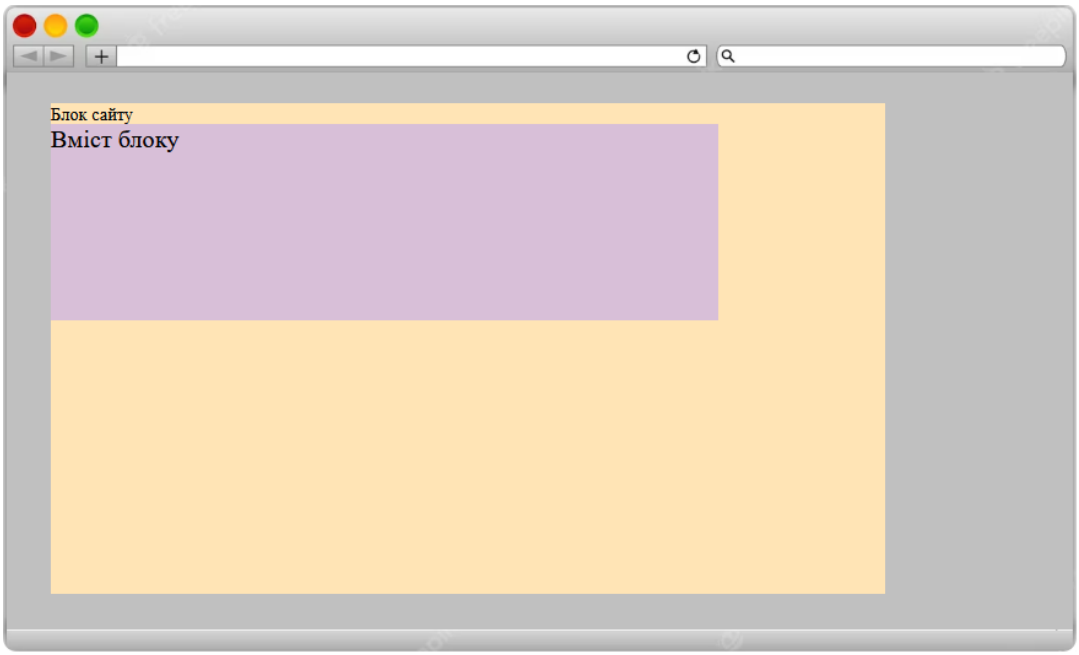

Рис.12. Початковий стан блоку

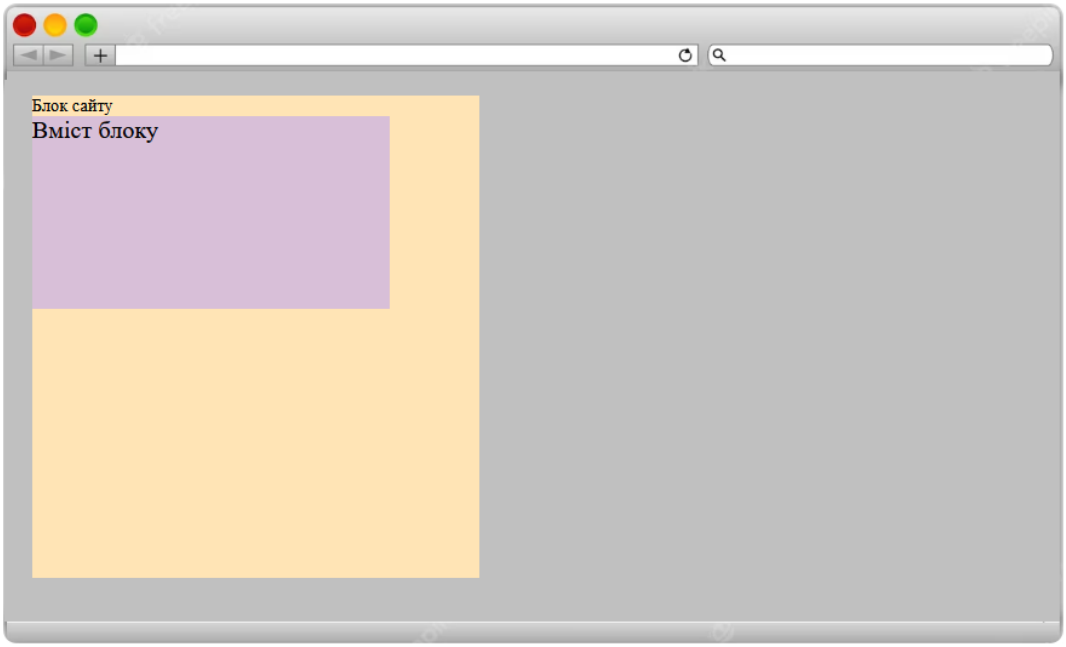

Рис.13. Довжина блоків пропорційно збільшується при розтяжці батьківського блоку

Типи даних у CSS. Колір.

Значна частина HTML-елементів може мати суцільне або градієнтне забарвлення, у тому числі із встановленням ступеня прозорості.

Загалом колірні значення визначаються такими способами:

Ключові слова

На сьогоднішній день їх налічується 140 штук. У міру розширення можливостей браузерів ця кількість зростає. Використання іменованих кольорів досить зручно, оскільки слова запам'ятовувати простіше, ніж числові значення.

Приклад - HTML

```
<h1>Відтінки зеленого</h1>
<div class="green">Green</div>
<div class="aquamarine">Aquamarine</div>
<div class="chartreuse">Chartreuse</div>
<div class="darkgreen">DarkGreen</div>
<div class="darkseagreen">DarkSeaGreen</div>
```

Приклад - CSS

```
body {
      background-color:honeydew;
      text-align: center;
}
h1 {
      color: forestgreen;
}
div {
```

```
        height: 50px;
        margin-bottom: 0.5em;
        color: honeydew;
        font-size: 2em;
}
.green {
        background-color: green;
}
.aquamarine {
        background-color: aquamarine;
}
.chartreuse {
        background-color: chartreuse;
}
.chartreuse {
        background-color: chartreuse;
}
.darkgreen {
        background-color: darkgreen;
}
.darkseagreen {
        background-color: darkseagreen;
}
```

Нижче продемонстровано навіть весь перелік іменованих відтінків зеленого кольору.

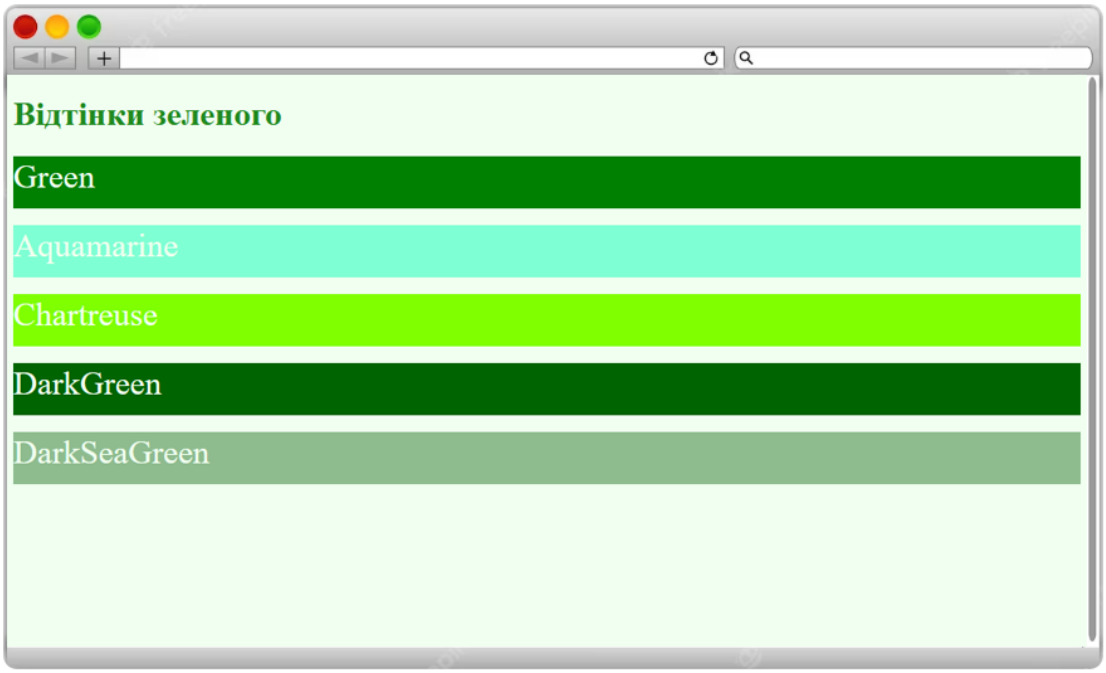

Рис.14. Іменовані відтінки зеленого кольору

Шістнадцяткові RGB-кольори

Не для кожного можливого відтінку є назва, тому можна задавати фарбування елементів у вигляді коду шістнадцяткового. HEX-значення позначаються решіткою, за якою слідує 6 цифр (або 3, у разі якщо дві цифри однієї тріади однакові, наприклад, #112233 можна скоротити до #123).

