МАРЧЕНКО Наталя Андріївна,
кандидат технічних наук,
доцент кафедри системного аналізу та
інформаційно-аналітичних технологій
Національного технічного університету
«Харківський політехнічний інститут»

МАЛЬКО Максим Миколайович,
кандидат технічних наук,
професор кафедри системного аналізу та
інформаційно-аналітичних технологій
Національного технічного університету
«Харківський політехнічний інститут»

СИДОРЕНКО Ганна Юріївна,
кандидат технічних наук,
доцент кафедри системного аналізу та
інформаційно-аналітичних технологій
Національного технічного університету
«Харківський політехнічний інститут»

У навчально-методичному посібнику викладено опис базових понять CSS, механізмів підключення таблиць стилів до html-документів, опис синтаксису та правил форматування різних типів html-елементів та css-селекторів, принципи позиціювання елементів у потоці HTML. Навчально-методичний посібник призначено для студентів спеціальностей 124 «Системний аналіз» та 186 «Видавництво і поліграфія».

Н. А. Марченко
М. М. Малько
Г. Ю. Сидоренко

ТЕХНОЛОГІЯ CSS

НАВЧАЛЬНО-МЕТОДИЧНИЙ ПОСІБНИК

Ужгород, 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет
«Харківський політехнічний інститут»

Н. А. Марченко, М. М. Малько, Г.Ю. Сидоренко

ТЕХНОЛОГІЯ CSS

Навчально-методичний посібник
для студентів спеціальностей 124 «Системний аналіз»,
186 «Видавництво і поліграфія»

Затверджено редакційно-
видавничою
радою університету,
протокол № 3 від 26.10.2022 р.

Ужгород
ТОВ «РІК-У»
2023

УДК 004.55(075)
М 21

Р е ц е н з е н т и :
М. А. Гринченко, канд. техн. наук, доцент,
завідувач кафедри «Стратегічне управління» НТУ «ХПІ»
М.В. Сидоров, докт. фіз.-мат. наук, професор,
завідувач кафедри «Прикладна математика» ХНУРЕ

Марченко Н. А.
М 21 Технологія CSS: навч.-метод. посіб. / Марченко Н. А., Малько М. М., Сидоренко Г. Ю. – Ужгород; Харків: ТОВ «РІК-У», НТУ «ХПІ», 2023 – 108 с.
ISBN 978-617-8046-96-5

Викладено опис базових понять CSS, механізмів підключення таблиць стилів до html-документів, опис синтаксису та правил форматування різних типів html-елементів та css-селекторів, принципи позиціювання елементів у потоці HTML.

Для студентів спеціальностей 124 «Системний аналіз», 186 «Видавництво і поліграфія» з дисциплін «Основ'и Internet-технологій».

Табл. 3. Іл. 29 Бібліогр. 4

УДК 004.55(075)

ISBN 978-617-8046-96-5

ВСТУП

Технологія CSS (Cascading Style Sheets – каскадні таблиці стилів) – одна з базових технологій у сучасному Інтернеті, де практично відсутні сайти, створені без її використання. Це пов'язано з тим, що згідно з рекомендаціями організації World Wide Web Consortium (W3C), що займається стандартизацією в Інтернет, для розробки навіть найпростішого вебдокументу треба застосовувати HTML (Hypertext Markup Language – мова розмітки гіпертексту) для створення його структури та смислового наповнення та CSS – для відображення його зовнішнього вигляду. Цим досягається один з фундаментальних принципів створення вебдокументів – відділення їх структури від подання.

CSS дозволяє представляти один і той же документ у різних стилях або методах виводу, таких як екранне уявлення, друковане подання, читання голосом (спеціальним голосовим браузером або програмою читання з екрана), або при виведенні пристроями, що використовують шрифт Брайля.

Отже, без знання CSS неможливо створити професійний дизайн сайту та впевнено почувати себе на ринку праці.

Навчально-методичний посібник містить опис базових понять CSS, механізмів підключення таблиць стилів до html-документів, опис синтаксису та правил форматування різних типів html-елементів та css-селекторів, принципи позиціювання елементів у потоці HTML. Теоретичний матеріал проілюстровано багатою кількістю прикладів. Завдання для лабораторних робіт наближені до практики та відображають повний цикл розробки статичного вебзастосунку.

1. ТЕОРЕТИЧНИЙ МАТЕРІАЛ

1.1. Історія виникнення та базові поняття CSS

У середині 90-х років XX століття почався бурний розвиток Веб. У той час дизайн сайтів створювався виключно можливостями HTML, що розширювалися у кожній його версії. При цьому html-код сайту ставав складнішим, і існувало багато проблем з відображенням сайту у різних браузерах.

Всі ці проблеми були помічені організацією W3C, і у 1995 р. консорціум почав публікацію робочої версії стандарту, що получив назву CSS. К 1996 р. він получив статус рекомендації, такої ж важливої, як і HTML.

З тих пір для створення подання вебсторінок призначена технологія каскадних таблиць стилів (Cascading Style Sheets, CSS). Таблиця стилів містить набір правил (*стилів*), що описують оформлення самої вебсторінки і окремих її фрагментів.

Головною особливістю сучасної версії CSS 3 є не тільки можливість візуального подання різних елементів, но і створення анімацій, підтримка лінійних і радіальних градієнтів, тіней, згладжування і багато іншого.

У CSS кожен стиль має бути прив'язаний до відповідного елементу вебсторінки або до самої вебсторінки. Після прив'язки параметри, що описані обраним стилем, починають застосовуватися до даного елементу. При цьому прив'язка може бути:

- *явна*, тобто вказується, який стиль якого елементу вебсторінки прив'язаний;
- *неявна* – стиль автоматично застосовується до всіх елементів вебсторінки, створених за допомогою, наприклад, певного html-тегу.

Серед переваг CSS можна виділити:

- *спритність* – CSS має кілька особливостей, що роблять код більш гнучким, легким у використанні, легким в управлінні, у змінах, найчастіше код CSS може бути змінений у режимі реального часу;
- *розділення форматування і вмісту* – головна особливість CSS, тобто правила форматування не будуть у цьому випадку самим змістом, вони будуть зберігатися окремо;

- *час завантаження, трафік* – CSS радикально покращує ці аспекти. Це пов'язано з тим, що інформація про форматування зберігається в окремому файлі конфігурації CSS, вона завантажиться один раз і не генерує небажаний трафік для кожного документа окремо, на відміну від сайту, створеного без використання CSS;
- *редизайн сторінки* – можливість переформатувати документ HTML. Якщо існує кілька сторінок HTML, їх можна швидко переформатувати, змінивши код CSS. Параметри будуть поширюватися на всі документи HTML, що використовують цю таблицю стилів.

1.2. Включення CSS у html-документи

Для включення таблиць стилів у html-документи існує декілька способів, що відрізняються між собою можливостями та своїм призначенням.

1.2.1. *Зв'язані стилі*

При використанні зв'язаних стилів їх опис розміщується в окремому текстовому файлі з розширенням `css`. Цей файл підключається до html-документа у розділі його заголовка, тобто розміщується всередині контейнерного тега `<head>`, за допомогою тега `<link>`:

```
<link type="text/css" rel="stylesheet" href="URL"
                media="пристрій">
```

Значення атрибутів тега `<link>` – `rel` і `type` залишаються незмінними, тобто відповідно `stylesheet` і `text/css`. Значення атрибуту `href` задає шлях до css-файлу, що може бути заданий як відносно, так і абсолютно.

Атрибут `media` дозволяє задати пристрій, для відображення на якому призначена відповідна таблиця стилів. Це дозволяє зробити різний стиль для відображення документа на екрані монітора і при його друці. При цьому допускається запис кількох значень через кому. Можливі значення атрибуту наступні:

- `all` – всі пристрої, значення за замовчуванням;
- `braille` – пристрої, засновані на системі Брайля, що призначені для людей з вадами зору;
- `handheld` – смартфони та пристрої з малою шириною екрану;
- `print` – друкувальні пристрої на зразок принтера;

- `screen` – екран монітора;
- `speech` – мовні синтезатори, а також програми для відтворення тексту вголос. Сюди ж входять мовні браузери;
- `projection` – проєктор;
- `tty` – телетайпи, термінали, портативні пристрої з обмеженими можливостями екрану. Для них не повинні використовуватися пікселі як одиниці виміру;
- `tv` – телевізор.

Також у HTML5 як значення можуть бути вказані *медіа-запити*, що дозволяє створювати адаптивний дизайн сайту.

Підключена таким чином таблиця стилів називається *зовнішньою* (*External Style Sheets*), а стилі також називаються *зовнішніми*.

1.2.2. *Глобальні стилі*

При використанні *глобальних* або *внутрішніх стилів* (*Internal Style Sheets*) властивості CSS описуються у самому html-документі і розташовуються безпосередньо у його заголовку за допомогою контейнерного тега `<style>`. Наприклад,

```
<style type="text/css">
h1 {font-size: 12pt; color: red}
</style>
```

Тобто всі заголовки `<h1>` у html-документі повинні бути виведено шрифтом розміром `12pt` червоного кольору.

Створена таким чином таблиця стилів називається *внутрішньою*. Перевага внутрішньої таблиці стилів полягає у тому, що вона є невід'ємною частиною html-документу. Але існує і два недоліка при використанні внутрішніх стилів. По-перше, внутрішні стилі застосовуються тільки до того html-документа, в якому ця таблиця стилів знаходиться. По-друге, внутрішня таблиця стилів не відповідає рекомендаціям W3C, що вимагають відокремлення змісту документа від його подання.

В одному і тому ж html-документі можуть бути присутніми відразу декілька таблиць стилів: кілька зовнішніх і внутрішня. У такому випадку дія всіх цих таблиць стилів додається.

1.2.3. *Вбудовані стилі*

Вбудований стиль (*Inline Styling*) є розширенням для одиночного тегу і використовується у поточному html-документі. Для

визначення стилю використовується універсальний атрибут `style`, значенням якого виступає набір стильових правил. Наприклад,

```
<p style="font-size:14pt; color:blue">Приклад
вбудованого стиля </p>
```

Вбудовані стилі рекомендується застосовувати на сайті обмежено або взагалі відмовитися від їх використання, оскільки у цьому випадку також порушується розділення між змістом документа та його поданням. Крім того додавання таких стилів збільшує загальний обсяг файлів, що веде до підвищення часу їх завантаження у браузері, і ускладнює редагування документів для розробників.

1.2.4. *Імпорт стилів*

У поточну стильову таблицю можна імпортувати вміст css-файлу за допомогою команди `@import`. Цей метод допускається використовувати спільно зі зв'язаними або глобальними стилями. Загальний синтаксис наступний.

```
@import url("ім'я файлу") типи носіїв;
@import "ім'я файлу" типи носіїв;
```

Після ключового слова `@import` вказується шлях до стильового файлу одним з двох наведених способів – за допомогою функції `url` або без неї. Типи носіїв співпадають зі вказаними при описі тегу `<link>`. Наприклад,

```
<style type="text/css">
@import: url("mystyle.css")
a {color: red}
</style>
```

1.2.5. *Правила каскадності та пріоритет стилів*

Для одного і того ж елемента вебсторінки можуть існувати різні описи у різних таблицях стилів. Тому постає питання об'єднання інформації, якщо у різних джерелах описані різні стильові властивості, та її пріоритету при однакових стильових властивостях.

Каскадність означає можливість об'єднання інформації про стиль з декількох джерел в одне узгоджене джерело.

Каскад стилів визначає впорядковану послідовність таблиць стилів. При цьому існують наступні *правила пріоритету стилів*:

- зовнішня таблиця стилів, посилання на яку (тег `<link>`) зустрічається в html-коді сторінки пізніше, має пріоритет перед тією, посилання на яку зустрілося раніше;
- внутрішня таблиця стилів має пріоритет перед зовнішніми;
- вбудовані стилі мають пріоритет перед будь-якими стилями, заданими у таблицях стилів;
- більш конкретні стилі мають пріоритет перед менш конкретними. Це означає, наприклад, що стильовий клас має пріоритет перед стилем перевизначення тегу, оскільки стильовий клас прив'язується до конкретних тегів. Аналогічно комбінований стиль має пріоритет перед стильовим класом;
- якщо до тегу прив'язані кілька стильових класів, то ті, що вказані правіше, мають пріоритет перед зазначеними лівіше.

Наприклад, якщо у зовнішній таблиці сказано, що текст абзаців повинен бути виведені шрифтом чорного кольору, а у внутрішній вказано розмір шрифту `12pt`, то у результаті текст абзаців документу буде у розміром `12pt` і чорного кольору. Якщо ж для одного з абзаців задано вбудований стиль, що задає розмір `14pt` і червоний колір, то він буде відформатований згідно з цих правил.

1.3. Особливості синтаксису CSS

Будь-яке визначення стилю включає *селектор* і *список властивостей стилю* з їх *значеннями*.

Основним поняттям є *селектор* – деяке ім'я стилю, для якого додаються параметри форматування. Загальний синтаксис селектора наступний:

```
селектор {властивість1: значення [;
властивість2: значення [...]]}
```

Тут і далі у прикладах частини, що можуть бути відсутніми, позначені квадратними дужками.

Серед особливостей можна виділити наступне:

- CSS не чутливий до регістру запису символів, переведення рядків і кількості пробільних символів;
- стильові властивості поділяються між собою крапкою з комою, у кінці цей символ можна опустити;

- всередині селекторів і імен стилів не повинні бути присутніми символи пробілів та переведення рядка;
- не допускається ніяких непробільний символів після закритої дужки;
- допускається використовувати коментарі, синтаксис яких наступний.

```
/* Текст коментаря */
```

1.4. Прості селектори CSS

До простих селекторів CSS належать селектори тегів, класи та ідентифікатори.

1.4.1. *Селектори тегів*

Як селектор може виступати будь-який тег HTML, для якого визначаються правила форматування, наприклад колір, фон, розмір тощо. Правила задаються у наступному вигляді.

```
тег {властивість1: значення; властивість2: значення;
                          ...}
```

Наприклад,

```
p {
text-align: justify; /* вирівнювання за шириною */
color: green; /* зелений колір тексту */
}
```

Хоча стиль можна застосувати до будь-якого тегу, результат форматування буде помітний лише для тегів, які відображаються у розділі тіла html-документа.

1.4.2. *Класи*

Класи застосовують, коли необхідно визначити стиль для індивідуального елемента вебсторінки або задати різні стилі для одного тега. При використанні спільно з тегами синтаксис для класів буде наступний.

```
тег.ім'я_класу {властивість 1: значення;
властивість2: значення; ...}
```

В іменах класів повинні використовуватися латинські символи, а також можуть містити у собі символ дефіса (-) і підкреслення (_).

Можна також використовувати класи і без вказівки тега.

```
.ім'я_класу {властивість 1: значення;
властивість2: значення; ...}
```

Щоб вказати в html-коді, що тег використовується з певним класом, до тегу додається універсальний атрибут `class`, значенням якого виступає ім'я класу, задане у таблиці стилів.

Нижче наведено приклад html-документа, для форматування якого використовуються класи, а його зображення у браузері – на рис. 1.1.

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS класи</title>
    <style type="text/css">
        p.head {text-align: center }
        .warn {color:red; font-weight: bold }
    </style>
</head>
<body>
<p class="head">Абзац заголовка документу</p>
<p class="warn">Попередження:</p>
<p>Звичайний абзац тексту …<span
class="warn">!!!</span></p>
</body>
</html>
```

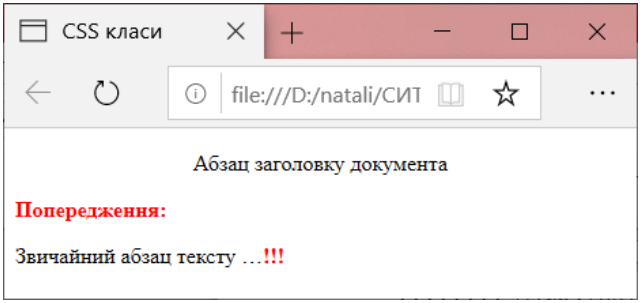

Рис. 1.1. Приклад CSS класів

Мультикласи дозволяють додати до будь-якого тега кілька класів, перераховуючи у значенні атрибута `class` їх назви через пробіл. У цьому випадку до елементу застосовується стиль, описаний у правилах для

кожного класу. Оскільки при додаванні декількох класів вони можуть містити однакові стильові властивості, але з різними значеннями, то береться значення у класі, який описаний у коді нижче.

Нижче наведено приклад html-документа з мультикласом, а його зображення у браузері – на рис. 1.2.

```
<!DOCTYPE html>
<html>
<head>
<title>Мультикласи</title>
<style type="text/css">
.layer1 { color: red; font-size:16pt}
.layer2 { color: blue; }
</style>
</head>
<body>
<p class="layer1">Текст красного кольору розміром
16pt</p>
<p class="layer2">Текст синього кольору</p>
<p class="layer1 layer2">Текст синього кольору
розміром 16pt</p>
</body>
</html>
```

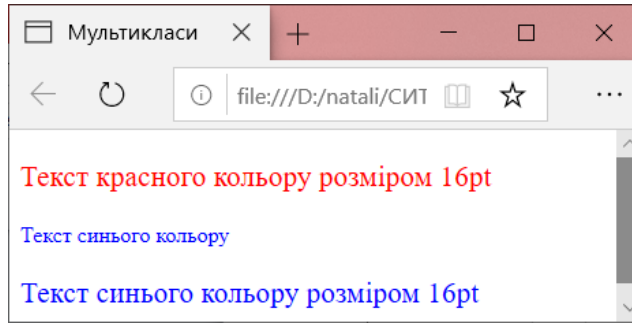

Рис. 1.2. Приклад мультикласів

1.4.1. *Ідентифікатори*

Ідентифікатор або *ID-селектор* визначає унікальне ім'я елемента, яке використовується для зміни його стилю і звернення до нього через скрипти. Синтаксис при використанні ідентифікатора наступний.

```
#ім'я_ідентифікатора {властивість 1: значення;
```

```
властивість2: значення; ...}
```

На відміну від класів ідентифікатори повинні бути унікальні, іншими словами зустрічатися у коді документа лише один раз.

Звернення до ідентифікатора відбувається аналогічно як і до класу, але як атрибут тега використовується `id`, значенням якого виступає ім'я ідентифікатора.

Нижче наведено приклад html-документа з ідентифікатором, а його зображення у браузері – на рис. 1.3.