Приклад - HTML

```
<h1>Можна використовувати три 16-річні числа</h1>
<h2>А можна також і 6 чисел</h2>
```

Приклад - CSS

```
h1 {
      color: #a0a;
}
h2 {
      color: #123456;
}
```

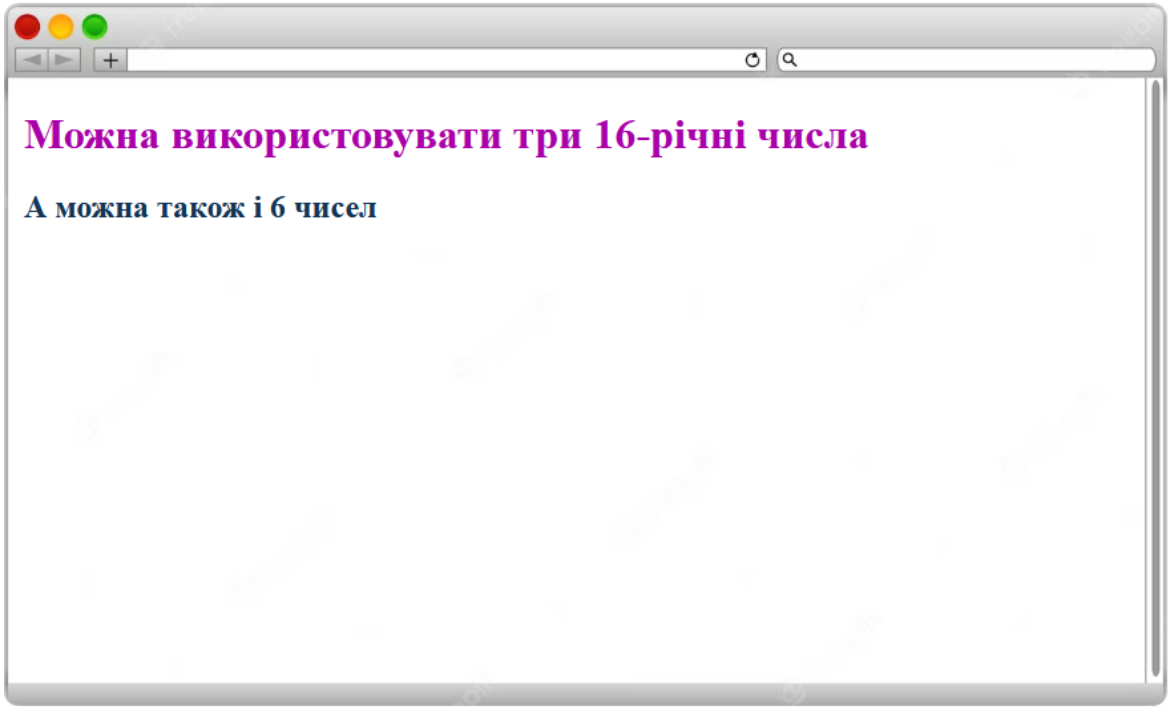

Рис.15. Використання HEX-значень для завдання кольору

Застосування функцій rgb() та rgba()

Дані функції приймають 3 цілих числа від 0 до 255, що задають діапазон відтінків червоного, зеленого та синього кольорів відповідно. Для rgba() останнім параметром можна призначити рівень прозорості від 0 до 1.

Приклад - HTML

```
<p class="p1">Непрозорий абзац</p>
<p class="p2">Прозорий абзац</p>
```

Приклад - CSS

```
body {
      background-color: rgb(255, 201, 53);
}
.p1 {
      background-color: rgb(118, 123, 202);
      height: 100px;
}
.p2 {
      background-color: rgba(224, 89, 89, 0.47);
      height: 10vh;
}
```

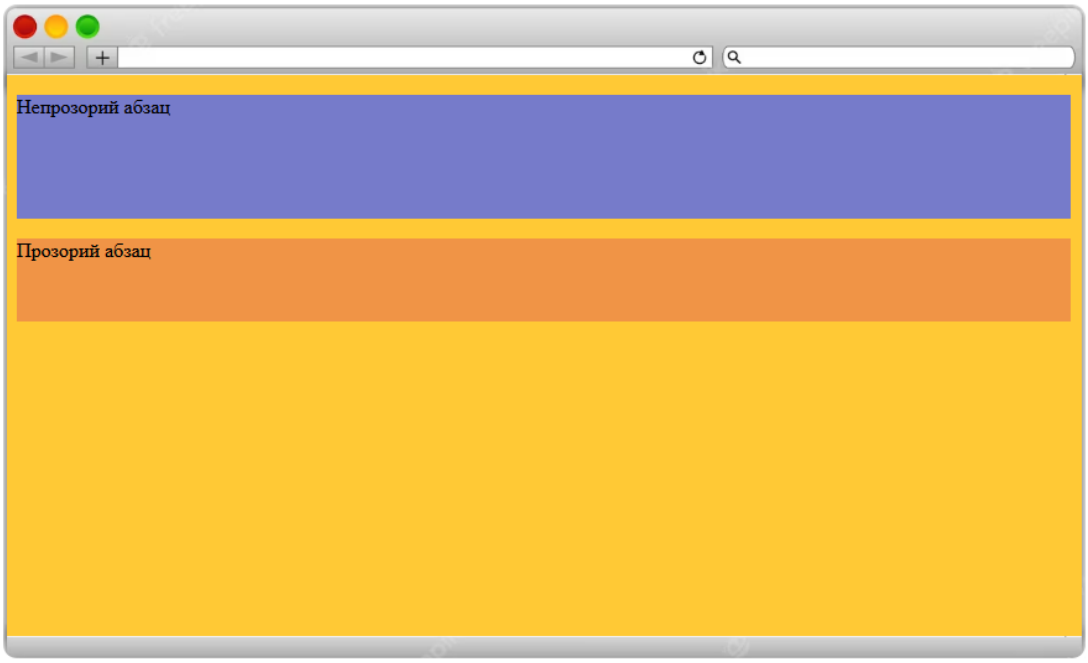

Рис.16. Використання функцій rgb() та rgba() для завдання кольору

Застосування функцій hsl() та hsla()

Відтінок (число від 0 до 360), насиченість та яскравість (від 0 до 100%) – порядок встановлення параметрів для даних функцій. У hsla() додатково виставляється прозорість.

Приклад - HTML

```
<p class="p1">Непрозорий абзац</p>
<p class="p2">Прозорий абзац</p>
```

Приклад - CSS

```
body {
      background-color: hsl(86, 70%, 81%);
}
.p1 {
      background-color: hsl(221, 37%, 49%);
      height: 100px;
```

```
}
.p2 {
      background-color: rgba(124, 29, 153, 0.38);
      height: 10vh;
}
```

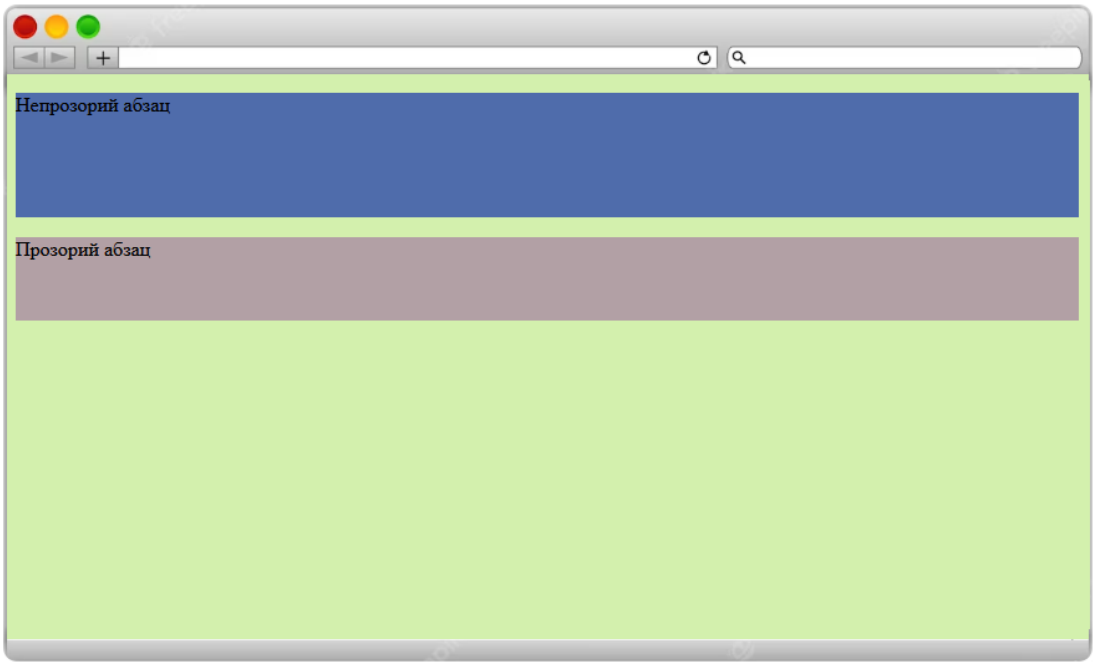

Рис.17. Використання функцій hsl() та hsla()для завдання кольору

Типи даних у CSS. Зображення (image).

Зображення – тип даних для представлення двовимірної графіки. Сюди включають об'єкти, що позначені:

- Функцією url().
- Градієнтами.
- Частинами сторінок, виділеними функцією element() - експериментальна властивість, яка поки що не підтримується більшістю браузерів.
- Зображеннями, певними функціями image() - поки не реалізовано.
- Змішування картинок за допомогою cross-fade() – працює не скрізь, потрібно вказувати якість з урахуванням браузера.
- Набором зображень image-set() – також не може скрізь працювати.

Для подальших кількох прикладів необхідно покласти файли з картинками в папку img. Для цього виберіть 3 довільні картинки, назвіть їх 1.png, 2.png та 3.png відповідно.

Приклад - HTML

```
<div class="bg-im"></div>
<div class="grad-im"></div>
<div class="mix-im"></div>
<div class="set-im"></div>
```

Приклад - CSS

```
div {
```

```
        height: 20vh;
        background-color: sandybrown;
        margin: 1vw;
}
.bg-im {
        background: url("../img/1.png") no-repeat center;
}
.grad-im {
        background: Linear-gradient(to left, red, blue);
}
.mix-im {
        background: cross-fade(url("../img/2.png"), url("../img/3.png"),
35%) no-repeat center;
        background: -webkit-cross-fade(url("../img/3.png"),
url("../img/1.png"), 35%) no-repeat center;
}
.set-im {
        background: url("../img/5.png") no-repeat center;
        background: -webkit-image-set("../img/3.jpg" 1dppx,
"../img/1.png" 2dppx) no-repeat center;
        background: -webkit-image-set("../img/2.png" 1dppx,
"../img/3.png" 2dppx) no-repeat center;
}
```

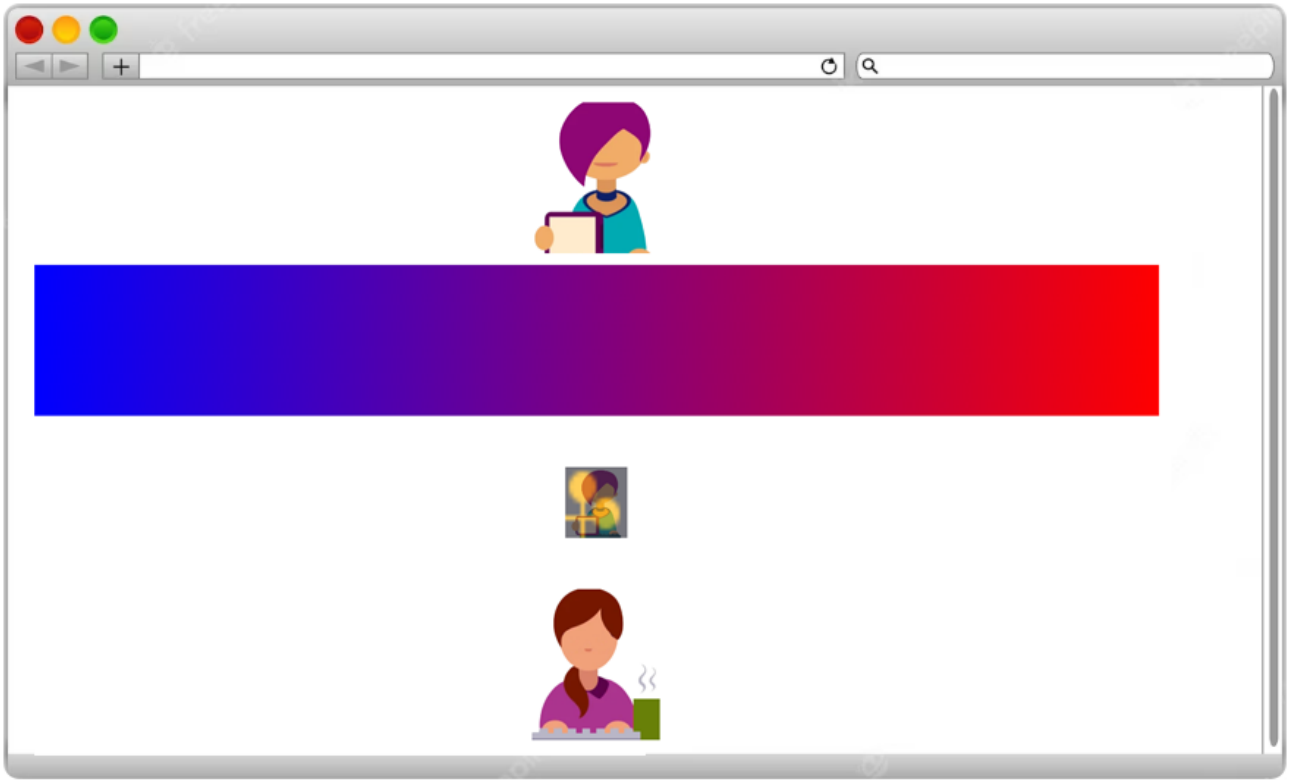

Рис.18. Різні способи представлення зображень у Chrome

Ці функції з прикладів не скрізь можуть працювати, поки що CSS не до кінця здатний замінити JavaScript у наведеному функціоналі.

Типи даних у CSS. Позиція.

Тип даних є актуальним для властивостей background-position і offset-anchor та відповідає за положення елемента щодо батьківського блоку (не обов'язково, що він перебуватиме всередині батька в результаті модифікацій).

Задається за допомогою ключових слів: top, bottom, left, right, center. Крім ключових слів, позиція може визначатися відсотками, абсолютними і відносними величинами або успадковуватися від об'єкта в DOM-дереві, що знаходиться вище.

Синтаксис наступний:

Синтаксис position: Одне значення

Завдання ключового слова, відсотка чи одиниці довжини.

Приклад - HTML

```
<div class="center"></div>
<div class="bottom"></div>
<div class="percent"></div>
```

Приклад - CSS

```
div {
      height: 30vh;
      background-color: sandybrown;
      margin: 1vw;
      background-size: 200px;
}
.center {
      background-image: url("../img/1.png");
      background-position: center;
      background-repeat: no-repeat;
}
.bottom {
      background-image: url("../img/1.png");
      background-position: bottom;
      background-repeat: no-repeat;
}
.percent {
      background-image: url("../img/1.png");
      background-position: 60%;
      background-repeat: no-repeat;
```

```
}
```

Перший блок вирівняв фонову картинку строго по центру вертикально та горизонтально. У другого блоку картинка перемістилася вниз по вертикалі, але відцентрована по осі X. У третьому блоці зображення зсунуто вліво на 60 % від ширини елемента, тоді як вертикальне центрування не порушилося.

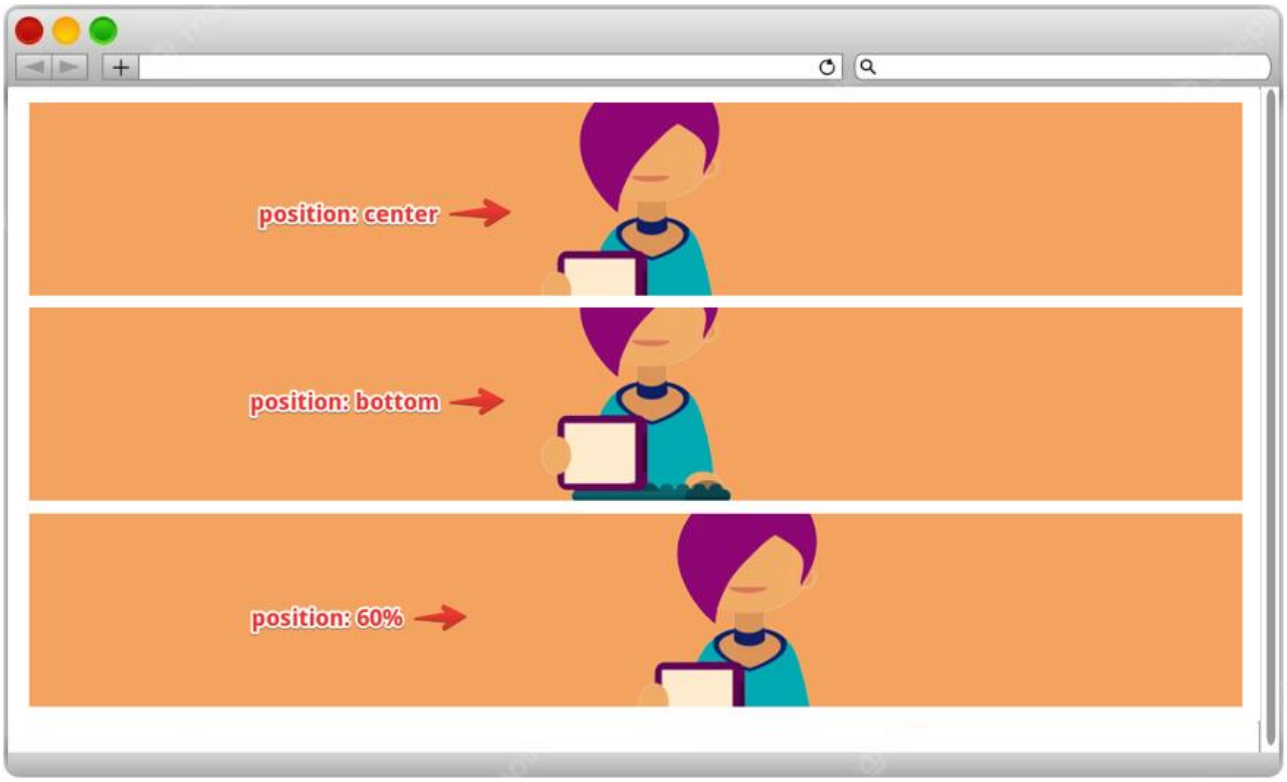

Рис.19. Вирівнювання картинки за допомогою властивості position

Синтаксис position: Два значення

У даному варіанті ключові слова left-right і top-bottom взаємовиключні, а друга величина означає зсув другої осі - зверху або зліва відповідно.