```
<!DOCTYPE html>
<html>
 <head>
  <title>Ідентифікатори</title>
  <style type="text/css">
     p {font-size:16pt}
     #id1 {color: blue; font-size:12pt}
  </style>
 </head>
<body>
 <p>Текст чорного кольору розміром 16pt</p>
 <p id="id1">Текст синього кольору розміром 12pt</p>
</body>
</html>
```

1.5. Задання розмірів у CSS

Для задавання розмірів різних елементів у CSS використовуються *абсолютні* і *відносні одиниці виміру*. Абсолютні одиниці не залежать від пристрою виведення, а відносні одиниці визначають розмір елемента щодо значення іншого розміру.

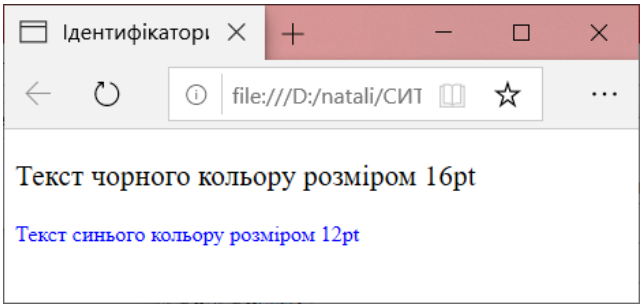

Рис. 1.3. Приклад ідентифікаторів

Відносні одиниці зазвичай використовують для роботи з текстом, або коли треба обчислити процентне співвідношення між елементами. У табл. 1.1 подані основні відносні одиниці.

Таблиця 1.1 – Відносні одиниці CSS

Одиниця	Опис
em	Розмір шрифта поточного елемента
ex	Висота символу x
px	Піксель
%	Процент

Одиниця `em` – це змінне значення, яке залежить від розміру шрифту поточного елементу. У кожному браузері закладений розмір тексту, який застосовується за замовчуванням. Тому спочатку `1em` дорівнює розміру шрифту, заданого у браузері за замовчуванням або розміру шрифту батьківського елементу. Відсотковий запис ідентичний `em`, тобто `1em = 100%`.

Одиниця `ex` визначається як висота символу «x» у нижньому регістрі. На `ex` поширюються ті ж правила, що і для `em`, а саме, він прив'язаний до розміру шрифту, заданого у браузері за замовчуванням, або до розміру шрифту батьківського елемента.

Піксель це елементарна точка, яка відображається монітором або іншим подібним пристроєм, наприклад, смартфоном. Розмір пікселя залежить від роздільної здатності пристрою і його технічних характеристик.

Абсолютні одиниці застосовуються рідше, ніж відносні, і зазвичай при роботі з текстом. У табл. 1.2 наведено основні абсолютні одиниці.

Таблиця 1.2 – Абсолютні одиниці CSS

Одиниця	Опис
cm	Сантиметр
mm	Міліметр
in	Дюйм, `1 in = 2,54 cm`
pt	Пункт, `1 pt = 1/72 in`
pc	Піка, `1 pc = 12 pt`

Найпоширенішою одиницею є *пункт*, який використовується для задання розміру шрифту не тільки в Інтернет-технологіях, а й у текстових редакторах і видавничих системах.

1.6. Задання кольору у CSS

У CSS кольори можна задавати різними способами: у шіснадцятковому коді, за назвою, у форматі RGB, RGBA, HSL, HSLA.

У *шіснадцятковому коді* кольори відповідають адитивній колірній моделі RGB. Колір у моделі RGB представлено сумою яскравостей трьох базових кольорів – червоного (Red), зеленого (Green) і синього (Blue). Назва моделі утворена з перших букв англійських назв цих кольорів.

Яскравість (інтенсивність) кожного базового кольору може приймати 256 (2^8) дискретних значень від 0 до 255. Змішування кольорів у різних пропорціях, варіюючи яскравість кожної складової, надає 256×256×256 = 16 777 216 кольорів.

У CSS, як і в HTML, для кожної колірної компоненти задається шістнадцяткове значення у межах від 00 до FF, що відповідає діапазону 0–255 у десятковому зчисленні. Потім ці значення об'єднуються в одне число, перед яким ставиться символ #. Наприклад, #800080 – темно-фіолетовий колір.

Допускається використовувати скорочену форму виду `#rgb`, де кожен символ слід подвоювати. Так, запис `#fe0` відповідає `#ffee00`.

Браузери підтримують звернення до деяких кольорів за їх *назвою*. У табл. 1.3 подано назви, шістнадцятковий код, значення у форматі RGB, HSL і опис.

Таблиця 1.3 – Стандартні кольори

Назва кольору	Шістнад-цяткове значення	RGB	HLS	Колір
black	#000000	rgb(0,0,0)	hsl(0,0%,0%)	чорний
silver	#C0C0C0	rgb(192,192,192)	hsl(0,0%,75%)	ясно-сірий
maroon	#800000	rgb(128,0,0)	hsl(0,100%,25%)	темно-червоний
red	#FF0000	rgb(255,0,0)	hsl(0,100%,50%)	червоний
green	#008000	rgb(0,128,0)	hsl(120,100%,25%)	зелений
lime	#00FF00	rgb(0,255,0)	hsl(120,100%,50%)	ясно-зелений
olive	#808000	rgb(128,128,0)	hsl(60,100%,25%)	оливковий
yellow	#FFFF00	rgb(255,255,0)	hsl(60,100%,50%)	жовтий
orange	#FFA500	rgb(255,165,0)	hsl(38.8,100%,50%)	жовтогарячий
navy	#000080	rgb(0,0,128)	hsl(240,100%,25%)	темно-синій
blue	#0000FF	rgb(0,0,255)	hsl(240,100%,50%)	синій
purple	#800080	rgb(128,0,128)	hsl(300,100%,25%)	темно-фіолетовий
fuchsia	#FF00FF	rgb(255,0,255)	hsl(300,100%,50%)	ясно-фіолетовий
teal	#008080	rgb(0,128,128)	hsl(180,100%,25%)	синьо-зелений
aqua	#00FFFF	rgb(0,255,255)	hsl(180,100%,50%)	морської хвилі
gray	#808080	rgb(128,128,128)	hsl(0,0%,50%)	сірий
white	#FFFFFF	rgb(255,255,255)	hsl(0,0%,100%)	білий

При заданні кольору у *форматі RGB* за допомогою функції `rgb` використовують значення червоної, зеленої та синьої компоненти у десятковому зчисленні. Кожна з трьох компонент кольору приймає значення від 0 до 255. Також допустимо задавати колір у процентному відношенні, при цьому 100% буде відповідати числу 255. Наприклад, rgb(255, 128, 128) або rgb(100%, 50%, 50%).

Формат RGBA схожий за синтаксисом до RGB, але включає у себе ще один параметр – *альфа-канал*, що задає прозорість елемента. Значення 0 відповідає повній прозорості, 1 – непрозорості, а деяке

проміжне значення, наприклад 0.5 – напівпрозорості. Наприклад, rgba(255, 128, 128, 0.5).

Назва *формату HSL* утворено від сполучення перших букв Hue (відтінок), Saturate (насиченість) і Lightness (яскравість). *Відтінок* – це значення кольору на колірному колі, що задається у градусах. 0° відповідає червоному кольору, 120° – зеленому, а 240° – синьому. Це основні три кольори які, змішуючись, утворюють ще три додаткових 60° – жовтий, 180° – блакитний і 300° – фіолетовий. Між цими шістьма основними і додатковими кольорами розташовані всі інші відтінки колірного спектру, тобто значення відтінку може змінюватися від 0 до 359.

Друге значення (Saturation) колірної моделі HSL визначає *насиченість* обраного відтінку і вказується у процентах у діапазоні від 0% до 100%. Значення 0% позначає відсутність кольору і відтінок сірого, 100% максимальне значення насиченості.

Яскравість вказується у відсотках від 0% до 100%. Малі значення роблять колір темнішим, а високі – світлішим, крайні значення 0% і 100% відповідають чорному і білому кольору відповідно. Приклади використання формату HSL подано у табл. 1.3.

Формат HSLA подібний за синтаксисом до HSL, але включає у себе альфа-канал, що задає прозорість елемента. Як і у випадку формату RGBA, значення 0 відповідає повній прозорості, 1 – непрозорості, а деяке проміжне значення – напівпрозорості.

1.7. Параметри шрифту

Використання шрифтів є одним із головних засобів форматування html-документів. Розглянемо стильові властивості, що задають параметри шрифту.

Властивість `font-family` задає ім’я шрифту, яким буде виведений текст:

```
font-family: <список імен шрифтів через кому>|
inherit
```

Імена шрифтів задаються у вигляді їх назв, наприклад, Arial або Times New Roman. Якщо ім’я шрифту містить пробіли, його потрібно взяти у лапки:

```
P {font-family: Arial}
H1 (font-family: "Times New Roman"}
```

Якщо даний атрибут стилю присутній у вбудованому стилі, лапки замінюють апострофами:

```
<p style="font-family: 'Times New Roman'">
```

Якщо зазначений шрифт присутній на комп'ютері користувача, браузер його використовує. Якщо ж такого шрифту немає, то текст виводиться шрифтом, заданим у настройках за замовчуванням.

Можна вказати кілька найменувань шрифтів через кому:

```
P {font-family: Verdana, Arial}
```

Тоді браузер спочатку буде шукати перший із зазначених шрифтів, у разі невдалого пошуку – другий, потім третій і тощо.

Замість імені конкретного шрифту як значення можна задати ім'я одного з *сімейств шрифтів*:

- *serif* – пропорційні шрифти, тобто які мають різну ширину для різних символів, із зарубками. Типовим представником цього сімейства є гарнітура Times;
- *sans-serif* – пропорційні шрифти без зарубок, наприклад Arial;
- *monospace* – моноширинні, тобто непропорційні шрифти, наприклад Courier;
- *cursive* – шрифти, що імітують рукописний текст;
- *fantasy* – декоративні шрифти.

Значення `inherit` вказує, що текст даного елемента вебсторінки повинен бути набраний тим же шрифтом, що і текст батьківського елемента. Тобто у даному випадку елемент вебсторінки «успадковує» шрифт від батьківського елементу. Для властивості `font-family` це значення за замовчуванням.

Зауваження. Значення `inherit` зустрічається у більшості CSS властивостей. Воно говорить браузеру, що елемент вебсторінки, до якого прив'язаний стиль, «успадковує» значення відповідного параметра у батьківського елемента. Для більшості атрибутів це значення за замовчуванням. Надалі дане значення для кожної властивості стилю описано не буде; якщо ж якийсь властивість стилю не підтримує його або існує інше значення за замовчуванням, про це буде згадано спеціально.

Властивість `font-size` визначає розмір шрифту, який може бути задано в одиницях виміру CSS або за допомогою одного з символьних значень:

```
font-size: <розмір>|xx-small|x-small|small|
medium|large|x-large|xx-large|larger|smaller|
inherit
```

Символьні значення від `xx-small` до `xx-large` задають сім наперед заданих розмірів шрифту, від найменшого до найбільшого. Значення `larger` і `smaller` представляють наступний розмір шрифту, відповідно за зростанням і за зменшенням. Наприклад, якщо для батьківського елемента визначено шрифт розміру `medium`, то значення `larger` встановить для поточного елемента розмір шрифту `large`.

Властивість `font-style` задає накреслення шрифту:

```
font-style: normal|italic|oblique|inherit
```

Доступні значення, що представляють звичайний шрифт (`normal`), курсив (`italic`), особливе декоративне накреслення, подібне до курсиву (`oblique`), а також успадкувати від батьківського елемента (`inherit`).

Коли для тексту встановлено курсивне або декоративне накреслення, браузер звертається до системи для пошуку відповідного шрифту. Якщо заданий шрифт не знайдений, браузер використовує спеціальний алгоритм для імітації потрібного виду тексту. Хоча курсив і декоративний шрифти подібні, вони мають відмінності. Курсив – це спеціальний шрифт, що імітує рукописний, декоративний шрифт утворюється шляхом нахилу звичайних знаків вправо.

Значенням за замовчуванням для властивості `font-style` є `normal`.

Властивість `font-style` задає насиченість («жирність») шрифту:

```
font-weight: bold|bolder|lighter|normal|
100|200|300|400|500|600|700|800|900
```

Для даної властивості доступні сім абсолютних значень від 100 до 900, що представляють відносну насиченість шрифту, від мінімальної до максимальної; при цьому звичайний шрифт буде мати насиченість 400 (або `normal`), а напівжирний – 700 (або `bold`). Значення за замовчуванням є 400 (`normal`). Значення `bolder` і `lighter` є відносними і представляють наступні ступені насиченості відповідно у більшу і меншу сторону.

На практиці насиченість у браузерах зазвичай обмежена всього двома варіантами: нормальне накреслення і напівжирне накреслення.

Властивість `font-variant` дозволяє створити *капітель*, тобто вивести текст за допомогою шрифту з малих прописних літер:

```
font-variant: normal|small-caps|inherit
```

Значенням за замовчуванням, як і у попередніх випадках, є `normal`, тобто регістр символів без зміни.

Приклад використання шрифтів для форматування тексту наведено нижче, а його відображення у браузері – на рис. 1.4.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Шрифти</title>
  <style>
   body {font-family: "Times New Roman"}
   h1  {font-size: 18pt}
   p {font-size: 14pt}
   .cap {font-variant: small-caps}
   .ob  {font-style:oblique}
   .it  {font-style:italic}
  </style>
 </head>
 <body>
 <h1>Звичайний заголовок</h1>
 <h1 class=cap>Заголовок-капітель</h1>
 <p>Звичайний текст</p>
 <p class=ob>Текст з декоративним накресленням</p>
 <p class=it>Текст з курсивним накресленням</p>
 </body>
</html>
```

Існує узагальнена властивість `font`, що дозволяє задати одночасно декілька значень з різних властивостей. Наприклад,

```
p {font: 12pt sans-serif}
```

В якості обов'язкових значень властивості `font` вказується розмір шрифту і його сімейство. Решта значень, такі як `font-style`, `font-variant`, `font-weight`, задаються за бажанням.

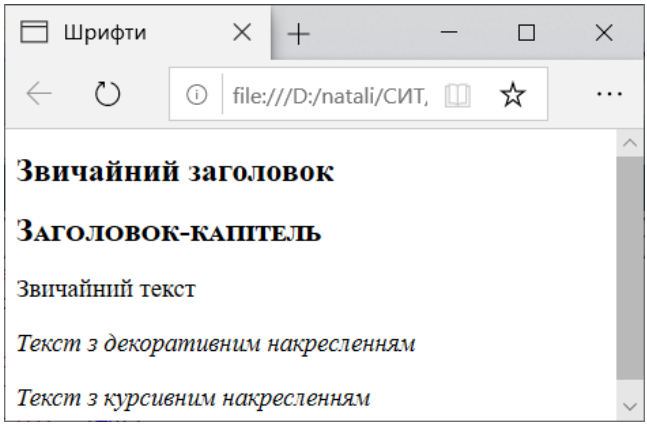

Рис. 1.4. Приклад використання шрифтів

1.8. Параметри тексту

Крім параметрів шрифту, у CSS 3 існує багато інших властивостей, що відповідають за відображенням тексту. Змінюючи властивості тексту, можна впливати на його положення, розміщати текст нижче або вище, керувати його вирівнюванням тощо. Розглянемо ці властивості більш докладно.

Властивість `color` задає колір тексту елемента за правилами задання кольорів. Значення за замовчуванням залежить від налаштувань браузера, зазвичай це чорний колір. Синтаксис властивості такий:

```
color: колір|inherit.
```

Властивість `text-align` задає горизонтальне вирівнювання тексту у межах елемента:

```
text-align: center|justify|left|right|start|end
```

Можливі значення:

- `center` – вирівнювання тексту по центру. Подібний спосіб вирівнювання часто використовується у заголовках і для створення підписів;
- `justify` – вирівнювання за шириною, що означає одночасне вирівнювання по лівому і правому краю. Щоб це зробити браузер додає пробіли між словами;

- `left` – вирівнювання тексту по лівому краю, це значення за замовченням;
- `right` – вирівнювання тексту по правому краю. Зазвичай застосовується для коротких заголовків обсягом не більше трьох рядків;
- `start` – аналогічно значенням `left` при напрямку тексту зліва направо, і `right`, коли текст йде навпаки справа наліво;
- `end` – значення, що протилежне попередньому, тобто аналогічно значенням `right`, якщо текст йде зліва направо і `left`, коли текст йде справа наліво.

Властивість `text-decoration` додає оформлення тексту у вигляді його підкреслення, перекреслення або лінії над текстом. Одночасно можна застосувати більш одного стилю, вказуючи значення через пробіл. Синтаксис властивості такий:

```
text-decoration: line-through||overline||underline
|none|inherit
```

Можливі значення:

- `line-through` – створює перекреслений текст;
- `overline` – лінія проходить над текстом;
- `underline` – встановлює підкреслений текст;
- `none` – скасовує всі ефекти, у тому числі і підкреслення у гіперпосилань, яке задано за замовчуванням. Це значення за замовчуванням.

Властивість `text-indent` встановлює величину відступу першого рядка блоку тексту в одиницях виміру CSS. Допускаються як додатні, так від'ємні значення. Значенням за замовчуванням є `0`, тобто відступ відсутній. Синтаксис властивості такий:

```
text-indent: <значення> |inherit.
```

Властивість `text-overflow` визначає параметри видимості тексту у блоці, якщо текст цілком не поміщається у задану область. Можливі два варіанти: текст обрізається (`clip`); текст обрізається і до кінця рядка додається три крапки (`ellipsis`). Дана властивість працює у тому випадку, якщо для блоку значення властивості `overflow` встановлено як `auto`, `scroll` або `hidden`. Значенням за замовчуванням є `clip`. Синтаксис:

```
text-overflow: clip | ellipsis
```

Властивість text-shadow додає ефект тіні до тексту, а також встановлює її параметри: колір тіні, зміщення щодо напису і радіус розмиття:

```
text-shadow: none | тінь [, тінь],
```

де параметр тінь складається з чотирьох компонентів <зміщення по x> <зміщення по y> <радіус> <колір>:

- зміщення по x – зміщення тіні по горизонталі щодо тексту. Додатне значення задає зміщення тіні вправо, від'ємне – вліво. Обов'язковий параметр;
- зміщення по y – зміщення тіні по вертикалі щодо тексту. Додатне значення задає зміщення тіні вниз, від'ємне значення піднімає тінь вище тексту. Обов'язковий параметр;
- радіус – задає радіус розмиття тіні. Чим більше це значення, тим сильніше тінь згладжується, стає ширше і світліше. Якщо цей параметр не заданий, за замовчуванням встановлюється рівним 0. Необов'язковий параметр;
- колір – задає колір тіні у будь-якому доступному CSS-форматі. За замовчуванням колір тіні збігається з кольором тексту. Необов'язковий параметр.