Приклад - HTML

```
<!-- Розміщення зображення внизу праворуч -->
<div class="bottom-right"></div>
<!-- Центрування зображення -->
<div class="center"></div>
<!-- Зсув на 10 пікселів вліво і 10% зверху -->
<div class="values"></div>
```

Приклад - CSS

```
div {
      height: 30vh;
      background-color: sandybrown;
      margin: 1vw;
      background-size: 200px;
}
```

```
.bottom-right {
      background-image: url("../img/1.png");
      background-position: bottom right;
      background-repeat: no-repeat;
}
.center {
      background-image: url("../img/1.png");
      background-position: 50% 50%;
      background-repeat: no-repeat;
}
.values {
      background-image: url("../img/1.png");
      background-position: 10px 10%;
      background-repeat: no-repeat;
}
```

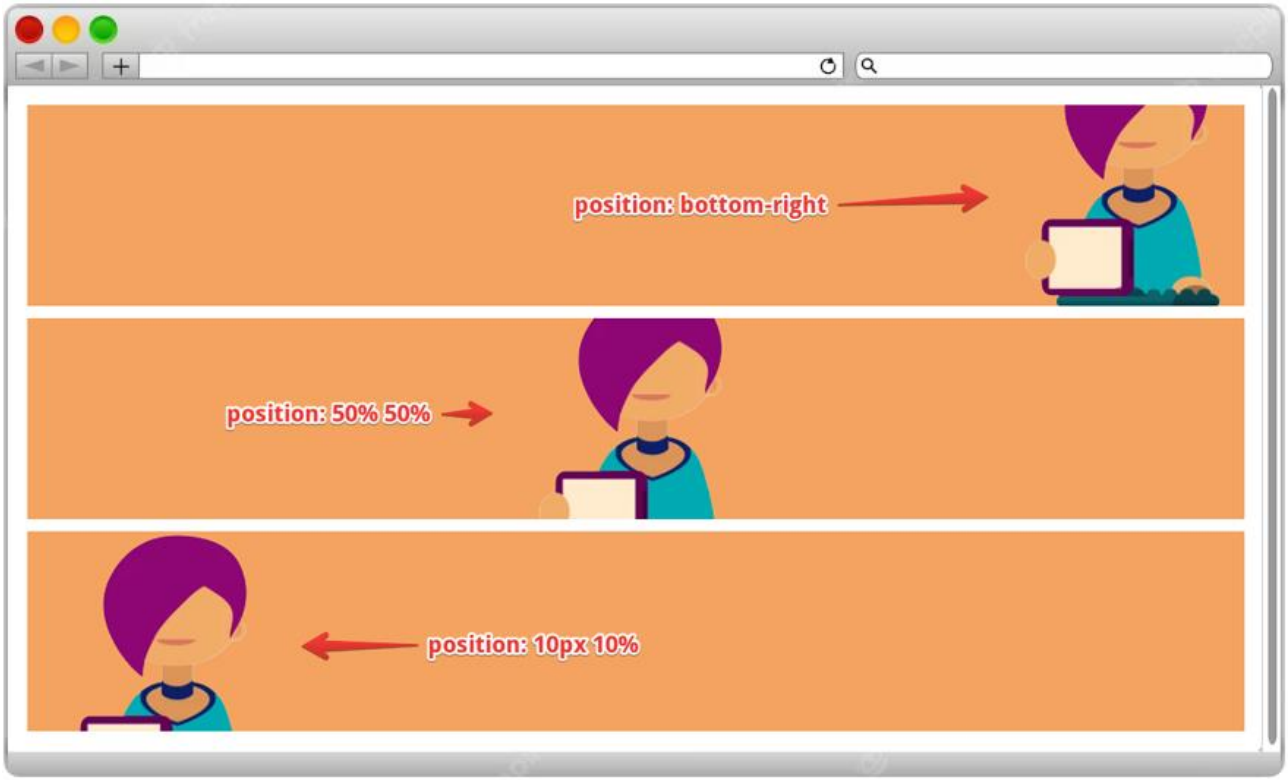

Рис.20. Вирівнювання картинки за допомогою властивості position

Синтаксис position: Три значення

Два ключові слова і зсув за останнім.

Приклад - HTML

```
<!--Розміщення зображення внизу справа з зсувом на 40 пікселів праворуч-->
<div class="bottom-right"></div>
<!-- Центрування картинки та розміщення її вниз з відступом від нижнього краю на 1em-->
```

```
<div class="center-bottom"></div>
```

Приклад - CSS

```
div {
      height: 30vh;
      background-color: sandybrown;
      margin: 1vw;
      background-size: 200px;
}
.bottom-right {
      background-image: url("../img/1.png");
      background-position: bottom right 40px;
      background-repeat: no-repeat;
}
.center-bottom {
      background-image: url("../img/1.png");
      background-position: center bottom 1em;
      background-repeat: no-repeat;
}
```

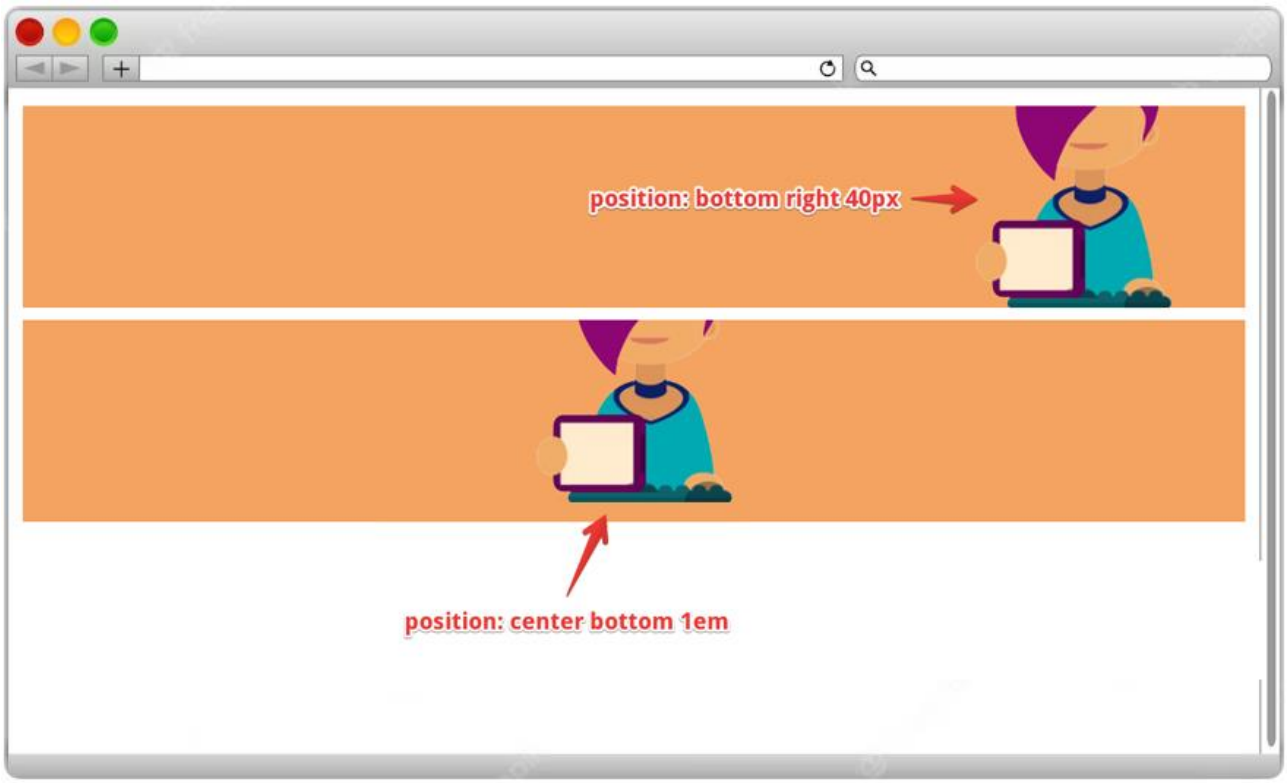

Рис.21. Вирівнювання картинки за допомогою властивості position

Синтаксис position: Чотири значення

Ключові слова та їх значення.

Приклад - HTML

```
<!--Розміщення зображення вниз з зсувом 10% і вправо з зсувом на 40px-->
```

```
<div class="bottom-right"></div>
<!-- Зображення знаходиться зліва вгорі і відсунуто на 25px-->
<div class="left-top"></div>
```

Приклад - CSS

```
div {
      height: 30vh;
      background-color: sandybrown;
      margin: 1vw;
      background-size: 200px;
}
.bottom-right {
      background-image: url("../img/1.png");
      background-position: bottom 10% right 40px;
      background-repeat: no-repeat;
}
.left-top {
      background-image: url("../img/1.png");
      background-position: top 25px left 25px;
      background-repeat: no-repeat;
}
```

Рис.22. Вирівнювання картинки за допомогою властивості position

Типи даних у CSS. Рядок.