Значення none скасовує додавання тіні. Це значення властивості за замовчуванням.

Допускається вказувати декілька параметрів тіні, розділяючи їх між собою комою. У CSS3 враховується наступний порядок: перша тінь у списку розміщується на самому верху, остання у списку – у самому низу.

Нижче наведено приклад використання властивостей text-decoration і text-shadow, а його відображення у браузері – на рис. 1.5.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Властивості тексту</title>
  <style>
   body {font-family:"Times New Roman"}
   h1  {font-size: 18pt}
   p {font-size: 14pt}
   .tdec {text-decoration: line-through  overline}
```

```
    .shd {text-shadow: 1px 1px 2px black, 0 0 1em
blue; /* Параметри тіни */
      color: white; /* Білий колір тексту */
      font-size: 2em; /* Розмір надпису */}
    </style>
  </head>
  <body>
   <h1>Заголовок</h1>
   <p>Звичайний текст</p>
   <p class=tdec>Текст з перекресленням</p>
   <p class=shd>Текст з тінню</p>
   </body>
</html>
```

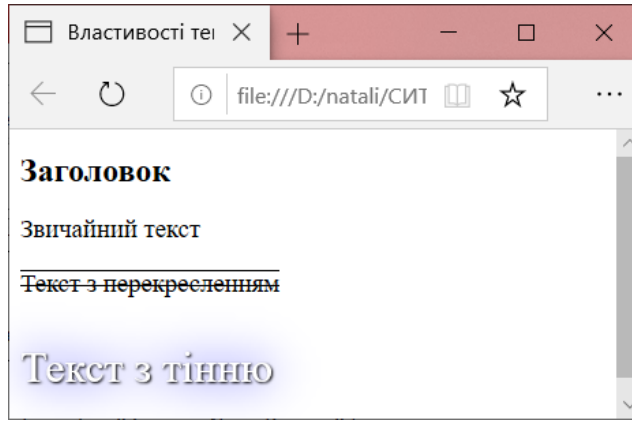

Рис. 1.5. Приклад використання властивостей тексту

Властивість `text-transform` управляє перетворенням тексту елемента у заголовні або прописні символи:

```
text-transform: capitalize|lowercase|uppercase|
none|inherit
```

Можливі значення:

- `capitalize` – перший символ кожного слова у реченні буде заголовним. Решта символів свій вигляд не змінюють;
- `lowercase` – всі символи тексту перетворюються до нижнього регістру;

- `uppercase` – всі символи тексту перетворюються до верхнього регістру;
- `none` – не міняє регістр символів, значення за замовчуванням.

Властивість `letter-spacing` визначає інтервал між символами у межах елементу. Браузери зазвичай встановлюють відстань між символами, виходячи з типу і виду шрифту, його розмірів і налаштувань операційної системи, що відповідає значенню `normal` та є значенням за замовчуванням. Для зміни цього значення допустимо використовувати як додатну величину в одиницях розміру CSS, що відповідає розрядженому шрифту, так і від'ємне значення для ущільненого шрифту. Синтаксис властивості:

```
letter-spacing: значення | normal | inherit
```

Властивість `line-height` встановлює міжрядковий інтервал тексту, відлік ведеться від базової лінії шрифту. За звичайних обставин відстань між рядками залежить від виду і розміру шрифту і визначається браузером автоматично. Це відповідає значенню `normal` та є значенням за замовчуванням. Будь-яке число більше нуля сприймається як множник від розміру шрифту поточного тексту. Наприклад, значення 1.5 встановлює полуторний міжрядковий інтервал. Як значення можуть виступати також будь-які одиниці довжини, прийняті у CSS. Дозволяється використовувати процентну запис, у цьому випадку за 100% береться висота шрифту. Від'ємне значення міжрядкового відстані не допускається. Синтаксис властивості:

```
line-height: множник|значення|відсотки|normal|
inherit
```

Властивість `white-space` встановлює порядок відображення пробілів між словами. Зазвичай браузер будь-яку кількість пробілів у html-коді показується на вебсторінці як один. Винятком є тег `<pre>`, поміщений у цей контейнер текст виводиться з усіма пробілами, як він був відформатований користувачем. Таким чином, `white-space` імітує роботу тега `<pre>`, але на відміну від нього не змінює шрифт на моноширинний:

```
white-space: normal|nowrap|pre|pre-line|pre-wrap|
inherit
```

Можливі значення:

- `normal` – текст у вікні браузера виводиться як зазвичай, переноси рядків встановлюються автоматично, значення за замовчуванням;

- `nowrap` – пробіли не враховуються, переноси рядків у html-коді ігноруються, весь текст відображається одним рядком. Разом з тим додавання тегу `<br>` переносить текст на новий рядок;
- `pre` – текст відображається з урахуванням всіх пробілів і переносів, як вони були додані в html-коді. Якщо рядок виходить занадто довгим і не поміщається у вікні браузера, то буде додана горизонтальна смуга прокрутки;
- `pre-line` – у тексті пробіли не враховуються, текст автоматично переноситься на наступний рядок, якщо він не поміщається у задану область;
- `pre-wrap` – у тексті зберігаються всі пробіли і переноси, однак якщо рядок по ширині не поміщається у задану область, то текст автоматично буде перенесений на наступний рядок.

Властивість `word-break` вказує, як робити перенос рядків усередині слів, які не поміщаються по ширині у задану область:

```
word-break: normal|break-all|keep-all
```

Можливі значення:

- `normal` – застосовуються правила перенесення рядків за замовчуванням. Як правило, у цьому випадку рядки не переносяться або переносяться у тих місцях, де явно задано перенесення, наприклад, за допомогою тегу `<br>`. Це значення властивості за замовчуванням;
- `break-all` – перенесення рядків додається автоматично, щоб слово помістилося у задану ширину блоку. Значення не працює для тексту китайською, корейською або японською;
- `keep-all` – не дозволяє перенесення рядків у словах на китайською, корейською або японською. Для інших мов діє як `normal`.

Властивість `word-spacing` встановлює інтервал між словами. Зазвичай інтервал між словами встановлює браузер автоматично. Це відповідає значенню `normal`, що є значенням за замовчуванням. Якщо для тексту задано вирівнювання за допомогою властивості `text-align` із встановленим значенням `justify` (вирівнювання по ширині), то інтервал між словами буде встановлено примусово, але не менше значення, зазначеного через `word-spacing`. Як значення приймаються будь-які одиниці довжини, прийняті у CSS. Відсоткова запис не допускається. Синтаксис властивості:

```
word-spacing: <розмір>|normal|inherit.
```

Властивість `vertical-align` управляє вертикальним вирівнюванням елемента:

```
vertical-align: baseline|bottom|middle|sub|super|
text-bottom|text-top|top|inherit|значення
```

Можливі значення:

- `baseline` – вирівнює базову лінію поточного елемента за базовою лінією батьківського елемента. Якщо батьківський елемент не має базової лінії, то за неї приймається нижня межа елемента. Це значення за замовчуванням;
- `bottom` – вирівнює основу поточного елемента по нижній частині елемента рядка, розташованого нижче всіх;
- `middle` – вирівнювання середньої точки елемента за базовою лінією батька плюс половина висоти батьківського елемента;
- `sub` – елемент зображується як підрядковий у вигляді нижнього індексу. Розмір шрифту при цьому не змінюється;
- `super` – елемент зображується як надрядковий у вигляді верхнього індексу. Розмір шрифту залишається без змін;
- `text-bottom` – нижня межа елемента вирівнюється за найнижчим краєм поточного рядка;
- `text-top` – верхня межа елемента вирівнюється за найвищим текстовим елементом поточного рядка;
- `top` – вирівнювання верхнього краю елемента по верху найвищого елемента рядка.

Як значення також можна використовувати відсотки, пікселі або інші доступні одиниці. Додатне число зміщує елемент вгору щодо базової лінії, у той час як від'ємне число опускає його вниз. При використанні відсотків відлік ведеться від значення властивості `line-height`, при цьому 0% аналогічно значенню `baseline`.

Якщо властивість `vertical-align` використовується для вирівнювання по вертикалі у комірках таблиці, то можливі значення такі:

- `baseline` – вирівнює базову лінію комірки з базовою лінією першого текстового рядка або іншого вкладеного елемента, значення за замовчуванням;
- `bottom` – вирівнює по нижньому краю комірки;
- `middle` – по середині комірки;
- `top` – вирівнює вміст комірки по її верхньому краю.

Нижче наведено приклад використання різних типів вертикального вирівнювання, а його відображення у браузері – на рис. 1.6.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Вертикальне вирівнювання</title>
  <style>
   body {font-family:"Times New Roman"}
   p {font-size: 14pt}
   #bt {vertical-align: bottom}
   #md {vertical-align: middle}
   #sb {vertical-align: sub}
   #sp {vertical-align: super}
   #tb{vertical-align: text-bottom}
   #tt {vertical-align: text-top}
   #top {vertical-align: top}
    </style>
 </head>
 <body>
  <p>Звичайний текст <span id="bt">bottom</span>
<span id="md">middle</span> <span id="sb">sub</span>
<span id="sp">super</span> <span id="tb">text-
bottom</span> <span id="tt">text-top</span> <span
id="top">top</span></p>
 </body>
</html>
```

1.9. Параметри фону

За допомогою CSS фон можна задати не тільки для всього документа або таблиці та її деякої частини, а й для фрагмента тексту, тобто вбудованого елемента або блокового елемента, яким, наприклад, може виступати декілька абзаців тексту. Але згідно існуючих правил вебдизайну фон, відмінний від основного фону сторінки, треба задавати з обережністю, оскільки сторінка може вийти строкатою та некомфортною для читання.

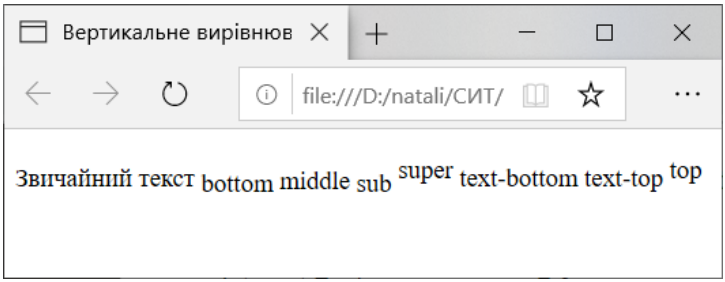

Рис. 1.6. Приклад вертикального вирівнювання тексту

Розглянемо існуючи властивості, що дозволяють задавати параметри фону. Властивість `background-color` служить для задання кольору фону:

```
background-color: <колір>|transparent| inherit
```

Значення `transparent` прибирає фон зовсім; тоді елемент вебсторінки отримає «прозорий» фон. За замовчуванням фон у елементів відсутній, а фон самої вебсторінки задає браузер.

Властивість `background-image` дозволяє призначити в якості фону графічне зображення:

```
background-image: none|url(<адреса файлу>).
```

Для задання адреси файлу використовується функція `url`, причому сама адреса записується у лапках. Наприклад,

```
table {background-image: url(
"/tbl_background.png")}
```

Значення `none` прибирає графічний фон і є значенням за замовчуванням.

Графічний фон виводиться поверх звичайного фону, заданого за допомогою властивості стилю `background-color`. І, якщо фонове зображення містить прозорі фрагменти, що можливо у форматах GIF і PNG, цей фон буде просвічуватися крізь них.

Якщо фонове зображення менше, ніж елемент вебсторінки, для якого воно задано, браузер буде повторювати це зображення, поки не заповнить їм весь елемент. Параметри цього повторення задає властивість `background-repeat`. При цьому у CSS 3 допустимо вказувати кілька значень для кожного фону, перераховуючи значення через кому:

```
background-repeat: <повторення> [,<повторення>],
```

де:

```
<повторення> = repeat-x|repeat-y|[repeat|space|
round|no-repeat]{1,2}
```

Припустимо вказувати два значення, перше ключове слово задає повторення по горизонталі, друге – по вертикалі.

Можливі значення:

- `no-repeat` – встановлює одне фонове зображення в елементі без його повторень, положення якого визначається властивістю `background-position` (за замовчуванням у лівому верхньому куті). Аналогічно `no-repeat no-repeat`;
- `repeat` – фонове зображення повторюється по горизонталі і вертикалі, значення за замовчуванням. Аналогічно `repeat repeat`;
- `repeat-x` – фоновий рисунок повторюється тільки по горизонталі. Аналогічно `repeat no-repeat`;
- `repeat-y` – фоновий рисунок повторюється тільки по вертикалі. Аналогічно `no-repeat repeat`;
- `space` – зображення повторюється стільки раз, щоб повністю заповнити область; якщо це не вдається, між картинками додається додатковий простір;
- `round` – зображення повторюється так, щоб в області помістилося ціле число рисунків; якщо це не вдається зробити, то фонові рисунки масштабуються.

За допомогою властивості стилю `background-position` можна вказати позицію фонового зображення щодо елемента вебсторінки, для якого воно призначене:

```
background-position: <горизонтальна позиція>
[<вертикальна позиція>]|inherit.
```

Горизонтальна позиція фонового зображення задається у наступному форматі:

```
<Числове значення>|left|center|right
```

Числове значення вказує місце розташування фонового зображення в елементі вебсторінки по горизонталі і може бути задано у будь-який з одиниць вимірювання CSS. Також можна вказати значення:

- `left` – фонове зображення притискається до лівого краю елемента вебсторінки, значення за замовчуванням;
- `center` – розташовується по центру;
- `right` – притискається до правого краю.

Формат задання *вертикальної позиції* фонового зображення такий:

```
<Числове значення>|top|center|bottom
```

Числове значення вказується аналогічно попередньому випадку. Також можливі наступні значення:

- `top` – фонове зображення притискається до верхнього краю елемента вебсторінки, значення за замовчуванням;
- `center` – розташовується по центру;
- `bottom` – притискається до нижнього краю.

Якщо для будь-якого елемента вебсторінки вказана тільки позиція фонового зображення по горизонталі, його вертикальна позиція буде дорівнювати значенню `center`.

При прокручуванні вмісту вебсторінки у вікні браузера також зазвичай і фонове зображення Властивість `background-attachment` дозволяє заборонити прокручувати фон:

```
background-attachment: scroll | fixed.
```

Значення `scroll` змушує браузер прокручувати фон разом зі вмістом вебсторінки; це значення за замовчуванням. Значення `fixed` фіксує фон на місці, і він не буде прокручуватися.

Зауваження. Зазвичай цю властивість використовують для фіксування фону, заданого для самої вебсторінки. Графічний фон у окремих її елементів фіксувати не слід.

Властивість `background-origin` визначає область позиціонування фонового зображення. Це властивість не застосовується, коли значення `background-attachment` задано як `fixed`:

```
background-origin: <горизонтальна позиція>
[,<вертикальна позиція>],
```

де

```
<позиція>=padding-box|border-box|content-box.
```

Можливі значення:

- `padding-box` – фон позиціонується щодо краю елемента з урахуванням товщини його межі;
- `border-box` – фон позиціонується щодо межі, при цьому лінія межі може перекривати зображення;
- `content-box` – фон позиціонується щодо вмісту елемента.

Універсальна властивість `background` дозволяє встановити одночасно до п'яти характеристик фону. Значення можуть йти у будь-якому порядку, браузер сам визначить, яке з них відповідає потрібній властивості. Наприклад,

```
div {background: url(img/fon.gif) repeat-y #fc0}
```

1.10. Параметри списків

Списки широко використовуються на вебсторінках завдяки тому, що є ефективним способом подання інформації. У HTML існують *марковані списки*, що створюються за допомогою тега `<ul>`, і *нумеровані* – тег `<ol>`. Розглянемо стильові властивості, що дозволяють управляти їх параметрами.

Властивість `list-style-type` змінює вид маркера для кожного елемента списку. Вона використовується тільки у разі, коли значення `list-style-image` встановлено як `none`:

```
list-style-type: circle|disc|square|armenian|
decimal|decimal-leading-zero|georgian|lower-alpha|
lower-greek|lower-latin|lower-roman|upper-alpha|
upper-latin|upper-roman|none|inherit
```

Значення залежать від типу списка. Для маркованого списка:

- `circle` – маркер у вигляді кола;
- `disc` – маркер у вигляді крапки, значення за замовчуванням;
- `square` – маркер у вигляді квадрата.

Для нумерованого списку:

- `armenian` – традиційна вірменська нумерація;
- `decimal` – маркер у вигляді арабських чисел, значення за замовчуванням;
- `decimal-leading-zero` – арабські числа з нулем попереду для цифр менше десяти;
- `georgian` – традиційна грузинська нумерація;

- `lower-alpha` – рядкові латинські літери;
- `lower-greek` – рядкові грецькі літери;
- `lower-latin` – це значення аналогічно `lower-alpha`;
- `lower-roman` – римські числа у нижньому регістрі;
- `upper-alpha` – великі латинські літери;
- `upper-latin` – це значення аналогічно `upper-alpha`;
- `upper-roman` – римські числа у верхньому регістрі.

Значення `none` скасовує маркер як для маркованого, так і для нумерованого списку.

Властивість `list-style-type` можна задавати як для самих списків, так і для окремих їх пунктів. В останньому випадку створюється список, в якому пункти мають різні маркери або нумерацію.

Властивість `list-style-image` дозволяє задати в якості маркера пунктів списку графічне зображення. У цьому випадку значення властивості `list-style-type`, якщо така задана, ігнорується:

```
list-style-image: none|<адреса файлу зображення>|
inherit.
```

Інтернет-адреса файлу з графічним маркером задається у такому ж форматі, що і інтернет-адреса файлу фонового зображення (див. п.1.9):

```
ul {list-style-image: url(/markers/dot.gif)}
```

Значення `none` прибирає графічний маркер і повертає простий, неграфічний. Це значення за замовчуванням.

Властивість `list-style-image` також можна задавати як для самих списків, так і для окремих їх пунктів.

Властивість `list-style-position` дозволяє вказати місце розташування маркера або нумерації у пункті списку, якщо текст пункту переноситься на наступний рядок:

```
list-style-position: inside|outside|inherit
```

Доступні значення:

- `inside` – маркер є частиною текстового блоку і відображається в елементі списку;
- `outside` – текст вирівнюється по лівому краю, а маркери розміщуються за межами текстового блоку, значення за замовченням.

Універсальна властивість `list-style` дозволяє одночасно задати стиль маркера, його положення, а також зображення, яке буде використовуватися в якості маркера. Наприклад,

```
ul {list-style: square outside}
```

1.11. Параметри блочних елементів

1.11.1. *Поняття контейнерних і блочних елементів*

При розробці сайту іноді необхідно застосувати стиль до довільного фрагменту тексту. Наприклад, виділити напівжирним шрифтом фрагмент абзацу, не використовуючи при цьому застарілі теги типу `<b>` або `<strong>` (див. рис. 1.1 та відповідний приклад). Для цього в HTML передбачені *контейнерні елементи* або *контейнери*.

Контейнер – це елемент вебсторінки, призначений тільки для виділення будь-якого її фрагмента. Таким фрагментом може бути частина блочного елемента (абзацу, заголовка, цитати, тексту фіксованого форматування і ін.), блочний елемент або відразу кілька блочних елементів. При цьому браузер ніяк не виділяє контейнер на вебсторінці.

Контейнер слугує двом цілям. По-перше, за його допомогою можна прив'язати до певного елементу або елементів вебсторінки потрібний стиль. По-друге, він може забезпечувати прив'язку поведінки до елементу або елементів вебсторінки.

Контейнери бувають *блочні* і *вбудовані*. Узагальненим блочним контейнером є тег `<div>`, а вбудованим – `<span>`. При цьому вбудований контейнер є частиною блочного елемента вебсторінки.

Блочні елементи дозволяють оперувати з текстом у термінах прямокутників, які цей текст займає. При цьому блок тексту стає елементом дизайну сторінки з теми ж властивостями, що і картинка, таблиця або прямокутна область застосунку.

Блочні елементи мають властивості:

- розміру;
- межі (`border`);
- відступу (`margin`) – зовнішній відступ;
- полів (`padding`) – внутрішній відступ;
- розміщення (`float`);
- обтікання (`clear`).

1.11.2. *Параметри розміру*

Властивості стилю `width` і `height` дозволяють задати, відповідно, ширину і висоту елемента вебсторінки:

```
width: auto|<ширина>|inherit
height: auto|<висота>|inherit
```

Можна задати як абсолютне у будь-яких одиницях CSS, так і відносне у відсотках від відповідного розміру батьківського елемента значення розміру. Значення `auto` віддає управління розміром браузеру, це значення за замовчуванням.

Часто при заданні відносної ширини або висоти елемента також вказують його мінімальні і максимальні можливі розміри. Зрозуміло, що при їх заданні значення розміру не зможе перевищити максимального і стати менше мінімального.

Властивості `min-width` і `min-height` дозволяють визначити мінімальну ширину і висоту елемента вебсторінки:

```
min-width: <ширина>|inherit
min-height: <висота>|inherit
```

Значення ширини або висоти може бути як абсолютним, так і відносним. Значенням за замовчуванням є `0`.

Аналогічно за допомогою властивостей `max-width` і `max-height` можна задати максимальну ширину і висоту елемента вебсторінки. Синтаксис властивостей співпадає з попереднім випадком.

1.11.3. *Параметри межі*

CSS дозволяє створити суцільну, пунктирну або точкову рамку по уявній межі елемента вебсторінки.

Властивості `border-left-width`, `border-top-width`, `border-right-width` і `border-bottom-width` задають товщину, відповідно, лівої, верхній, правій і нижній сторін рамки. Синтаксис всіх властивостей однаковий, тому він показаний нижче на прикладі властивості `border-left-width`:

```
border-left-width: thin|medium|thick|<товщина>
```

Можна вказати абсолютне або відносне числове значення товщини рамки, або одне з символьних значень: `thin` (тонка), `medium` (середня, значення за замовчуванням) або `thick` (товста). В останньому випадку реальна товщина рамки залежить від браузера.

Узагальнена властивість `border-width` дозволяє вказати значення товщини відразу для всіх сторін рамки, вказуючи через пробіл від одного до чотирьох значень:

- якщо вказано одне значення, воно задає однакову товщину усіх боків рамки;
- якщо вказані два значення, перше задає товщину верхньої і нижньої, а друге – лівої і правої сторін рамки.
- якщо вказані три значення, перше задає товщину верхньої, друге – лівої і правої, а третє – нижньої сторін рамки;
- якщо вказані всі чотири значення, перше задає товщину верхньої, друге – правої, третє – нижньої, а четверте – лівої сторін рамки.

Властивості стилю `border-left-color`, `border-top-color`, `border-right-color` і `border-bottom-color` задають колір, відповідно, лівої, верхній, правій і нижній сторін рамки:

```
border-left-color: transparent|<колір>|inherit
```

Значення `transparent` задає прозорий колір, крізь який буде просвічуватися фон батьківського елемента.

Узагальнена властивість `border-color` дозволяє вказати колір відразу для всіх сторін рамки. Вона веде себе так само, як і властивість `border-width`.

Властивості стилю `border-left-style`, `border-top-style`, `border-right-style` і `border-bottom-style` задають стиль ліній, якими буде нарисована, відповідно, ліва, верхня, права і нижня сторона рамки:

```
border-left-style: none | hidden |dotted | dashed|
solid | double | groove |ridge | inset | outset |
inherit
```

Тут доступні наступні значення:

- `none` – рамка відсутня, значення за замовчуванням;
- `hidden` – має той же ефект, що і `none` за винятком застосування властивостей до комірок таблиці, в якій значення властивості `border-collapse` встановлено як `collapse`. У цьому випадку нижня межа у комірці не буде доступна взагалі;
- `dotted` – пунктирна лінія;
- `dashed` – штрихова лінія;

- `solid` – суцільна лінія;
- `double` – подвійна лінія;
- `groove` – ефект тривимірної втиснутої лінії;
- `ridge` – ефект тривимірної опуклої лінії;
- `inset` – тривимірна опуклість;
- `outset` – тривимірне втиснення.

Узагальнена властивість `border-style` дозволяє вказати стиль відразу для всіх сторін рамки за правилами, аналогічними до розглянутих для властивості `border-width`.

Існують також узагальнені властивості `border-left`, `border-top`, `border-right` і `border-bottom` дозволяють задати всі параметри для, відповідно, лівої, верхній, правій і нижній сторони рамки.. Наприклад,

```
h1 {border-bottom: 5px double red}.
```

Узагальнена властивість `border` дозволяє задати всі параметри для всіх сторін рамки.

Нижче подано приклад різних стилів для рамок комірок таблиці, а його відображення у браузері – на рис. 1.7.

```
<!DOCTYPE html>
<html>
 <head>
   <meta charset="utf-8">
  <title>border</title>

  <style>
  table {
   background: #ccc;
   width: 100%;
   color: red;
   font-size: 20pt;
   text-align: center;
   border-spacing: 10px; /*Відступ між комірками*/
   }
   td {    border-color: white;
           border-width: 5px}
   #hd {border-style: hidden}
   #dt {border-style: dotted}
   #ds {border-style: dashed}
   #sl {border-style: solid}
```

```
    #dbl {border-style: double}
    #gr {border-style: groove}
    #rd {border-style: ridge}
    #in {border-style: inset}
    #out {border-style: outset}
   </style>
  </head>
  <body>
   <table>
     <tr><td>none</td><td id=hd>hidden</td></tr>
     <tr><td id=dt>dotted</td>
         <td id=ds>dashed</td></tr>
     <tr><td id=sl>solid</td>
         <td id=dbl>double</td></tr>
    <tr><td id=gr>groove</td>
        <td id=rd>ridge</td></tr>
    <tr><td id=in>inset</td>
        <td id=out>outset</td></tr>
   </table>
</body>
</html>
```

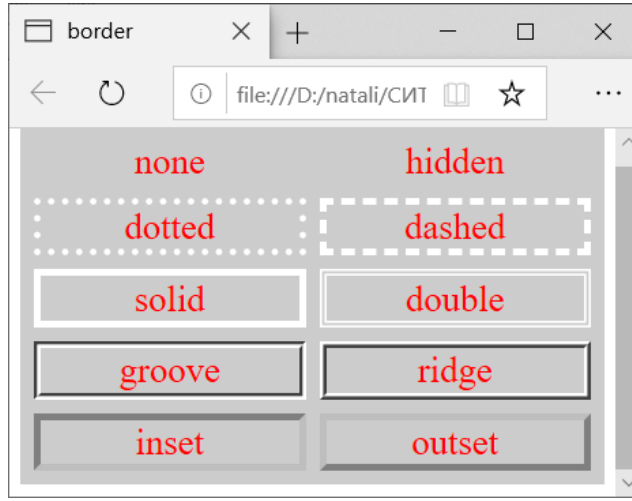

Рис. 1.7. Приклад параметрів рамок

У наведеному прикладі рамки у комірок таблиці дублюються. Але на практиці таблиця зазвичай має одну рамку між сусідніми комірками. Для

управління такою можливістю у CSS існує властивість border-collapse, що використовується для тегу <table>:

```
border-collapse: collapse| separate |inherit.
```

Можливі значення:

- collapse – лінія між комірками відображається тільки одна, також ігнорується значення атрибута cellspacing;
- separate – навколо кожної комірки відображається своя власна рамка, у місцях зіткнення комірок показуються відразу дві лінії, значення за замовчуванням.

Властивість border-spacing задає відстань між рамками комірок у таблиці. Вона не працює у разі, коли для таблиці встановлено властивість border-collapse із значенням collapse:

```
border-spacing: значення1 [значення2]
```

Одне встановлене значення задає одночасно відстань по вертикалі і горизонталі між рамками комірок. Якщо значень два, то перше задає горизонтальну відстань, а друге – вертикальну. За замовчуванням це значення дорівнює нулю.

Однією з особливостей CSS 3 є можливість створення заокруглених границь в елементів. Вона реалізується використанням властивостей border-bottom-left-radius, border-top-left-radius, border-bottom-right-radius, border-top-right-radius, що дозволяють встановити радіус скруглення з відповідної сторони та мають однаковий синтаксис, наприклад:

```
border-bottom-left-radius: [значення|відсотки]
[значення|відсотки]
```

Як значення вказуються числа у будь-якому допустимому для CSS форматі. Якщо значення вказано у відсотках, то радіус заокруглення розраховується від ширини блоку. Необов'язкове друге значення призначене для створення еліптичного кута, перше значення при цьому встановлює радіус по горизонталі, а друге – радіус по вертикалі. Значення за замовчуванням дорівнює нулю.

Узагальнена властивість border-radius дозволяє одночасно встановити від одного до чотирьох значень для границь, при цьому значення, записані через слеш (/), дозволяють створити еліптичні кути:

```
border-radius: <радіус> {1,4} [/<радіус> {1,4}]
```

У <радіус> можна використовувати одне, два, три або чотири значення, перераховуючи їх через пробіл, за наступними правилами:

- одне значення вказує однаковий радіус для всіх чотирьох кутів границі;
- якщо вказано два значення, то перше значення задає радіус верхнього лівого і нижнього правого кута, друге значення – верхнього правого і нижнього лівого кута;
- якщо вказано три значення, то перше значення задає радіус для верхнього лівого кута, друге – одночасно для верхнього правого і нижнього лівого, а третє – для нижнього правого кута;
- задані чотири значення по черзі встановлюють радіус для верхнього лівого, верхнього правого, нижнього правого і нижнього лівого кутів.

Нижче подано приклад різних рамок зі заокругленнями, а його відображення у браузері – на рис. 1.8.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>border-radius</title>
  <style>
   div {
    background: #ccc;
    border: 1px solid black;
    padding: 15px; /* Поля навколо тексту */
    margin-bottom: 10px; /* Відступ знизу */
    font-size: 14pt}
   #r1 {border-radius: 50px 0 0 50px}
   #r2 {border-radius: 40px 10px}
   #r3 {border-radius: 13em/3em}
   #r4 {border-radius: 13em 0.5em/1em 0.5em}
   #r5 {border-radius: 10px}
  </style>
 </head>
 <body>
  <div id="r1">border-radius: 50px 0 0 50px</div>
  <div id="r2">border-radius: 40px 10px</div>
  <div id="r3">border-radius: 13em/3em</div>
  <div id="r4">border-radius: 13em 0.5em/1em
0.5em</div>
```

```
  <div id="r5">border-radius: 10px</div>
 </body>
</html>
```

1.11.4. *Параметри відступів*

Стандарт CSS пропонує засоби для створення відступів двох типів:

- відступ між уявною межею елемента вебсторінки і його вмістом – *внутрішній відступ*. Такий відступ належить даному елементу вебсторінки і знаходиться всередині його;
- відступ між уявною межею даного елемента вебсторінки і уявними межами сусідніх елементів вебсторінки – *зовнішній відступ*. Такий відступ не належить даному елементу вебсторінки і знаходиться поза ним.

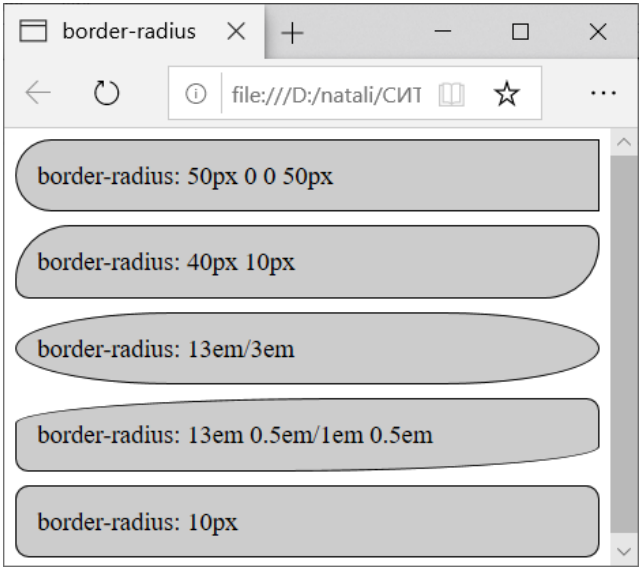

Рис. 1.8. Приклад рамок зі заокругленнями

Щоб краще зрозуміти різницю між внутрішнім і зовнішнім відступами, розглянемо комірку таблиці, що наповнена текстом, має уявну межу і оточена іншими комірками. Тоді:

- внутрішній відступ – це відступ між межею комірки і текстом, що у ній міститься;
- зовнішній відступ – це відступ між межами окремих комірок таблиці.

Властивості `padding-left`, `padding-top`, `padding-right` і `padding-bottom` дозволяють задати величини *внутрішніх відступів*, відповідно, ліворуч, зверху, праворуч і знизу елемента вебсторінки:

```
padding-left: <відступ> | inherit
```

В якості величини відступу можна вказати як абсолютне, так і відносне значення. Значення за замовчуванням дорівнює нулю.

Узагальнена властивість `padding` дозволяє відразу вказати величини внутрішніх відступів з усіх боків елемента вебсторінки, вказуючи від одного до чотирьох значень за правилами, що описані для властивості `border-width`:

```
padding: <відступ> {1, 4} | inherit
```

Для задання *зовнішніх відступів* використовуються властивості `margin-left`, `margin-top`, `margin-right` і `margin-bottom`, а також узагальнена властивість `margin`. Правила їх використання аналогічні до описаних вище для властивостей `padding`.

1.11.5. *Параметри розміщення і обтікання*

За замовченням блочні елементи вебсторінки розташовуються на ній по вертикалі, строго один за одним, у тому порядку, в якому вони визначені в html-коді. Однак стандарт CSS дозволяє встановити для блочного елемента вирівнювання по лівому або краю батьківського елемента, тобто блочного контейнера, в який він вкладений, або самої вебсторінки. При цьому блочний елемент буде притискатися до відповідного краю батька, а решта вмісту буде обтікати його з протилежного боку.

Властивість стилю `float` задає, з якого краю батьківського елементу буде вирівнюватися даний елемент вебсторінки:

```
float: left | right| none | inherit
```

Можливі три значення:

- `left` – елемент вебсторінки вирівнюється по лівому краю батьківського елемента, а решта вмісту обтікає його справа;
- `right` – елемент вебсторінки вирівнюється по правому краю батьківського елемента, а решта вмісту обтікає його зліва;
- `none` – елемент розміщається за своїм попередником і розташовується нижче його, значення за замовченням. Наприклад:

```
table.left-aligned {float: left}
```

Після застосування даного стилю до таблиці вона буде вирівняна по лівому краю батьківського елемента, а решта вмісту буде обтікати її справа.

Властивість стилю `float` зазвичай застосовують для блочних контейнерів, формуючих дизайн вебсторінок. Такі контейнери називають *плаваючими*.

При створенні контейнерного вебдизайну, заснованого на плаваючих контейнерах, часто доводиться поміщати будь-які контейнери нижче тих, що були вирівняні по лівому або правому краю батьківського елемента. Якщо просто прибрати у них властивість `float` або задати для неї значення `none`, результат буде непередбачуваним. Тому CSS надає можливість однозначно вказати, що даний блочний елемент у будь-якому випадку повинен розташовуватися нижче плаваючих контейнерів, що передують йому. Для цього служить властивість стилю `clear`:

```
clear: left | right | both | none | inherit
```

Як можна бачити, доступних значень тут чотири:

- `left` – елемент вебсторінки повинен розташовуватися нижче всіх елементів, для яких у властивості `float` задано значення `left`;
- `right` – елемент вебсторінки повинен розташовуватися нижче всіх елементів, для яких у властивості `float` задано значення `right`;
- `both` – елемент вебсторінки повинен розташовуватися нижче всіх елементів, для яких у властивості `float` задано значення `left` або `right`;
- `none` – скасовує дію властивості `clear`, при цьому обтікання елемента відбувається, як задано за допомогою властивості `float` або інших налаштувань, значення за замовченням.

Нижче подано приклад створення плаваючого блоку тексту, а його відображення у браузері – на рис. 1.9.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>float</title>
  <style>
   div {
     margin: 10px;
     font-size: 14pt;
         }
   #layer {
```

```
    float: left;
    background: #ccc;
    border: 1px solid black;
    padding: 10px;
    width: 40%;
    }
  </style>
 </head>
 <body>
    <div id="layer">
  Важлива інформація
  </div>
    <div>
    <p>За замовченням блочні елементи вебсторінки
розташовуються на ній по вертикалі, строго один за
одним, у тому порядку, в якому вони визначені в
html-коді. Однак стандарт CSS дозволяє встановити
для блочного елемента вирівнювання по лівому або
краю батьківського елемента, тобто блочного
контейнера, в який він вкладений, або самої
вебсторінки.</p>
  </div>
</body>
</html>
```

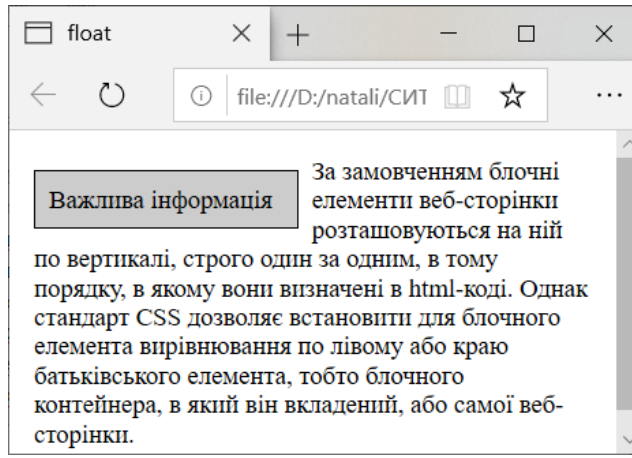

Рис. 1.9. Приклад створення плаваючого блока

1.12. Параметри виділення

Існує багато способів привернути увагу відвідувача до певних елементів вебсторінок, використавши теги HTML або властивості стилю CSS. За допомогою CSS 3 це можна зробити у ще один спосіб, використовуючи так зване *виділення*.