До рядкових елементів можна застосувати наступні властивості: content (змінює вміст елемента згенерованим контентом), font-family (задає шрифти) та quotes (визначає поведінку цитат).

Приклад - HTML

```
<h1>Застосовуємо шрифти</h1>
<p>Тут використано інший шрифт</p>
```

Приклад - CSS

```
p {
     font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
}
```

Для зазначення шрифту, що складається більше ніж з двох слів, потрібно його назву укладати у лапки.

Типи даних у CSS. Опції.

Сучасний CSS постійно запроваджує новий функціонал. Крім розглянутих вище (для завдання кольору) є інші:

- calc() - дозволяє обчислювати значення властивостей.
- min() – визначає мінімальну величину.
- max() - повертає максимум.
- clamp() – працює з діапазоном меж і на підставі бажаного значення задає властивість.
- attr() – отримує значення атрибута, яке можна використовувати в таблицях стилів.

Приклад - HTML

```
<h1 data-before="Змінюємо " data-after="!">атрибути</h1>
```

Приклад - CSS

```
h1::before {
     content: attr(data-before);
}
h1::after {
     content: attr(data-after);
}
```

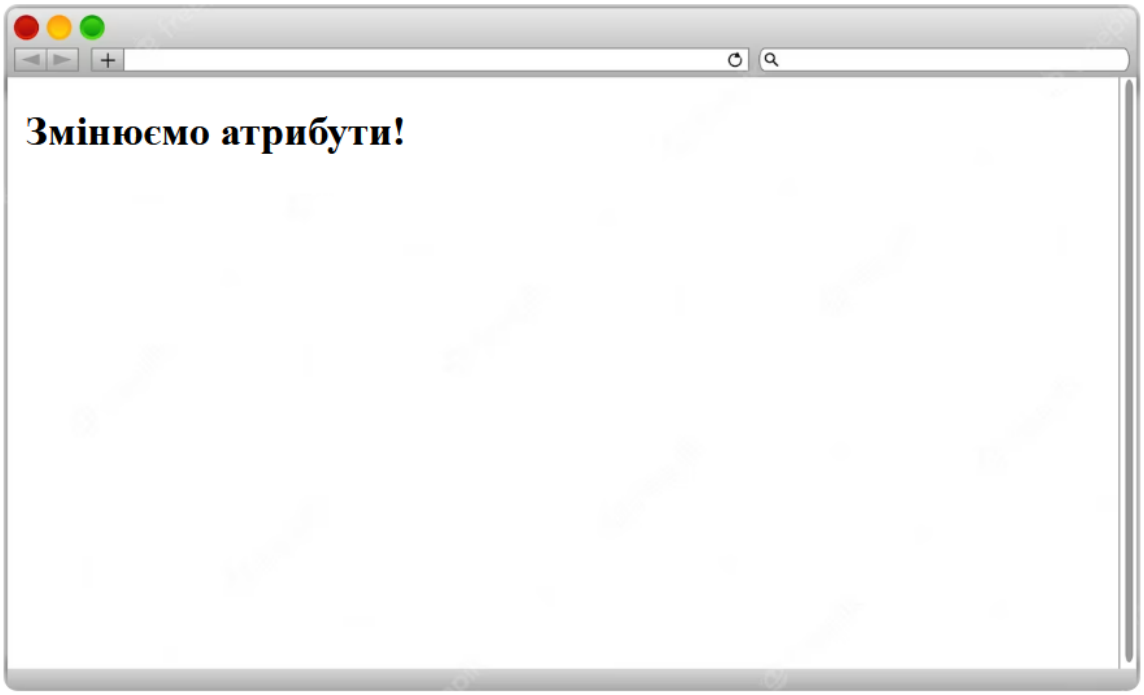

Рис.23. Формування вмісту елемента за допомогою функції attr()

У цьому прикладі створено власні властивості, значення яких використані обчислено з HTML-коду. Тепер заголовок виглядає так: Змінюємо атрибути!

Підходи до завдання розмірів елементів у HTML-документах

Розміри у css задаються чи визначаються кількома варіантами.

Природний розмір

Багато HTML-елементів мають так званий природний розмір: вони займають стільки місця на сторінці, скільки становить обсяг їхнього контенту. Зокрема, теги і не мають початкових параметрів висоти, тут висота набувається за наповнення контентом, додавання меж, відступів тощо.

Приклад - HTML

```
<article></article>
<article></article>
<article></article>
```

Приклад - CSS

```
article {
     border: seagreen 4px solid;
}
```

Оскільки контент статей порожній, то вони не мають жодних розмірів. Після того, як задати їм межі, вони збільшились на суму цих значень.

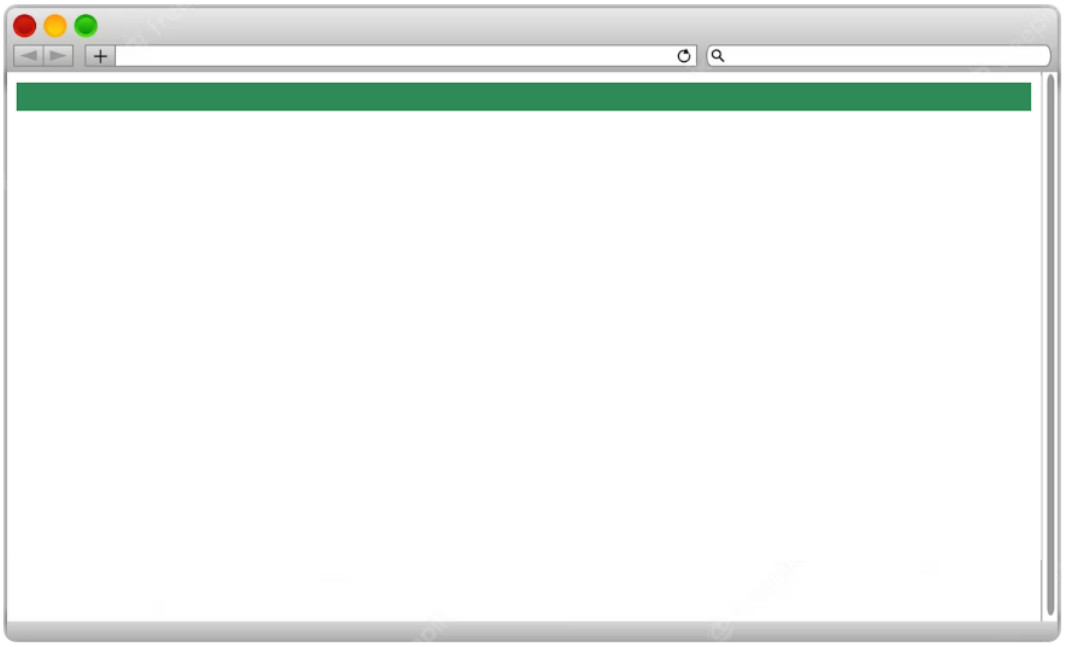

Рис.24. Три елементи з порожнім контентом, але заданими межами у 4px

Заданий розмір

Найзручніше самим визначати ширину та висоту елементів. Цим гарантується, що сторінка виглядатиме так, як заплановано. Але якщо привласнити малі значення цим властивостям, то контент може в них не поміститися і почне заходити за межі блоків.

Приклад - HTML

```
<article>Програмуємо красиво</article>
<article>Вивчаємо властивості CSS</article>
<article>Заглиблюємося в стилі</article>
```

Приклад - CSS

```
article {
      border: seagreen 2px dotted;
      height: 100px;
      width: 10vw;
}
```

Тепер заплановані статті стануть ідентичними за розмірами, що візуально приємніше.

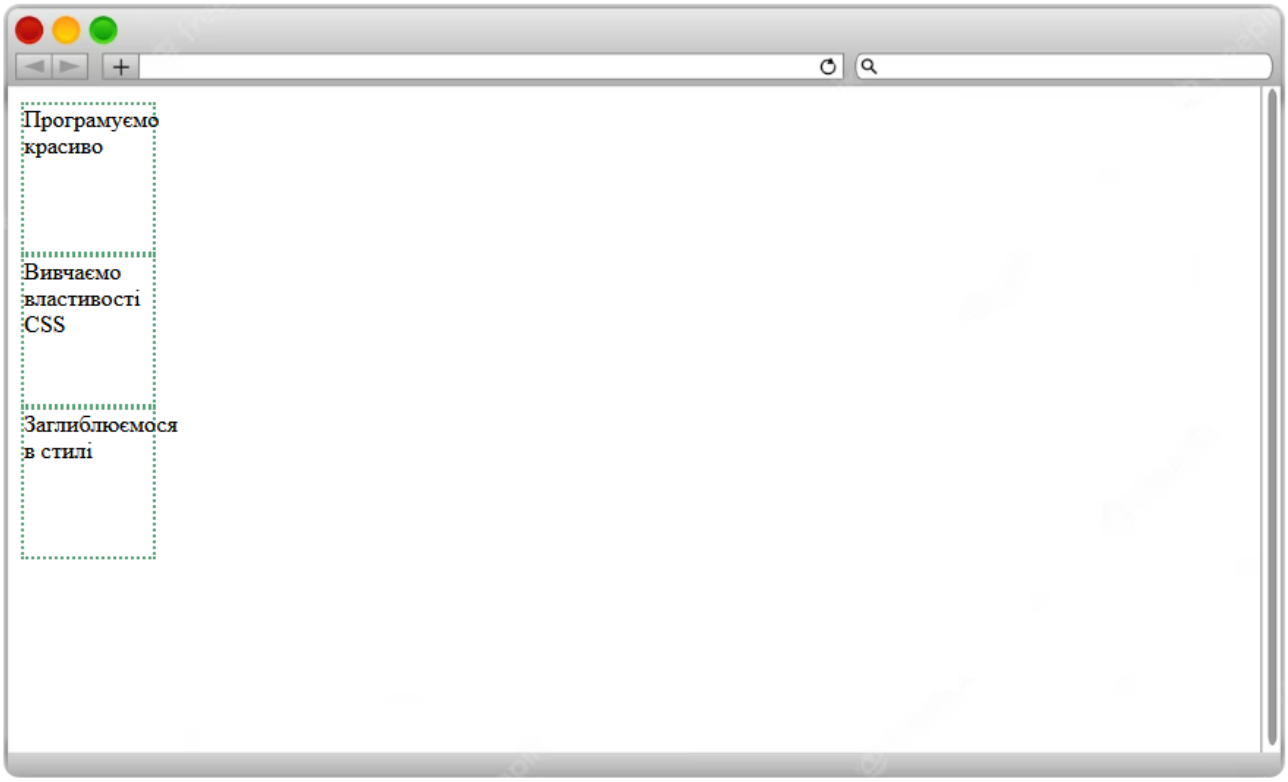

Рис.25. Три елементи із заданими розмірами: height=100px та width=10vw

Розміри у відсотках

Форматувати блоки можна у пікселях, а й у відсоткових значеннях. Призначення відсоткових параметрів потребує наявності батьківського елемента. Якщо у батька немає розміру (порожній блок), то відсотки не дадуть жодних результатів.

Приклад - HTML

```
<article>
      <h1>Перша стаття</h1>
</article>
<article>
      <h1>Друга стаття</h1>
</article>
<article>
      <h1>Третя стаття</h1>
</article>
```

Приклад - CSS

```
article {
      border: rgb(67, 0, 99) 2px solid;
      height: 100px;
      width: 90vw;
}
h1 {
      width: 75%;
```

```
        height: 90%;
        padding: 5%;
        text-align: center;
}
```

Якщо тегу <article> не задати висоту, то відсоткові розміри для заголовків не матимуть жодної дії.

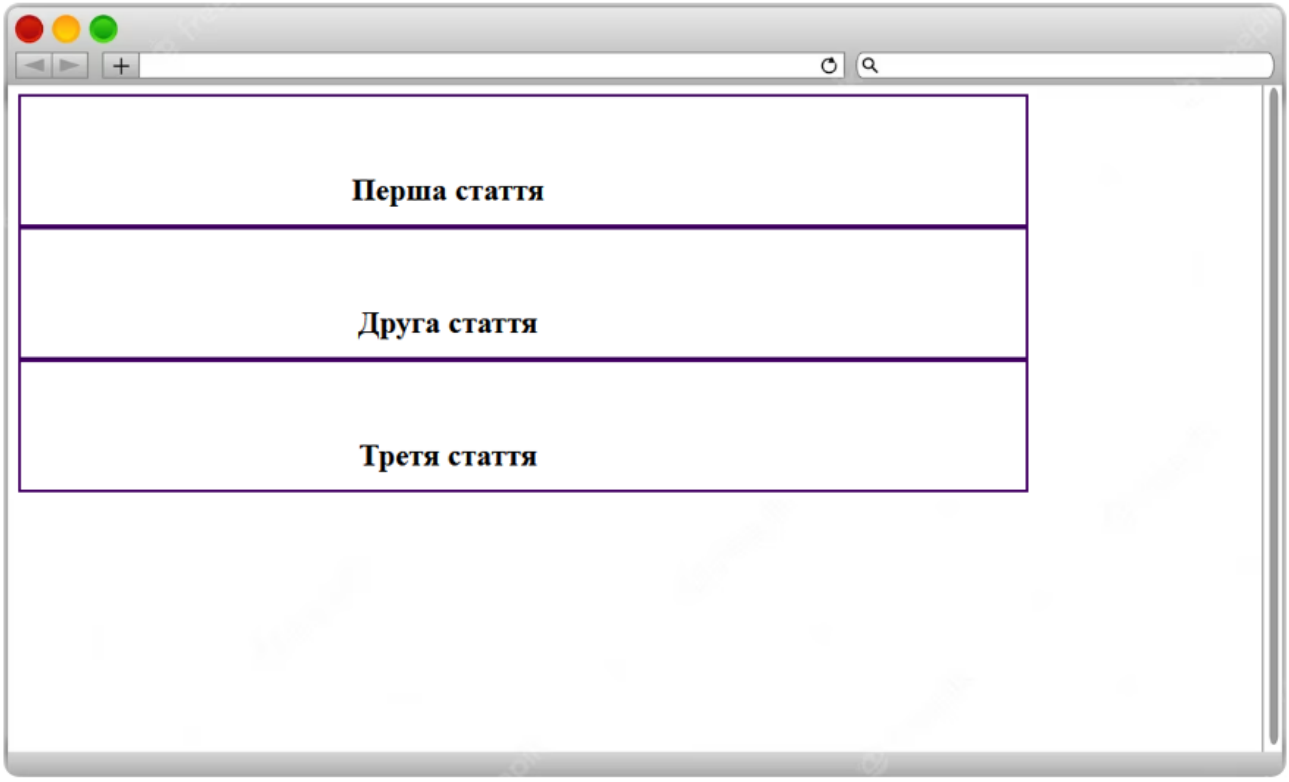

Рис.26. Три елементи з розмірами px і vw і заголовками у відсотках

Мінімальні та максимальні розміри

Жорстко заданий формат елементів – не завжди добре. У деяких користувачів сторінка виглядатиме надто маленькою (багато порожнього простору по боках), а іншим доведеться користуватися горизонтальною прокруткою, що наразі є недобре. З іншого боку, якщо параметри не задати, то сторінку можна зменшувати до нескінченності.

Є проміжне рішення, основна мета якого: обмежити розміри блоків для естетичності та зручності. Для цього використовують min та max при визначенні ширини та висоти об'єктів.

Приклад - HTML

```
<div>Сайт у розробці</div>
<div>Повідомимо, коли буде готово</div>
```

Приклад - CSS

```
div {
        min-height: 100px;
        background-color: skyblue;
        margin: 3%;
        max-width: 500px;
```

```
    padding: 2em;
    text-align: right;
}
```

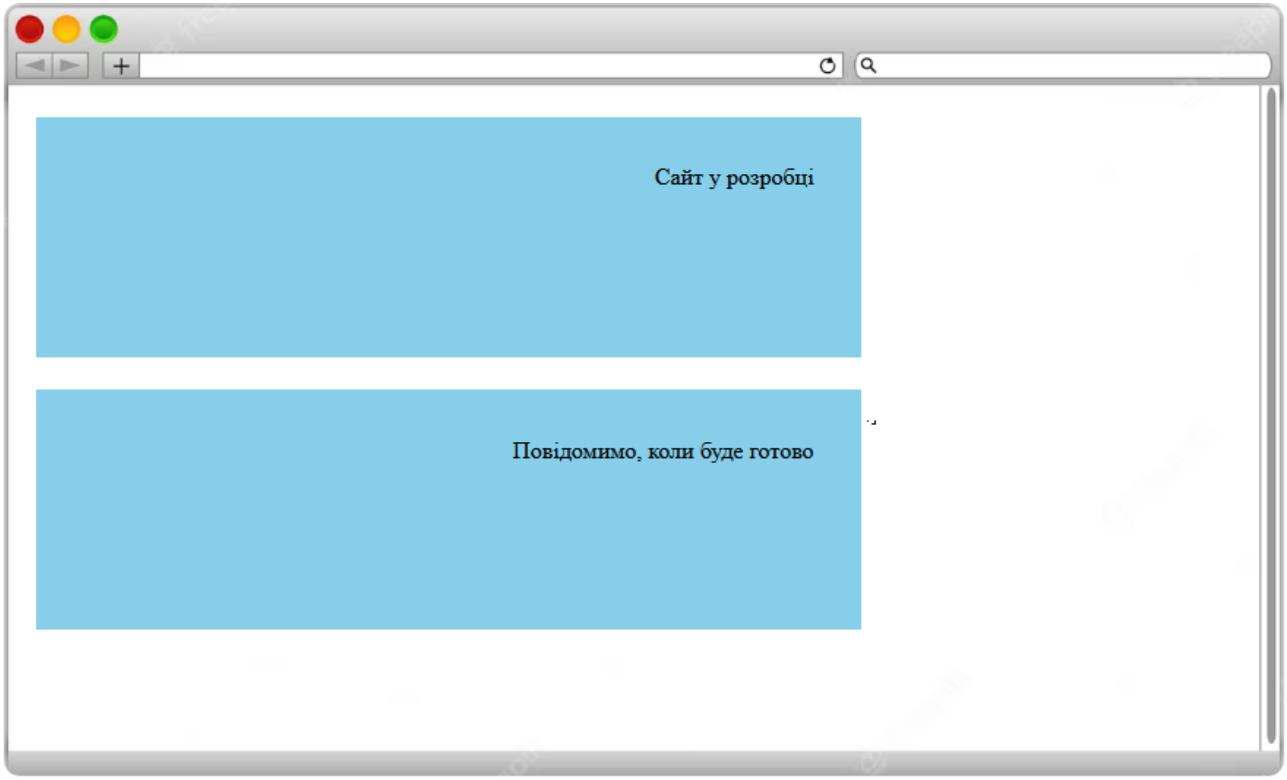

Рис.27. Елементи з обмеженим максимальним значенням ширини 500px

Розміри в залежності від величини вікна браузера

Багато розробників сьогодні користуються зручними одиницями довжини: vw і vh. Вони спрощують процес створення сайтів, які виглядають максимально схожими на різні пристрої. Масштаби елементів не змінюватимуться у смартфоні, планшеті, дисплеї.