Виділення CSS 3 являє собою рамку, якою оточується даний елемент вебсторінки. Для виділення можна задавати товщину, колір і стиль. На відміну від рамки, створеної за допомогою властивостей `border`, виділення не збільшує розміри елемента вебсторінки. Тому виділення широко використовується при контейнерному дизайні сайту.

Для задання параметрів виділення CSS 3 призначений чотири спеціальні властивості стилю. При цьому їх синтаксис та можливі значення співпадають з відповідними для властивостей `border`, що були розглянуті раніш:

- властивість `outline-width` задає товщину рамки виділення;
- властивість `outline-color` задає колір рамки виділення;
- властивість `outline-style` задає стиль ліній, якими буде нарисована рамка виділення;
- узагальнена властивість `outline` дозволяє задати відразу всі параметри для рамки виділення.

Нижче подано приклад використання властивості `outline`, а його вид у браузері – на рис. 1.10.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>outline</title>
  <style>
   img {
    padding: 20px; /* Поля навколо зображення*/
    margin-right: 10px; /* Відступ справа */
    margin-bottom: 10px; /* Відступ знизу */
    outline: 1px solid black; /* Виділення*/
    background: #ccc; /* Колір фону */
    float: left; /* Обтікання праворуч */
   }
  </style>
 </head>
```

```
 <body>
   <img src="img/erizo1.jpg" alt="Їжачок1">
   <img src="img/erizo2.jpg" alt="Їжачок2">
</body>
</html>
```

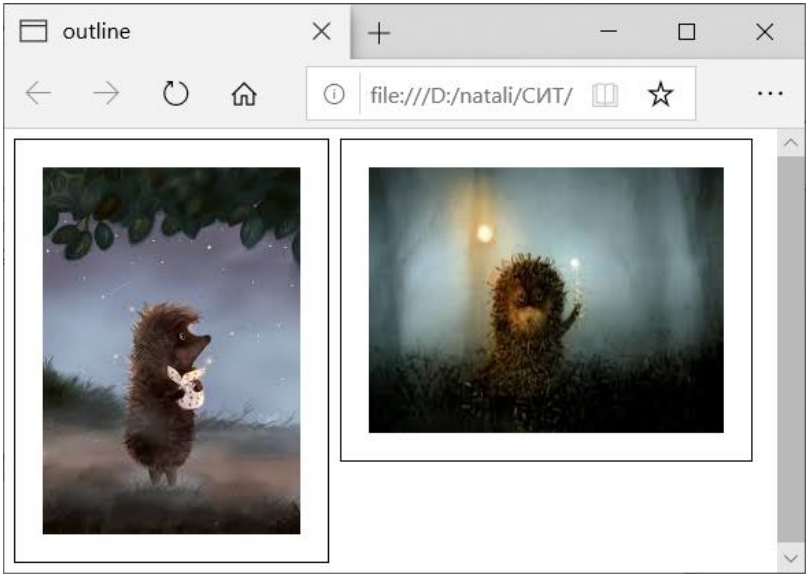

Рис. 1.10. Приклад виділення

1.13. Управління видимістю елементів

Ще одна група властивостей стилю управляє тим, як елемент буде відображатися на вебсторінці, тобто буде він блочним або вбудованим, і чи буде він відображатися взагалі. Ці властивості можуть бути застосовні до будь-яких елементів вебсторінок.

Властивість `visibility` дозволяє вказати, чи буде елемент відображатися на вебсторінці:

```
visibility: visible | hidden | collapse | inherit
```

Він може приймати три значення:

- `visible` – елемент відображається на вебсторінці, значення за замовчуванням;

- `hidden` – елемент не відображається на вебсторінці, проте під нього все ще виділено на ній місце. Іншими словами, замість елемента на вебсторінці видно порожнє місце;
- `collapse` – якщо це значення застосовується не до рядків або колонок таблиці, то результат його використання буде таким же, як `hidden`. У разі використання `collapse` для вмісту комірок таблиць, то задані рядки і колонки прибираються, а таблиця перебудовується.

Властивість `visibility` застосовується рідка і тільки для створення спеціальних ефектів, на зразок елемента, що зникає або з'являється, тим самим додаючи вебсторінці поведінку за допомогою JavaScript.

Багатофункціональна властивість `display` дозволяє задати вигляд елемента вебсторінки: буде він блоковим, вбудованим, пунктом списку, чи буде взагалі відображатися:

```
display: none | inline | block | inline-block |
list-item | run-in | table |inline-table |
table-caption | table-column | table-column-group|
table-header-group | table-row-group |
table-footer-group | table-row | table-cell |
inherit
```

Доступних значень у цієї властивості багато:

- `none` – елемент не буде відображатися на вебсторінці, а зайняте їм місце звільняється, тобто вебсторінка перебудовується;
- `inline` – вбудований елемент;
- `block` – блочний елемент;
- `inline-block` – блочний елемент, який буде обтікатися вмістом інших сусідніх блокових елементів;
- `list-item` – елемент виводиться як блочний і до нього додається маркер списку;
- `run-in` – встановлює елемент як блоковий або вбудований у залежності від контексту. Якщо за таким елементом йде блочний елемент, він стає частиною даного блочного елемента (фактично вбудованим у нього елементом), в іншому випадку він сам стає блочним елементом;
- `table` – визначає, що елемент є блочною таблицею; аналог використання тегу `<table>`;
- `inline-table` – таблиця, відформатована як вбудований елемент;

- `table-caption` – заголовок таблиці; аналог використання тега `<caption>`;
- `table-column` – стовпець таблиці; аналог використання тега `<col>`;
- `table-column-group` – група стовпців таблиці; аналог використання тега `<colgroup>`;
- `table-header-group` – секція заголовка таблиці; аналог використання тега `<thead>`;
- `table-row-group` – секція тіла таблиці; аналог використання тега `<tbody>`;
- `table-footer-group` – секція завершення таблиці; аналог використання тега `<tfoot>`;
- `table-row` – рядок таблиці; аналог використання тегу `<tr>`;
- `table-cell` – комірка таблиці; аналог використання тегів `<td>` або `<th>`;

Зауваження. Деякі значення властивості `display` певні браузери можуть не підтримувати.

Значення за замовчуванням властивості `display` залежить від конкретного елемента вебсторінки. Так, для абзацу значенням за замовчуванням буде `block`, а для графічного зображення – `inline`.

Властивість `opacity` дозволяє зробити будь-який елемент вебсторінки частково або повністю прозорим. Дана властивість змінює прозорість елементів, для яких встановлено фонове зображення (картинка) або заданий фон за допомогою кольору або градієнта. Якщо елемент, для якого застосовано властивість `opacity`, містить у собі інші елементи, то вони також змінять свою прозорість.

Властивість `opacity` приймає значення у діапазоні від 0 (повністю прозорий) до 1 (непрозорий),

Нижче подано приклад використання властивості `opacity`, а його вид у браузері – на рис. 1.11.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>opacity</title>
  <style>
   img {
```

```
    padding: 20px; /* Поля навколо зображення*/
    margin-right: 10px; /* Відступ справа */
    margin-bottom: 10px; /* Отступ снизу */
    outline: 1px solid black; /* Параметри
виділення*/
    background: #ccc; /* Колір фону */
    float: left; /* Обтікання праворуч */
   }
  #op {opacity: 0.5}
  </style>
 </head>
 <body>
   <img src="img/erizo1.jpg" alt="Їжачок1">
   <img src="img/erizo1.jpg" alt="Їжачок1
напівпрозорий" id="op">
  </body>
</html>
```

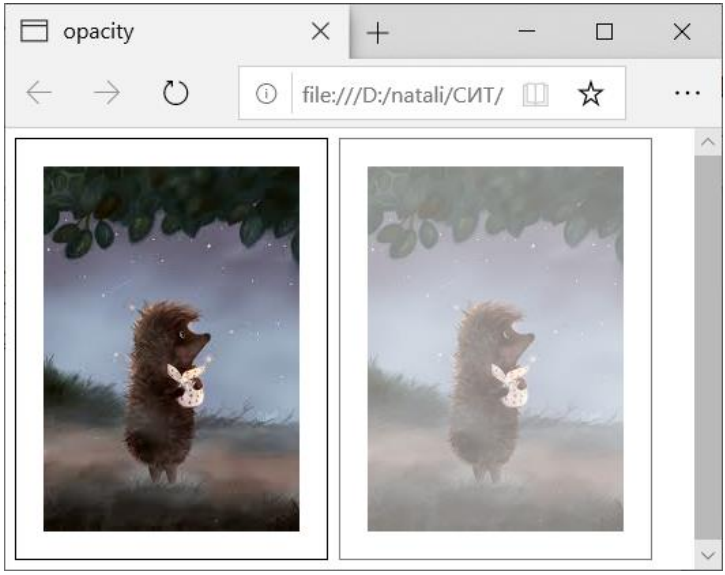

Рис. 1.11. Приклад використання напівпрозорості

Властивість `overflow` управляє відображенням вмісту блочного елемента, якщо він цілком не вміщується і виходить за область заданих розмірів:

`overflow: auto |hidden| scroll | visible |inherit.`

Можливі значення:

- `visible` – розміри блочного елемента збільшуються, щоб повністю помістити весь вміст, значення за замовчуванням;
- `hidden` – вміст, що не поміщається у блочний елемент, буде обрізано. Блочний елемент збереже свої розміри;
- `scroll` – у блочному елементі з'являться смуги прокручування, за допомогою яких можна переглянути вміст. Ці смуги прокручування бу-дуть присутні завжди, навіть якщо у них немає потреби;
- `auto` – смуги прокручування з'являться у блочному елементі, тільки якщо у них виникне необхідність.

Властивості стилю `overflow-x` і `overflow-y` задають поведінку блочного елемента при виході вмісту за межі його кордонів, відповідно по горизонталі і вертикалі. Синтаксис і доступні значення у них ті ж, що і у властивості `overflow`.

1.14. Позиціювання елементів за допомогою CSS

1.14.1. *Базовий потік документа*

HTML-документ складається з великої кількості елементів, вкладених один до одного. Щоб з цих елементів і CSS побудувати зображення сторінки, їх необхідно якось у ній розташувати. За замовчуванням розташування всіх елементів на сторінці здійснюється у *нормальному* або *базовому потоці*. Це означає наступне:

- виведення елементів на вебсторінку браузер здійснює у тому порядку, в якому вони слідують у html-коді. Нижче наведено фрагмент html-коду, а його схематичне відображення у браузері – на рис. 1.12;

```
<body>
  <div>1</div>
  <div>2</div>
  <div>3</div>
</body>
```

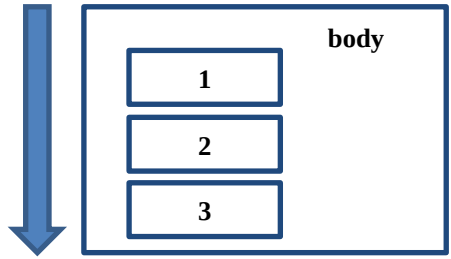

Рис. 1.12. Просте розташування html-елементів

- в html-коді елементи вкладені один до одного. Для врахування цього при виведенні використовують так звані *уявні шари* для відображення елементів. При цьому шар елемента тим вище (ближче до користувача), чим даний елемент є більш вкладеним у коді, тобто глибше розташований у ньому (див. нижче фрагмент html-коду та його схематичне відображення у браузері на рис. 1.13);

```
<body>
   Цей елемент знаходиться позаду інших елементів.
   <div>
     Цей вкладений елемент належить уявному шару,
який заходиться поверх шару батька.
     <span> Цей елемент ще ближче, його шар
розташовується над шаром його батька. </span>
   </div>
</body>
```

Цей елемент знаходиться позаду інших елементів.

Цей вкладений елемент належить уявному шару, який заходиться поверх шару батька. Цей елемент ще ближче, його шар розташовується над шаром його батька.

Рис. 1.13. Розташування html-елементів у вигляді уявних шарів

- положення елемента у потоці залежить від значення властивості `display`. Наприклад, елементи, що має блочне відображення (`display: block`) відображаються у потоці як прямокутні області, кожна з яких на новій лінії одна під одною зверху вниз.

Ширина елементів з *блочним відображенням* за замовчуванням дорівнює доступній ширині батьківського елемента, тобто елемента, в який кожен з них безпосередньо вкладений. Висота їх за замовчуванням дорівнює такій величині, якої буде достатньо, щоб відобразити весь контент, який знаходиться у кожному з них.

Елементи з *рядковим відображенням* (`display: inline`) виводяться інакше. Вони на відміну від блочних елементів не розміщуються кожен на новому рядку, а слідують один за одним зліва праворуч. Якщо простір праворуч закінчився, то вони переносяться на наступний рядок, а не на нову лінію як елементи з блочним відображенням.

Крім `block`, `inline` є й інші варіанти відображення елементів, але всі вони розташовуються у базовому потоці документа.

Нижче наведено фрагмент html-коду та його схематичне відображення у браузері – на рис. 1.14.

```
<body>
  <div class="block-1">
    <span class="inline-1">inline 1</span>
    <span class="inline-2">inline 2</span>
    <span class="inline-3">inline 3</span>
    <span class="inline-4">inline 4</span>
  </div>
  <div class="block-2">
    <span class="inline-1">inline 1</span>
    <span class="inline-2">inline 2</span>
  </div>
  <div class="block-3">
    <span class="inline-1">inline 1</span>
    <span class="inline-2">inline 2</span>
    <span class="inline-3">inline 3</span>
  </div>
</body>
```

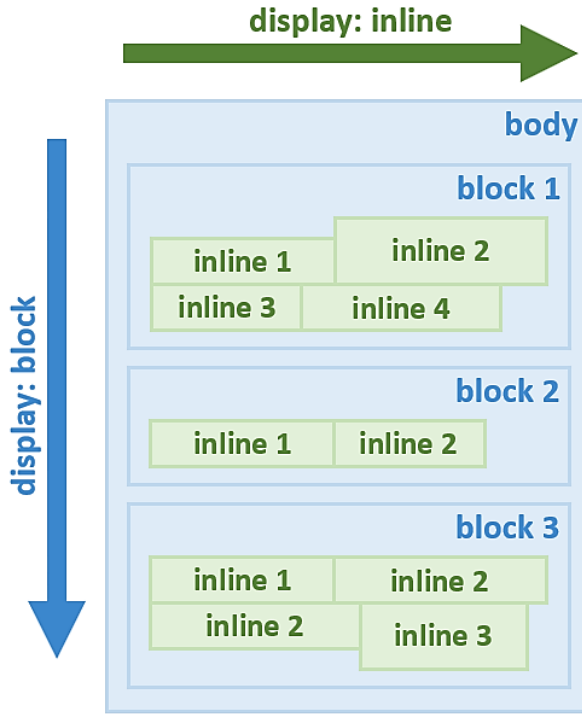

Рис. 1.14. Розташування html-елементів у залежності від властивості `display`

1.14.2. *Властивості позиціювання*

У CSS існують властивості, за допомогою яких елементи можна виключити з основного потоку документа і задати їм інше положення поза базового потоку елементів. До цих властивостей відносяться `position` і `float`, що була розглянута раніше.

Властивість `position` – це одна з властивостей, за допомогою якої можна змінити базову поведінку елементів у потоці:

```
position: absolute | fixed | relative | static |
inherit
```

Властивість має значення:

- `static` – статичне позиціонування, значення за замовчуванням. Воно означає, що елемент знаходиться у базовому потоці;

- `relative` – відносне позиціонування;
- `absolute` – абсолютне позиціонування;
- `fixed` – фіксоване позиціонування.

Відмінності між типами позиціювання буде розглянуто нижче більш докладно.

Кожен елемент у потоці займає певну область, але область елемента не завжди за ним зберігається при його позиціонуванні. Це, наприклад, відбувається при заданні елементу абсолютного або фіксованого позиціювання. Тоді інші елементи його «не бачать» і розташовуються, ігноруючи його присутність у html-коді.

Після встановлення типу позиціювання необхідно розмістити елемент. Для цього існують властивості `top` і `left`, що визначають позицію лівого верхнього кута блоку по вертикалі і горизонталі щодо батьківського елементу, не включаючи відступ, поле і ширину рамки.

Аналогічно властивості стилю `right` і `bottom` визначають позицію правого нижнього кута блоку. Всі чотири властивості мають однаковий синтаксис:

```
left: значення | відсотки | auto | inherit
```

Значення координат задаються у будь-яких одиницях виміру CSS. Вони можуть бути як додатними, так і від'ємними. При заданні значення у відсотках, положення елемента обчислюється залежно від розмірів батьківського елементу. Так, для властивостей `left` і `right` в якості розміру буде розглядатися ширина батьківського елемента, а для `top` і `bottom` – його висота.