Приклад - HTML

```
<p>Зручно читати і на моніторі, і на невеликому екрані телефону</p>
```

Приклад - CSS

```
p {
    height: 70vh;
    width: 95vw;
    font-size: 4vmax;
}
```

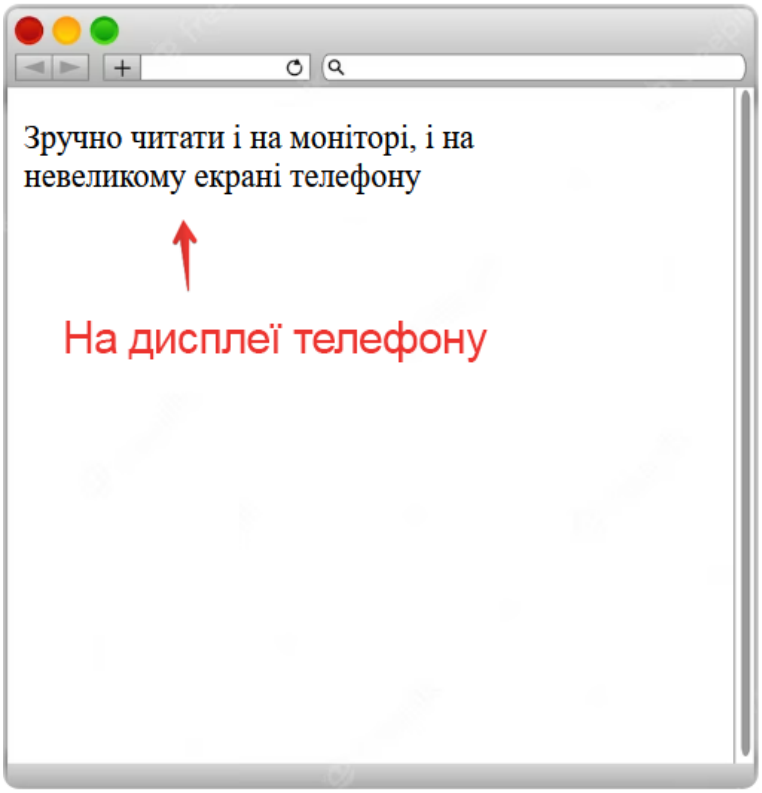

Рис.28. Відображення тексту на маленькому екрані

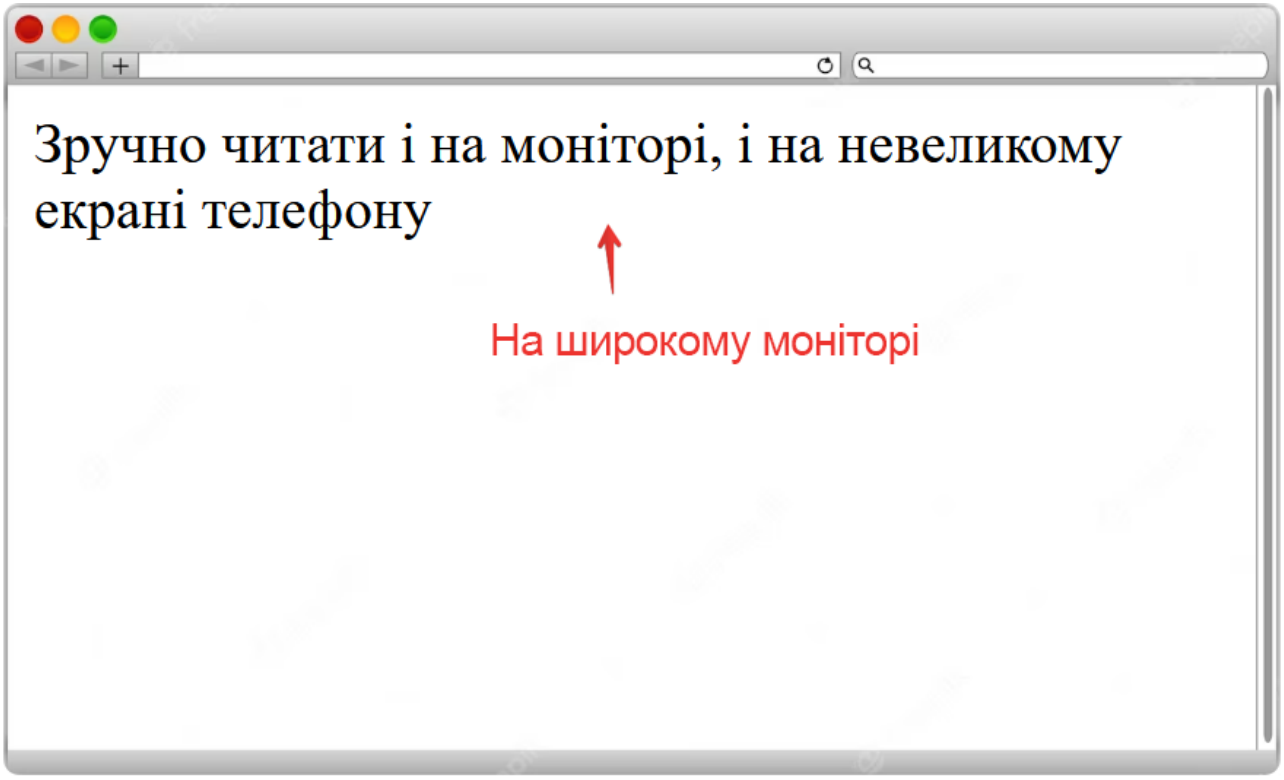

Рис.29. Відображення тексту на екрані з більшою шириною – розмір шрифту збільшився

HTML-елементи: загальні властивості

Специфіка взаємодії з HTML-елементами через каскадні таблиці стилів залежить від типу елемента та його властивостей. Ознайомимося з трьома базовими категоріями стилів, які застосовуються до більшості об'єктів.

Фон

Будь-який блоковий елемент можна наділити фоном із певними параметрами. CSS властивість background – загальноприйняте скорочення налаштування цілого ряду параметрів. Спочатку подивимося весь його функціонал без скороченої форми :

Властивість	Опис
background-color	Служить для завдання кольору фону з допустимого набору значень (можуть використовуватися як вбудовані імена, так і через функції або hex-числа).
background-image	Задає як фон певне зображення або градієнт. Картинка заповнює весь доступний їй простір, за необхідності повторюючись, якщо розмір блоку більший за неї.
background-repeat	Керує повторенням зображення в елементі (по горизонталі, вертикалі, в обох напрямках, не повторюється).
background-size	Допомагає змінювати розмір фонового зображення
background-position	Позиціонування фону в блоці (по центру, по боках).
background-attachment	Контроль за поведінкою зображення при гортанні елемента (рухатиметься разом з контентом або залишатиметься на місці).

Всі ці властивості можна одночасно зазначити в атрибуті background.

Приклад - HTML

```
<div></div>
<div></div>
```

Приклад - CSS

```
div {
      width: 80vw;
      height: 30vh;
      background-color: lightcoral;
      background-image: url("../img/2.png");
      background-repeat: no-repeat;
      background-size: 20%;
```

```
        background-position: bottom right;
}
div + div {
        background: rgb(255, 189, 142) url("../img/1.png") repeat-x top/22%;
}
```

У першому випадку використано повну форму властивостей фону, а у другому – скорочену.

Рис.30. Два елементи з різними параметрами фону

Межі

Визначення меж в елементах можна проводити як відразу для 4-х сторін, так і для кожної окремо. Основні властивості:

Властивість	Опис
`border-width`	Ширина рамки (у пікселях, відсотках тощо). Може записуватись через скорочений атрибут border.
`border-style`	Вид рамки (суцільна, у вигляді крапок, хвилясті лінії тощо). Може записуватись через скорочений атрибут border.
`border-color`	Колір рамки. Може записуватись через скорочений атрибут border.

Властивість	Опис
`border-radius`	Заокруглення кутів.

Приклад - HTML

```
<div></div>
```

Приклад - CSS

```
div {
      width: 30vw;
      height: 30vw;
      background-color: lightcoral;
      border: purple 5px groove;
      border-radius: 50%;
}
```

В результаті операцій над елементом та завдання йому заокруглених меж максимальної величини квадратний перетворився у коло.

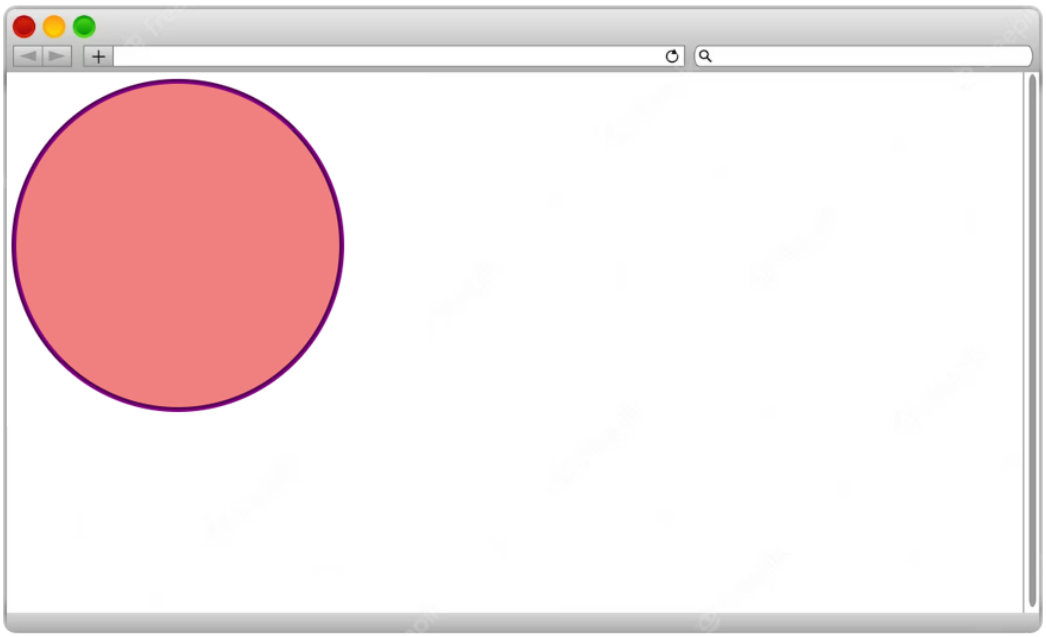

Рис.31. Круг із квадратного блоку за допомогою округлення кордонів

Перекриття вмісту (overflow)

Нерідко трапляються ситуації, коли всередині контейнера не міститься його вміст. В результаті збивається верстка, блоки накладаються один на одного. Цією поведінкою можна керувати. Для цього призначена властивість overflow. Її можливі значення:

Значення	Опис
`visible`	Контент повністю відображається (за замовченням).
`hidden`	Вміст обрізається і стає невидимим.
`scroll`	Інформація хоч і обрізається, але її можна подивитися за допомогою смуги прокручування
`auto`	Використання налаштувань браузера користувача.
`inherit`	Наслідувати властивість від батька.

Приклад - HTML

```
<p class="visible">Багато різних текстів, наданих сайтом для студентів та інших користувачів</p>
<p class="hidden">Багато різних текстів, наданих сайтом для студентів та інших користувачів</p>
<p class="scroll">Багато різних текстів, наданих сайтом для студентів та інших користувачів</p>
```

Приклад - CSS

```
p {
      width: 130px;
      height: 70px;
      border: indigo 8px solid;
      padding: 5px;
      margin-bottom: 40px;
}
.visible {
      overflow: visible;
}
.hidden {
      overflow: hidden;
}
.scroll {
      overflow: scroll;
}
```

Всі три параграфи поводяться по-різному: у першому текст виходить за його межі, у другому – обрізається, а у третьому – обрізається з прокруткою.

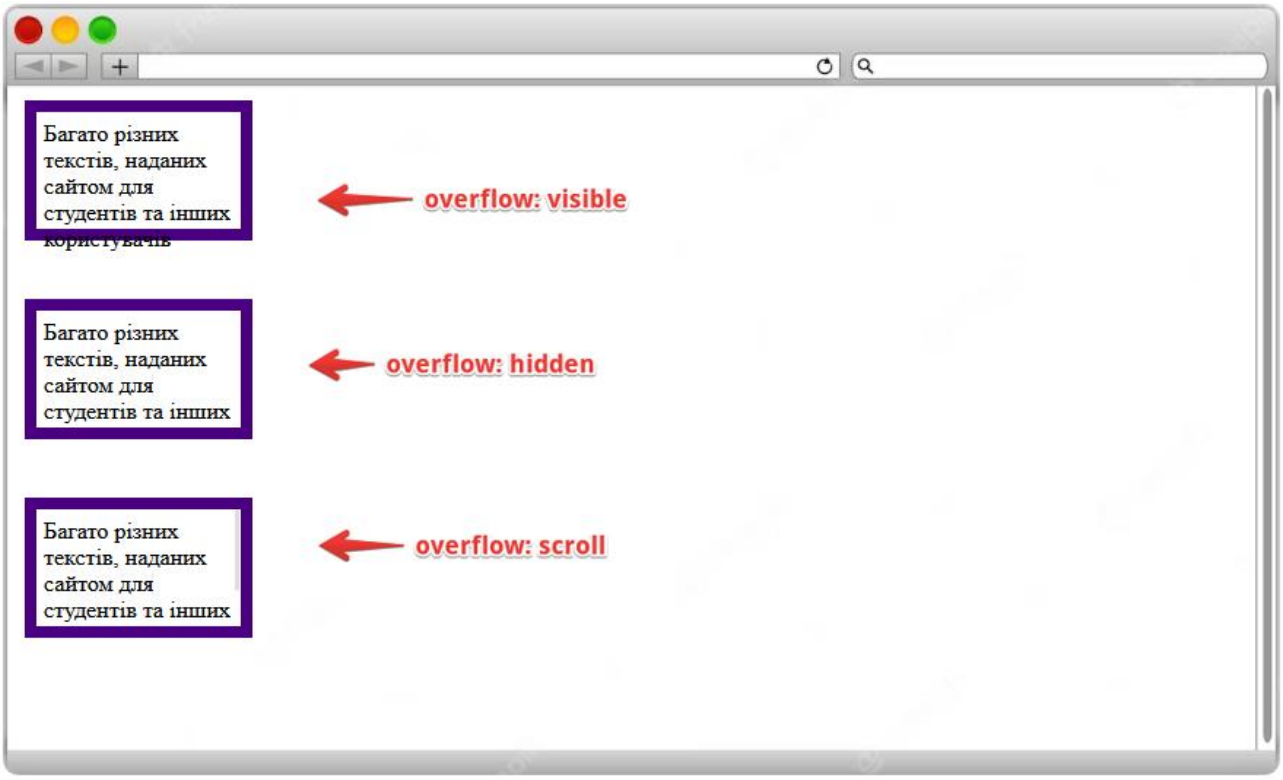

Рис.32. Три елементи з різним перекриттям

Запитання

1. Якщо блоковому елементу як фон задати синій колір з прозорістю 0, то якого кольору він буде в результаті?

Ця операція повністю нівелює колір блокового елемента. Він набуде кольору свого батька, якщо він призначений, а в іншому випадку "скористається" налаштуваннями браузера.

Незважаючи на дивовижність призначення абсолютної прозорості об'єкту, вона може використовуватися в динаміці, анімаціях, коли у відповідь на дії користувача колір буде змінюватися, як і прозорість.

2. Як розмістити фонову картинку строго по центру та зверху поточного елемента?

CSS

```
background-position: center top;
background-repeat: no-repeat;
```

Важливо не забути відключити повторення картинки по горизонталі (яке за замовчуванням активовано), щоб не отримати розмножену фотографію, коли вона невеликого розміру.

3. Чим відрізняються одиниці виміру em і rem ?

При визначенні величини шрифту, наприклад, використання rem відштовхується від кореневого елемента web-сторінки (яким є <html>), а em значення як базу беруть батька.

За замовчуванням rem дорівнюють 16 пікселям. Ніхто не заважає змінити цю величину. У більшості випадків (при роботі зі складними сайтами) використання rem зручніше, тому що множинна вкладеність здатна зробити шрифт дуже маленьким або більшим.

Завдання

Завдання 1

Для більш красивого відображення написів на сторінці Іван вирішив запровадити свій власний шрифт під назвою Mega Font Cool. Допоможіть дизайнеру за допомогою каскадних таблиць стилів прив'язати його до сайту. Не забудьте про те, що не всі відвідувачі мають такий шрифт у себе на комп'ютері. До речі, шрифт відноситься до сімейства моноширинних.

Завдання 2

Створіть блок квадратної форми розміром 600 на 600 пікселів. Всередину нього помістіть такий самий блок, але вдвічі менше. Продовжуйте операцію доти, доки не вийде якась матрьошка, що складається з 7 вкладених елементів.

Кожен об'єкт усередині свого батька повинен розташовуватися строго у центрі. Намагайтеся мінімізувати кількість коду і вдатися лише до вивченого матеріалу.

Джерела інформації

1. CSS. Типи даних та робота з HTML-елементами
   https://smartiqa.ru/courses/web/lesson-5-css