Відлік координат залежить від значення властивості `position`. Якщо вона дорівнює `absolute`, як батько виступає вікно браузера і положення елемента визначається від його лівого краю.

Якщо у деякій області вебсторінки знаходяться кілька елементів, які позиціонуються, то вони можуть перекривати один одного у певному порядку. При цьому за замовчуванням вище виявляється той елемент, який нижче описаний в html-коді. Але порядок перекриття елементів (їх положення перпендикулярне екрану, тобто уздовж осі Z) можна змінити. Це здійснюється у CSS за допомогою властивості `z-index`:

```
z-index: число | auto | inherit
```

Властивість `z-index` може приймати додатні, від'ємні цілі числа, нуль та `auto`. При цьому чим більше в елемента значення `z-index`, тим

ближче він розташовується до користувача, і, отже, перекриває всі елементи у цій області, в яких значення `z-index` менше.

Значення `auto` свідчить про те, що порядок елементів будується автоматично, виходячи з їхнього положення в html-коді і приналежності до батьківського елемента, оскільки дочірні елементи мають той же номер, що їх батьківський елемент.

1.14.3. *Статичне позиціювання*

Властивість `position` зі значенням `static` елементів призначається за умовчанням. Це значення означає, що елемент не позиціонується, тобто відображається як зазвичай (в потоці).

Явна установка елементу css-властивості `position: static` може знадобитися тільки у тому випадку, коли потрібно перевизначити інше значення `position`, що був встановлений елементу.

Встановлення властивостей для задавання положення елемента `left`, `top`, `right` і `bottom` ніякого впливу на нього не робить, тому що його місцезнаходження визначається потоком документа.

Нижче подано приклад статичного позиціювання двох блоків елементів, а його відображення у браузері – на рис. 1.15.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>position:static</title>
  <style>
   div {
     font-size: 16pt;
     width: 200px;
     border: 1px solid black;
     }
   .bl1 {
      height: 100px;
      background: #aaa
   }
   .bl2 {
      height: 50px;
      background: #eee
   }
  </style>
 </head>
```

```
 <body>
   <div class="bl1">Блок 1 з position:static</div>
   <div class="bl2">Блок 2 з position:static</div>
  </body>
</html>
```

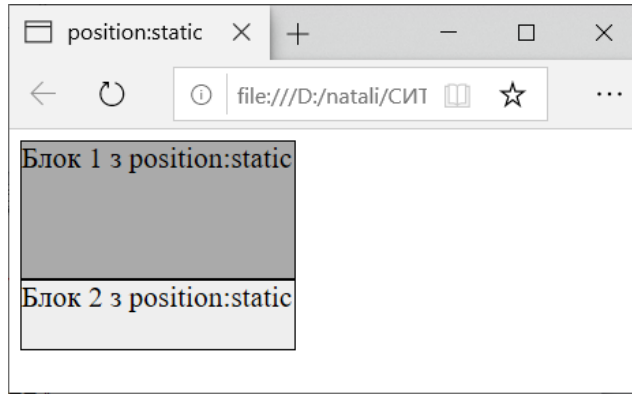

Рис. 1.15. Статичне позиціювання елементів

1.14.4. *Відносне позиціювання*

Установка відносного позиціювання елементу здійснюється за допомогою задання йому властивості `position: relative`.

Елемент з відносним позиціюванням поводиться як елемент у потоці за винятком того, що його поточне положення можна змінити за допомогою властивостей `left`, `top`, `right` і `bottom`.

Нижче подано приклад відносного позиціювання елементів, а його відображення у браузері – на рис. 1.16.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>position:relative</title>
  <style>
   div {
     font-size: 16pt;
     width: 200px;
     border: 1px solid black;
     }
   .bl1 {
```

```
      height: 100px;
      background: #aaa
    }
    .bl2 {
      position: relative;
      top: -20px;
      left: 20px;
      height: 50px;
      background: #eee
    }
  </style>
 </head>
 <body>
 <div class="bl1">Блок 1 з position:static</div>
 <div class="bl2">Блок 2 з position:relative</div>
  </body>
</html>
```

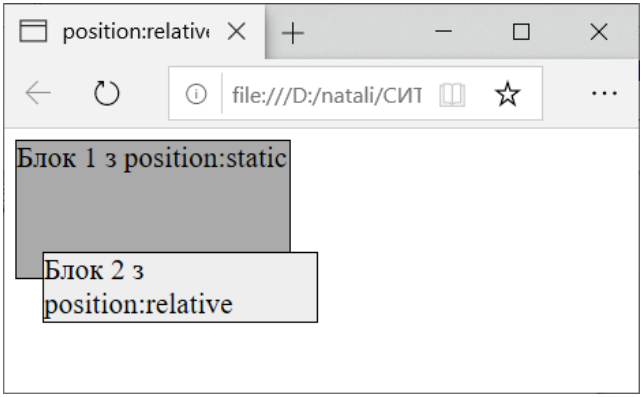

Рис. 1.16. Відносне позиціювання елементів

1.14.5. *Абсолютне позиціювання*

Установка абсолютного позиціонування елементу здійснюється шляхом задавання йому властивості `position` зі значенням `absolute`. При цьому позиціонування виконується відносно найближчого позиціонованого батьківського елемента. Під позиціонованим елементом розуміється елемент з властивістю `position`, що дорівнює `relative`, `absolute` або `fixed`.

Розглянемо фрагмент html-кода.

```
<div id="b1" style="position: absolute">
  <div id="b2" style="position: relative">
    <div id="b3" style="position: absolute">
      ...
    </div>
  </div>
</div>
```

У цьому прикладі позиціонування елемента `b3` буде виконувати відносно `b2`, тому що він є позиціонованим і по відношенню до `b3` є ближчим предком. Якщо елемент `b2` не був би позиціонованим, то позиціонування елемента `b3` виконувалося б відносно `b1`.

Якщо серед предків в елемента з `position: absolute` нема позиціонованого елемента, то у цьому випадку він буде позиціонуватися відносно вебсторінки, тобто тегу `<body>`.

Коли елементу задано абсолютне позиціонування без вказівки властивостей, що визначають його положення (`top`, `left`, `right` і `bottom`), він буде перебувати у тому місці, в якому він був би розташований, якби перебував у потоці (при цьому при обчисленнях його положення враховуються тільки елементи, що розташовані до нього у коді і знаходяться у потоці). Всі інші елементи вебсторінки його бачити не будуть, і, отже, вони будуть розташовані на сторінці, а не звертаючи на нього уваги.

CSS-властивості для управління положенням абсолютно позиціонованого елемента працюють по-іншому ніж при відносному позиціюванні. У цьому випадку властивості `top`, `bottom`, `left` і `right` задають положення елемента відносно найближчого позиціонованого батьківського елемента або `body`, якщо такого предку немає.

Встановити ширину (або висоту) абсолютно позиціонувати можна за допомогою установки йому двох координат `top` і `bottom` (або `left` і `right`).

Якщо елементу одночасно встановити властивості `top`, `bottom` і `height`, то пріоритет буде мати top і `height`.

Абсолютне позиціонування застосовується дуже часто спільно з відносним позиціонуванням у дизайнерських цілях, коли необхідно розмістити різні елементи відносного один одного, так само може застосовуватися для створення меню, розмітки сайту тощо.

Нижче подано приклад абсолютного позиціювання елементів, а його відображення у браузері – на рис. 1.17.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>position:absolute</title>
  <style>
   div {
     font-size: 16pt;
     width: 200px;
     border: 1px solid black;
     }
   .bl1 {
      height: 100px;
      background: #aaa
   }
   .bl2 {
      position: absolute;
      top: 50px;
      left: 50px;
      height: 50px;
      background: #eee
   }
  </style>
 </head>
 <body>
  <div class="bl1">Блок 1 з position:static</div>
  <div class="bl2">Блок 2 з position:absolute </div>
  </body>
</html>
```

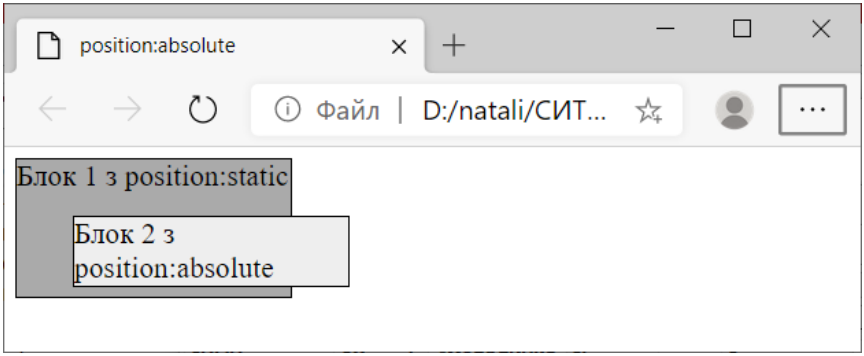

Рис. 1.17. Абсолютне позиціювання елементів

1.14.6. *Фіксоване позиціювання*

Задання елементу фіксованого позиціонування здійснюється за допомогою установки йому властивості `position` зі значенням `fixed`.

Фіксоване позиціювання подібне до абсолютного, але на відміну від нього елемент завжди прив'язується до країв вікна браузера, і залишається у такому положенні навіть при прокручуванні сторінки.

Фіксоване позиціювання застосовується для закріплення на вебсторінці навігаційних меню, кнопки «вгору», панелей з кнопками соціальними мереж тощо.

Нижче подано приклад фіксованого позиціювання елементів, а його відображення у браузері – на рис. 1.18.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>position:fixed</title>
  <style>
   div {
     font-size: 16pt;
     width: 200px;
     border: 1px solid black;
     }
   .bl1 {
       background: #aaa
   }
   .bl2 {
      position: fixed;
      top: 50px;
      left: 50px;
      height: 50px;
      background: #eee
   }
  </style>
 </head>
 <body>
   <div class="bl1">Фіксоване позиціювання подібне
до абсолютного, але на відміну від нього елемент
завжди прив'язується до країв вікна браузера, і
залишається у такому положенні навіть при
прокручуванні сторінки.</div>
   <div class="bl2">Блок 2 з position:fixed</div>
```

```
  </body>
</html>
```

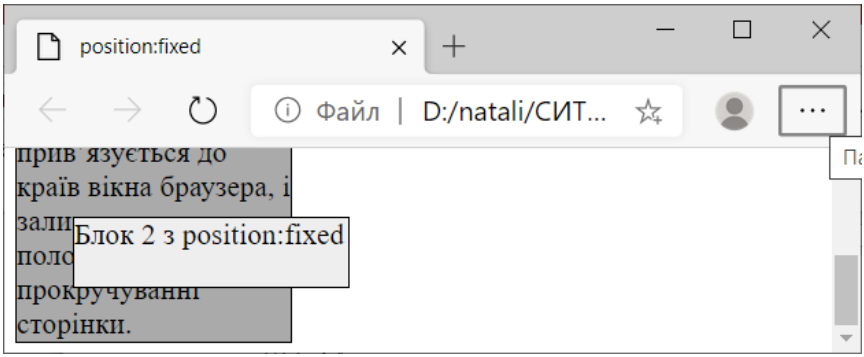

Рис. 1.18. Фіксоване позиціювання елементів

1.14.7. *Спільне використання різних типів позиціювання*

При професійній верстці сайтів часто комбінують різні типи позиціювання елементів. Зокрема, відносне позиціонування дуже часто використовується разом з абсолютним або фіксованим позиціонуванням. У цьому випадку необхідно розуміти правила, що задають положення елементів.

Розглянемо приклади.

Якщо розташувати блоки з абсолютним позиціонуванням у блок, для якого задано відносне позиціювання, то відстані будуть вже розраховуватися не від краю вікна браузера, а від меж відносного блоку.

Нижче подано фрагмент html-коду та його схематичне зображення у браузері на рис. 1.19.

```
<body>
<!-- Зелений блок з відносним позиціюванням -->
<div style="width: 400px; height:400px;
position:relative; top:50px; left:50px; border: 1px
solid black; background:green;">
<!-- Червоний блок з абсолютним позиціюванням -->
  <div style="width: 100px; height:100px;
position:absolute; top:50px; left:50px; border: 1px
solid black; background:red;"></div>
<!-- Синій блок с з абсолютним позиціюванням -->
```

```
  <div style="width: 100px; height:100px;
position:absolute; bottom:50px; right:50px; border:
1px solid black; background:blue;"></div>
</div>
</body>
```

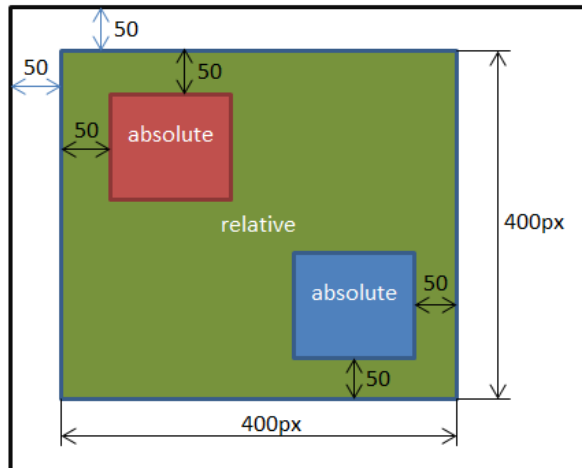

Рис. 1.19. Приклад 1 спільного використання відносного і абсолютного позиціювання елементів

Створення фіксованих макетів, що складаються з трьох блоків, вирівняних по верхньому краю. Для наочності встановимо висоту `400px` блоку з відносним позиціюванням. Фрагмент html-коду та його схематичне зображення у браузері на рис. 1.20 подано нижче.

```
<body>
<div style="width: 800px; height:400px;
position:relative; border: 1px solid black;
background:green;">
  <div style="width: 200px; height:200px;
position:absolute; left:0px; border: 1px solid
black; background:red;">Лівий блок</div>
  <div style="width: 400px; height:200px;
position:absolute; left:200px; border: 1px solid
black; background:blue;">Основний блок</div>
  <div style="width: 200px; height:200px;
position:absolute; left:600px; border: 1px solid
black; background:red;">Правий блок</div>
```

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

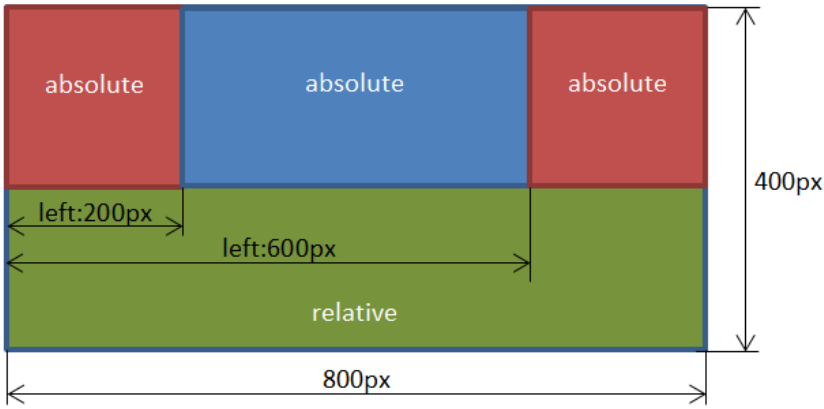

Рис. 1.20. Приклад 2 спільного використання відносного і абсолютного позиціювання елементів

При одночасному використанні декількох блоків з позиціюванням часто також застосовують властивість `z-index`, що призначена для позиціонування елементів по осі Z. Чим більше значення властивості `z-index`, тим ближче елемент розташований, і навпаки, чим менше значення, тим далі розташований елемент. Фрагмент html-коду та його схематичне зображення у браузері на рис. 1.21 подано нижче.

```
<body>
<div style="width: 300px; height:300px;
position:relative; border: 1px solid black;
background:green;">
  <div style="width: 100px; height:100px; position:
absolute; z-index: 1; left: 50px; top: 50px; border:
1px solid black; background: red;"></div>
  <div style="width: 100px; height: 100px; position:
absolute; z-index: 2; left: 100px; top: 100px;
border: 1px solid black; background: blue;"></div>
  <div style="width: 100px; height: 100px; position:
absolute; z-index: 3; left: 150px; top: 150px;
border: 1px solid black; background: yellow;"></div>
</div>
</body>
```

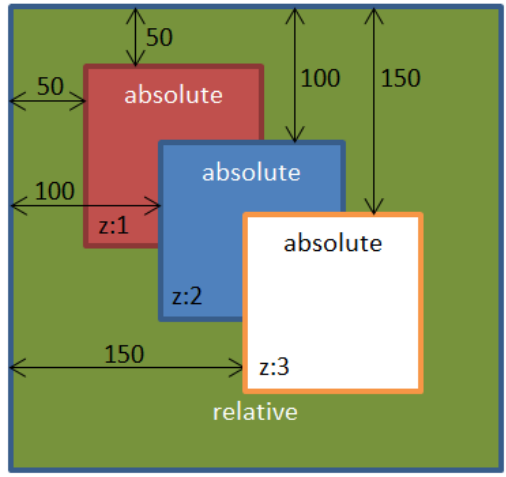

Рис. 1.21. Приклад 3 спільного використання відносного і абсолютного позиціювання елементів та властивості `z-index`

1.15. Спеціальні CSS селектори

Спеціальний селектор – це селектор, що неявно прив'язує стиль до елемента вебсторінки відповідно до складнішого критерія, ніж ім'я тегу або класу. Таким критерієм може бути, наприклад, порядковий номер елемента у батьківському елементі, вказівка на певну частину вмісту абзацу (перший рядок або першу букву), стан гіперпосилання (переглянуте або ні) та ін. Зазвичай спеціальні селектори використовують у комбінованих стилях, щоб зробити їх більш конкретними.

Існує кілька різновидів спеціальних селекторів, які будуть розглянуті нижче.

1.15.1. *Комбінатори*

Комбінатори – різновид спеціальних селекторів, що прив'язує стиль до елементу вебсторінки на підставі його місця розташування відносно інших елементів. Серед них виділяють:

- селектори нащадків (комбінатор <пробіл>), що застосовують стилі до елементів, розташованим всередині елемента-контейнера;
- дочірній селектор (комбінатор >);
- сестринські селектори (комбінатори + і ~ (тильда)).

Комбінатор <пробіл> дозволяє прив'язати стиль до елементу вебсторінки, вкладеному в інший елемент, причому не обов'язково безпосередньо:

```
<елемент 1> <елемент 2> {<стиль>}
```

Стиль буде прив'язаний до елементу 2, який так чи інакше вкладений в елемент 1. При цьому елемент 2 може бути вкладений в інший елемент, вкладений в елемент 1, або навіть у кілька таких елементів послідовно. Тут і далі як елементи 1 і 2 можуть виступати будь-які прості селектори CSS.

Дочірній елемент є прямим нащадком елемента, що його містить. У одного елемента може бути кілька дочірніх елементів, а батьківський елемент у кожного елемента може бути тільки один. Дочірній селектор дозволяє застосувати стилі тільки якщо дочірній елемент йде відразу за батьківським елементом і між ними немає інших елементів, тобто дочірній елемент більше ні у що не вкладено.

Комбінатор > дозволяє прив'язати стиль до елементу вебсторінки, безпосередньо вкладеному в інший елемент:

```
<елемент 1>> <елемент 2> {<стиль>}
```

Нижче подано код, що ілюструє відмінності між селекторами нащадків і дочірніми селекторами, а його вигляд у браузері – на рис. 1.22.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Селектори нащадків і дочірні</title>
  <style>
   body {font-size: 16pt}
   h1 {font-size: 20pt}
   div a {font-style: italic}
   blockquote > p {font-style: italic}
  </style>
 </head>
 <body>
   <h1>Селектори нащадків</h1>
   <a href=''>Звичайне посилання</a>
   <div>
     <p>Звичайний абзац.</p>
     <a href=''>Посилання курсивне</a>
     <p>Абзац з  <a href=''>посиланням</a>.</p>
```

```
    </div>
    <h1>Дочірні селектори</h1>
    <blockquote>
       <p>Цей абзац буде курсивом всередині
цитати.</p>
       <div>
         <p>Звичайний абзац всередині цитати.</p>
       </div>
     </blockquote>
  </body>
</html>
```

Сестринські відносини виникають між елементами, що мають загального батька. Селектори сестринських елементів дозволяють вибрати елементи з групи елементів одного рівня.

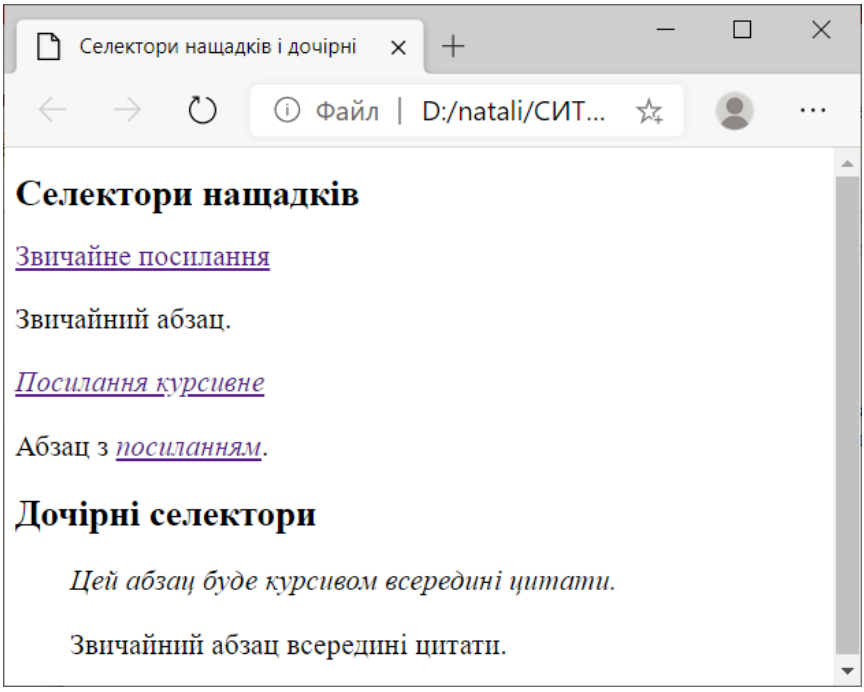

Рис. 1.22. Відмінності між селекторами нащадків та дочірніми селекторами

Комбінатор + дозволяє прив'язати стиль до елементу вебсторінки, безпосередньо наступного за іншим елементом. При цьому обидва дочірніх елемента повинні бути вкладеними в один і той же батьківський:

```
<елемент 1> + <елемент 2> {<стиль>}
```

Стиль буде прив'язаний до *першого* елементу 2, який повинен йти безпосередньо за елементом 1.

Комбінатор ~ (тильда) дозволяє прив'язати стиль до елемента вебсторінки, що йде за іншим елементом і, можливо, є відокремленим від нього іншими елементами. При цьому обидва дочірніх елемента повинні бути вкладеними в один і той же батьківський:

```
<елемент 1> ~ <елемент 2> {<стиль>}
```

Стиль буде прив'язаний до *всіх* елементів 2, які йдуть за елементом 1. При цьому елементи 2 можуть бути відокремлені від елемента 1 іншими елементами.

Нижче подано код, що ілюструє відмінності між сестринськими селекторами, а його вигляд у браузері – на рис. 1.23.

```
<html>
 <head>
  <meta charset="utf-8">
  <title>Сестринські селектори</title>
  <style>
   h1 { font-size: 24pt}
   h2 { font-size: 20pt}
   p, div  { font-size: 18pt}
   h1 + p {font-size: 14pt}
   h2 ~ p {font-size: 14pt}
  </style>
 </head>
 <body>
   <h1>Заголовок рівня 1</h1>
   <p>Абзац шрифтом меншого розміру (14pt).</p>
   <p>Звичайний абзац (шрифт розміром 18pt).</p>
   <h2>Заголовок рівня 2</h2>
   <div>Звичайний текст (шрифт розміром 18pt).</div>
   <p>Абзац шрифтом меншого розміру (14pt).</p>
   <p>Абзац шрифтом меншого розміру (14pt).</p>
 </body>
</html>
```

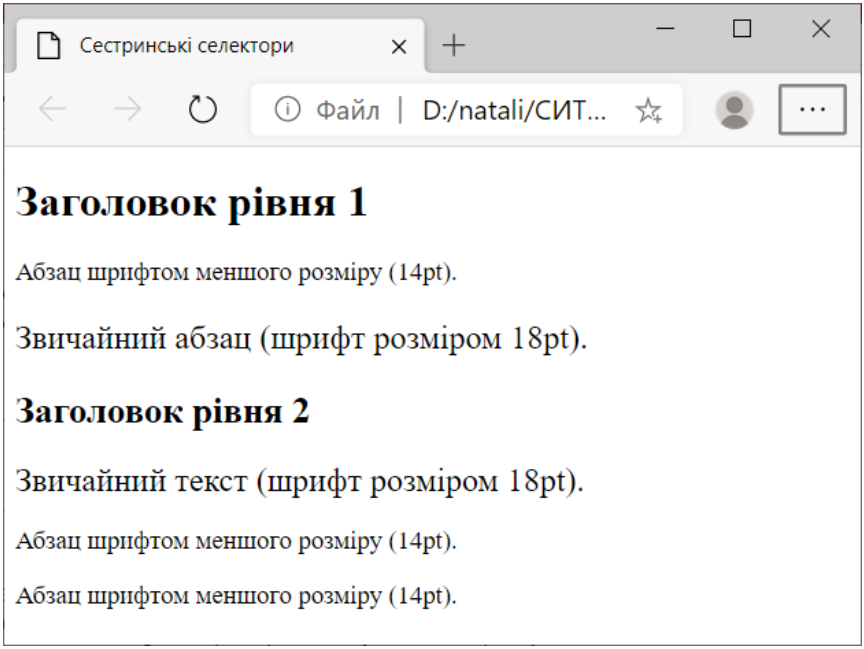

Рис. 1.23. Сестринські селектори

1.15.2. *Селектори по атрибутам тегу*

Селектори по атрибутам тегу – це спеціальні селектори, які прив'язують стиль до тегу на підставі того, чи присутній у ньому певний атрибут або чи має він певне значення.

Селектори по атрибутам тегу використовуються не самі по собі, а тільки у сукупності зі стилями перевизначення тегу або комбінованими стилями. Їх записують безпосередньо після основного селектора без пробілів і беруть у квадратні дужки, причому основний селектор може бути відсутнім.

Селектор виду

```
<основний селектор>[<ім'я атрибута>] {<стиль>}
```

прив'язує стиль до елементів, теги яких мають атрибут з вказаним ім'ям. Наприклад,

```
td[rowspan] {background-color: grey}
```

Цей стиль буде прив’язаний до комірок таблиці, теги яких мають атрибут `rowspan`, тобто до комірок з об’єднанням по горизонталі.

Селектор виду

```
<основний селектор>[<ім'я атрибута>=<значення>]
{<стиль>}
```

прив’язує стиль до елементів, теги яких мають атрибут із зазначеними ім’ям і значенням. Наприклад,

```
td[rowspan=2] {background-color: grey}
```

Даний стиль буде прив’язаний до комірок таблиці, теги яких мають атрибут `rowspan` зі значенням 2, тобто до подвійних комірок з об’єднанням по горизонталі.

Селектори виду

```
<основний селектор>[<ім'я атрибута>~=
<список значень, розділених пробілами>] {<стиль>}
```

прив’язує стиль до елементів, теги яких мають атрибут з вказаним ім’ям і значенням, що збігається з одним із зазначених у списку. Наприклад,

```
td[rowspan~=2 3] {background-color: grey}
```

Цей стиль буде прив’язаний до комірок таблиці, теги яких мають атрибут `rowspan` зі значенням 2 або 3, тобто до подвійних і потрійних комірок з об’єднанням по горизонталі.

Селектори виду

```
<основний селектор>[<ім'я атрибута>|=<значення>]
{<стиль>}
```

прив’язує стиль до елементів, теги яких мають атрибут, що починається з вказаного значення або з фрагмента значення, після якого йде дефіс. Наприклад,

```
div[class|="block"] {background-color: grey}
```

Цей стиль буде прив’язаний до всіх тегів `<div>`, які мають атрибут `class` з назвою, що починається з `block` і містить дефіс.

Селектори виду

```
<основний селектор>[<ім'я атрибута>^=<підрядок>]
{<стиль>}
```

прив’язує стиль до елементів, теги яких мають атрибут з вказаним ім’ям і значенням, що починається з вказаного підрядку. Наприклад,

```
img[src^="/pictures"] {margin: 5px}
```

Цей стиль буде прив’язаний до графічних зображень, теги яких мають атрибут `src` зі значенням, що починається з підрядка "`/pictures`", тобто до зображень, узятих з папки `.pictures`.

Селектор виду

```
<основний селектор>[<ім'я атрибута>$=<підрядок>]
{<стиль>}
```

прив’язує стиль до елементів, теги яких мають атрибут з вказаним ім’ям і значенням, що закінчується зазначений підрядком. Наприклад,

```
img[src$="gif"] {margin: 10px}
```

Даний стиль буде прив’язаний до графічних зображень, теги яких мають атрибут `src` зі значенням, що закінчується підрядком "`gif`", тобто до зображень у форматі GIF.

Селектор виду

```
<основний селектор>[<ім'я атрибута>*=<підрядок>]
{<стиль>}
```

прив’язує стиль до елементів, теги яких мають атрибут з вказаним ім’ям і значенням, що містить вказаний підрядок. Наприклад,

```
a[href*="css"] {background-color: grey}
```

Цей стиль буде прив’язаний до гіперпосилань, теги яких мають атрибут `href` зі значенням, що мають в URL-адресі підрядок "css".

1.15.3. *Псевдокласи*

Псевдокласи – це найпотужніший різновид спеціальних селекторів. Вони дозволяють прив’язати стиль до елементів вебсторінки на основі їх стану, наприклад за наведенням на них курсора миші, або розташування у батьківському елементі.

Псевдокласи були запроваджені для вибору на основі інформації, яка лежить поза деревом документу або яка не може бути виражена за допомогою інших простих селекторів.

Псевдокласи також використовуються не самі по собі, а тільки у сукупності з іншими стилями. Для їх запису використовують основний селектор, далі без пробілів ставиться двокрапка і назва псевдокласу:

```
<основний селектор>:<псевдоклас> {<стиль>}
```

Назва псевдокласу не залежить від регістру. Деякі псевдокласи взаємно виключають один одного, а інші можуть застосовуватися одночасно до того ж елемента. Псевдокласи можуть бути динамічними у тому сенсі, що елемент може придбати або втратити псевдоклас, коли користувач взаємодіє з документом.

Псевдокласи, у свою чергу, самі діляться на групи:

- псевдокласи гіперпосилань та інших елементів;
- псевдокласи елементів форм;
- цільовий псевдоклас;
- структурні псевдокласи;
- псевдоклас, що задає мову;
- псевдокласи `:not` і `*`.

Псевдокласи гіперпосилань призначені для прив'язки стилів до посилань на основі їх стану: є гіперпосилання було відвідане або ні, клацає на ньому відвідувач мишею або тільки навів курсор миші і ін.

Все псевдокласи гіперпосилань, доступні у стандарті CSS 3:

`:link` – невідвідуване гіперпосилання;

`:visited` – відвідане гіперпосилання;

`:active` – гіперпосилання або інший елемент, наприклад кнопка, на якому відвідувач у даний момент клацає мишею;

`:focus` – гіперпосилання або інший елемент, що отримав фокус;

`:hover` – гіперпосилання або інший елемент, на який наведено курсор миші.

Псевдокласи гіперпосилань діють спільно зі стилями пере-визначення тегу `<a>`. У результаті стилі будуть застосовані до всіх гіперпосилань на вебсторінці. Наприклад,

```
a:link {text-decoration: none}
a:visited {text-decoration: overline}
a:active {text-decoration: underline}
```

Також можна застосувати стильове оформлення для всіх гіперпосилань відповідного класу:

```
a.special:link {color: darkred}
```

```
a.special:visited {color: darkviolet}
a.special:active {color: red}
```

Псевдокласи гіперпосилань можна комбінувати, записуючи їх один за одним:

```
A:visited: hover {text-decoration: underline}
```

Цей стиль буде застосований до гіперпосилань, що були відвідані, над якими знаходиться курсор миші.

Псевдокласи елементів форм дозволяють задавати різне стильове оформлення у залежності від типу елементу форми і встановлених атрибутів. Крім псевдокласів `:active`, `:focus` і `:hover`, що були розглянуто раніше і також можуть використовуватися для елементів форм, CSS 3 підтримає наступні псевдокласи:

- `:invalid` – вибирає елементи `<input>`, значення яких є неприпустимим згідно з його типу, зазначеному в атрибуті `type`. Наприклад, поле `<input type="number">` не може містити літери, а поле `<input type="email">` повинно містити правильну адресу електронної пошти;
- `:valid` – протилежний `:invalid` і застосовується до полів форми, вміст яких проходить перевірку у браузері на відповідність зазначеного типу;
- `:readonly` – застосовується до полів форми, в яких заданий атрибут `readonly`;
- `:read-write` – протилежний `:readonly` і використовується до полів форми, доступних для зміни;
- `:checked` – застосовується до елементів інтерфейсу, таких як перемикачі (`radio`) і прапорці (`checkbox`), коли вони знаходяться у положенні «включено»;
- `:default` – застосовує стиль до елементів форм, які встановлені за замовчуванням у групі подібних елементів;
- `:disabled` – застосовує стиль до заблокованих елементів форм, тобто тих, в яких вказано атрибут `disabled`;
- `:enabled` – протилежний `:disabled` і використовується для застосування стилю до доступних (незаблокованих) елементів форм;
- `:indeterminate` – задає стиль для елементів форм, таких як прапорці і перемикачі, коли вони знаходяться у невизначеному стані, тобто коли ні один з перемикачів з групи не вибрано;

- `:optional` – застосовує стильові правила до поля форми, в якого не заданий атрибут `required`;
- `:required` – застосовує стильові правила до тегу `<input>`, у якого встановлений атрибут `required`;
- `:in-range` – задає стиль для елементів форм, значення яких знаходяться у заданому діапазоні та задаються за допомогою атрибутів `min` і `max`;
- `:out-of-range` – протилежний `:in-range` і використовується для застосування стилю для елементів форм, значення яких не входять у встановлений діапазон.

Нижче подано приклад використання псевдокласів елементів форм, а його вигляд у браузері – на рис. 1.24.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Псевдокласи форм</title>
  <style>
   body, input {font-size: 16pt}
   input:invalid {
    background: #fdd;
   }
   input:valid {
    background: white;
   }
  </style>
 </head>
 <body>
  <form>
   <p>Адреса сайту</p>
   <p><input type="url" required></p>
   <p>Адреса електроної пошти</p>
   <p><input type="email" required></p>
   <p><input type="submit" value="Надіслати"/></p>
  </form>
 </body>
</html>
```

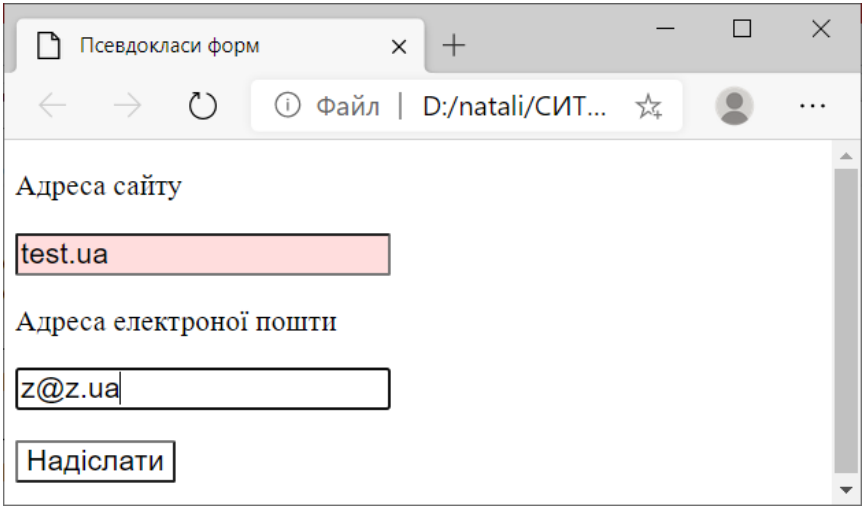

Рис. 1.24. Псевдокласи елементів форм

Цільовий псевдоклас `:target` використовується для переходу по гіперпосиланню до вибраного фрагменту документа. Для цього до адреси документа, на який необхідно перейти, додається символ # і вказується ім'я ідентифікатора. Наприклад,

```
http://example.com/html/top.html#section_2.
```

Такий запис адреси називають *цільовим елементом* або *якорем*.

Псевдоклас `:target` застосовується до цільового елементу.

Нижче подано приклад використання псевдокласу `:target`, а його вигляд у браузері – на рис. 1.25.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>:target</title>
  <style>
  body {font-size:16pt}
   h2:target {
    background: #ddd;
    padding: 3px;
   }
  </style>
```

```
 </head>
 <body>
  <ul>
    <li><a href="#h1">Вступ</a></li>
    <li><a href="#h2">Основна частина</a></li>
  </ul>
  <h2 id="h1">Вступ</h2>
   <p>Вступна частина документа.</p>
  <h2 id="h2">Основна частина</h2>
   <p>В основній частині розкривається вміст
документа.</p>
 </body>
</html>
```

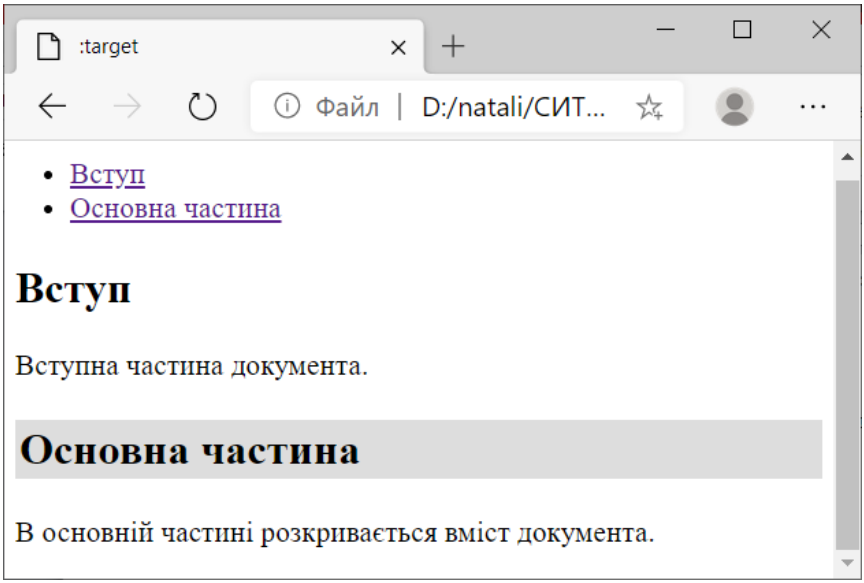

Рис. 1.25. Цільовий псевдоклас :target

Структурні псевдокласи дозволяють відбирати елементи і прив'язати до них стиль на підставі інформації, яка відображена у дереві документа і не може бути отримана за допомогою простих селекторів або їх комбінації. Відлік починається з елемента з індексом 1.

Псевдокласи `:first-child` і `:last-child` прив'язують стиль до елементу вебсторінки, який є, відповідно, першим і останнім дочірнім елементом свого батька, наприклад

```
tr:first-child {font-weight: bold}
tr:last-child {background: #dfd}
```

Ці стилі будуть застосовані до першого і останнього рядків таблиці.

У CSS 3 були додані псевдокласи `:first-of-type` і `:last-of-type`, що призначені до прив'язування стилю до, відповідно, першого і останнього елемента даного типу серед дочірніх елементів батьківського елемента-контейнера та працюють аналогічно до `:first-child` і `:last-child`.

Нижче подано html-код, що ілюструє роботу псевдокласів `:first-of-type`, `:last-of-type`, `:first-child` і `:last-child`, а його вид у браузері – на рис. 1.26.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Псевдокласи first-і last-</title>
  <style>
  body {font-size:16pt}
  table {border-collapse: collapse}
  tr {background: #eee}
  td {border: solid black 1px;
      padding:10px;
      text-align:center}
  tr:first-child {font-weight: bold}
  tr:last-child {background: #dfd}
  p:first-child {font-weight: bold}
  p:first-of-type {background: #dfd}
  </style>
 </head>
 <body>
  <p>Абзац 1</p>
  <div>
  <p>Абзац 2</p>
  <p>Абзац 3</p>
  <table>
    <tr><td>Напівжирний текст1</td>
        <td>Напівжирний текст2</td></tr>
    <tr><td>Звичайний текст1</td>
        <td>Звичайний текст2</td></tr>
    <tr><td>Звичайний текст3</td>
```

```
        <td>Звичайний текст4</td></tr>
      <tr><td>Текст на іншому фоні1</td>
        <td>Текст на іншому фоні2</td></tr>
    </table>
    </div>
     </body>
  </html>
```

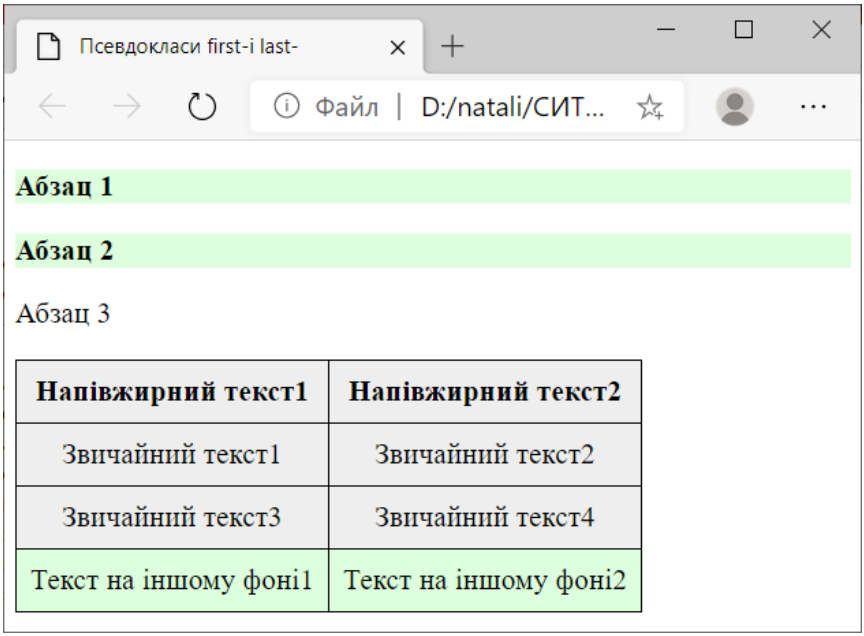

Рис. 1.26. Структурні псевдокласи `:first-of-type`, `:last-of-type`, `:first-child` і `:last-child`

Псевдоклас `:only-child` застосовується до дочірнього елемента вебсторінки, тільки якщо він єдиний у батька.

Псевдоклас `:only-of-type` прив'язує стиль до елемента вебсторінки, який є єдиним дочірнім елементом у свого батьківського елемента, що сформований за допомогою даного тега.

Нижче подано html-код, що ілюструє відмінності між псевдокласами `:only-child` і `:only-of-type`, а його вигляд у браузері – на рис. 1.27. У цьому прикладі псевдоклас `:only-child` буде застосовано до тексту першого абзацу, оскільки у ньому тег `<span>` є єдиним елементом.

Псевдоклас :only-of-type прив'яже стиль до рисунку у третьому абзаці, оскільки він у абзаці єдиний рисунок.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Псевдокласи only</title>
  <style>
   body {font-size:16pt}
   img:only-of-type {
    border: 2px solid red   }
   span:only-child {font-weight: bold}
  </style>
 </head>
 <body>
  <p><span>Напівжирний текст</span></p>
  <p><img src="img/erizo2.jpg">
     <img src="img/erizo2.jpg"></p>
  <p><img src="img/erizo2.jpg"><span>Звичайний
текст</span></p>
 </body>
</html>
```

Псевдоклас :nth-child дозволяє застосувати стиль до елементів вебсторінки, вибравши їх за порядковими номерами, під якими вони визначені у своєму батьківському елементі:

```
<основний селектор>:nth-child(odd| even |<число>|
<вираз>) {<стиль>}
```

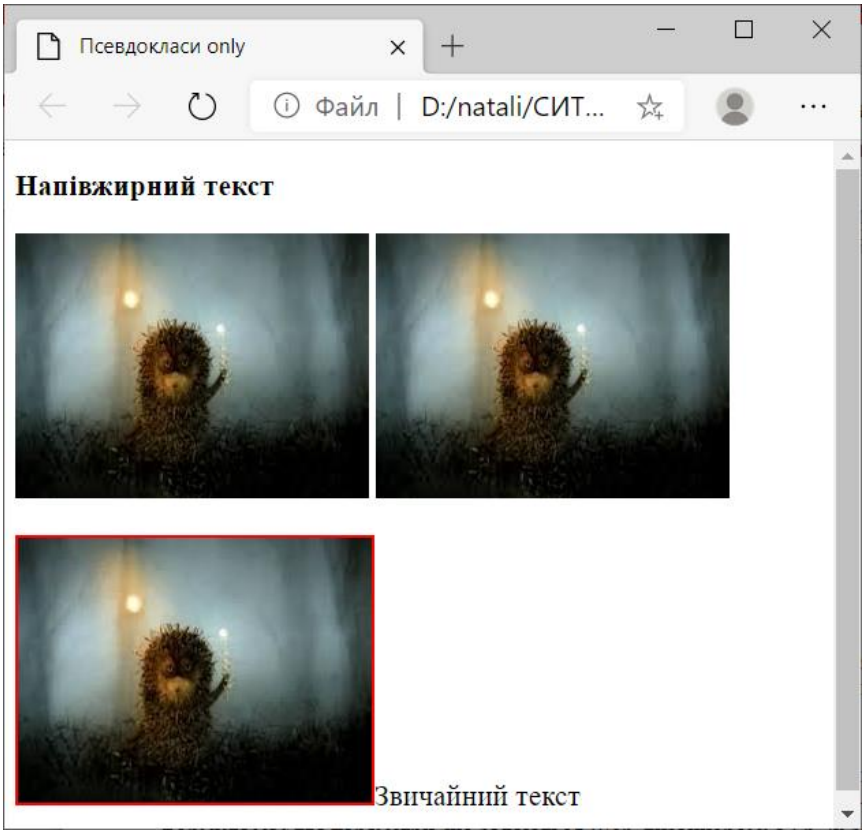

Рис. 1.27. Відмінності між псевдокласами `:only-of-type` і `:only-child`

Можливі значення параметрів псевдокласу `:nth-child`:

- `odd` – всі непарні номери елементів;
- `even` – всі парні номери елементів;
- `число` – порядковий номер дочірнього елемента відносно свого батька, нумерація починається з 1;
- `вираз` – здається у вигляді `an+b`, де `a` і `b` – цілі числа, а `n` – лічильник, який автоматично приймає значення 0, 1, 2 .... Якщо `a` дорівнює нулю, то воно не пишеться і запис скорочується до `b`. Аналогічно, якщо `b` дорівнює нулю, то воно також не вказується і вираз записується у формі `an`. Параметри `a` і `b` можуть бути від'ємними числами, у цьому випадку знак

плюс змінюється на мінус, наприклад, `5n-1`. За рахунок використання від'ємних значень `a` і `b` деякі результати можуть також вийти від'ємними або дорівнювати нулю. Однак на елементи впливають тільки додатні значення через те, що нумерація елементів починається з 1.

Наприклад, запис

```
tr:nth-child(2n+1) {background: #ede}
```

дозволяє застосувати стиль до кожного непарного рядку таблиці. Це еквівалентно запису

```
tr:nth-child(even) {background: #ede}.
```

Псевдоклас `:nth-last-child` аналогічний розглянутому псевдокласу `:nth-child` за тим винятком, що відлік дочірніх елементів ведеться не з початку, а з кінця батьківського елемента. Наприклад,

```
tr:nth-last-child(2) {background: #ede}
```

Даний стиль буде застосований до передостаннього рядка таблиці. Запис

```
#main P:nth-last-child(3) {font-weight: bold}
```

дозволяє вивести напівжирним шрифтом текст третього з кінця абзацу, що знаходиться у контейнері, який має атрибут `id` із значенням `main`.

У стандарті CSS також присутні псевдокласи `:nth-of-type` і `:nth-last-of-type`, що працюють так само, як і псевдокласи `:nth-child` і `:nth-last-child`.

До структурних псевдокласів також належить псевдокласи `:empty`, що дозволяє прив'язати стиль до елементів, які не мають дочірніх елементів, тексту або пробілів, та `:root`, що визначає кореневий елемент документа. У HTML цей селектор завжди відповідає елементу `<html>`.

Псевдоклас `:lang` визначає *мову*, яка використовується у документі або його фрагменті. За допомогою псевдокласу `:lang` можна задавати настройки, характерні для різних мов, наприклад, вид лапок у цитатах:

```
<основний селектор>:lang(<мова>) {<стиль>}
```

Мова задається за допомогою стандартизованих дволітерних позначень. Наприклад, `ua` – українська; `en` – англійська; `de` – німецька; `fr` – французька; `es` – іспанська; `it` – італійська і ін.

Нижче подано приклад, що використовує псевдоклас `:lang` для задання виду лапок у цитатах, а його вид у браузері – на рис. 1.28.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Псевдоклас :lang</title>
  <style>
   body {font-size: 16pt}
   q:lang(de) {
    quotes: "\201E" "\201C";
   }
   q:lang(en), q:lang(es) {
    quotes: "\201C" "\201D";
   }
   q:lang(fr), q:lang(ua) {
    quotes: "\00AB" "\00BB";
   }
  </style>
 </head>
 <body>
    <p>Відомі крилаті фрази різними мовами<p>
    <ul>
    <li>Англійською: <q lang="en">To be or not to
be</q>.</li>
    <li>Французькою: <q lang="fr">A la guerre comme
à la guerre</q>.</li>
    <li>Німецькою: <q lang="de">Anderer Fehler sind
gute Lehrer</q>.</li>
     <li>Іспанською: <q lang="es">Amar es el más
poderoso hechizo para ser amado</q>.</li>
    </ul>
 </body>
</html>
```

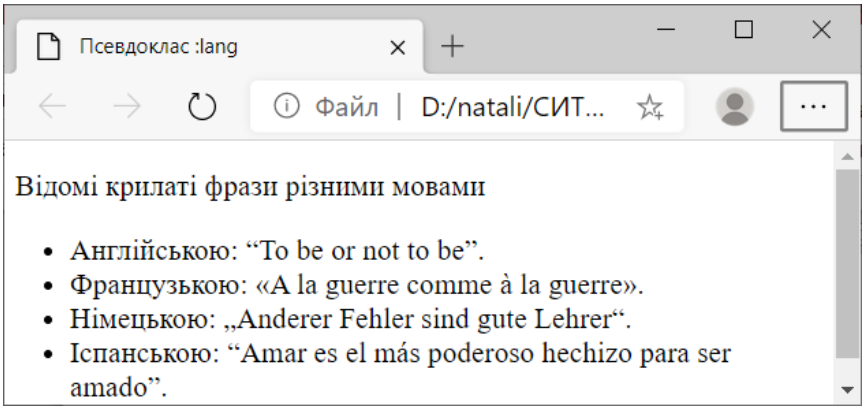

Рис. 1.28. Приклад використання псевдокласу `:lang`

Особливий *псевдоклас* `:not` дозволяє прив’язати стиль до будь-якого елементу вебсторінки, які не задовольняють заданим умовам. Такою умовою може бути будь-який селектор:

```
<Основний селектор>:not (<селектор вибору>)
{<стиль>}
```

У результаты застосування стиль буде прив’язаний до елементу вебсторінки, що задовольняє основному селектору і не задовольняє селектору вибору. Наприклад:

```
div:not (#cmain) {background-color: yellow}
```

Тут в якості основного селектора виступає стиль перевизначення тега `<div>`, а в якості селектора вибору – іменований стиль `cmain`. У результаті даний стиль буде застосований до всіх блокових контейнерів, за винятком `cmain`.

Інший приклад демонструє застосування стилю до всіх рядків таблиці, за винятком першої:

```
tr:not (:nth-child 1)) {background-color: grey}
```

Псевдоклас `*` ("зірочка") позначає будь-який елемент вебсторінки. Наприклад,

```
#cmain *:first-child {border-bottom: medium solid
black}
```

Цей стиль буде застосований до першого елементу будь-якого типу, який безпосередньо вкладений у контейнер `cmain`.

1.15.4. *Псевдоелементи*

Псевдоелементи – різновид спеціальних селекторів, що прив'язують стиль до певного фрагменту елемента вебсторінки, що не представлений у структурі html-документа і тому не належить до його об'єктної моделі. Таким фрагментом може бути, наприклад, перший символ або перший рядок в абзаці.

Особливістю псевдоелементів є їх використання тільки у сукупності з іншими стилями. Згідно синтаксису псевдоелементи записують відразу після основного селектора без пробілів та починають з подвійної двокрапки:

```
<Основний селектор>::<псевдоелемент> {<стиль>}
```

Псевдоелемент `::first-letter` прив'язує стиль до першої букви тексту в елементі вебсторінки, якщо їй не передує вбудований елемент, який не є текстом, наприклад, зображення. Це дозволяє створювати у тексті *буквицю* і *виступаючий ініціал*.

Буквиця являє собою збільшену першу літеру, базова лінія якої нижче на одну або кілька рядків базової лінії основного тексту. *Виступаючий ініціал* – це збільшена прописна буква, базова лінія якої збігається з базовою лінією основного тексту. Наприклад:

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

До псевдоелемента `::first-letter` можуть застосовуватися тільки обмежений набір css-властивостей. До них відносяться властивості, що мають відношення до кольору, фону, меж, шрифтів, а також відступів padding і margin.

Псевдоелемент `::first-line` прив'язує стиль до першого рядка тексту в елементі вебсторінки і також має обмежений набір css-властивостей, аналогічних до `::first-letter`.

```
p::first-line {text-transform: uppercase}
```

Псевдоелементи `::before` і `::after` використовуються разом з властивістю `content` і дозволяють вивести необхідні дані відповідно до та після вмісту елемента. За замовчуванням вони мають властивість відображення `display` зі значенням `inline`. Якщо потрібно встановити

інше відображення, то його необхідно вказати явно (наприклад, `display: block`).

Вміст даних псевдоелементів задається за допомогою властивості `content`. При цьому якщо псевдоелемент буде без вмісту, то цю властивість все одно необхідно вказувати і використовувати в якості його значення порожній рядок, тобто `content: ""`. Без вказівки `content` псевдоелементи відображатися не будуть.

Ще однією особливістю псевдоелементів `::before` і `::after` є те, що вони не наслідують стилі. Тому у разі необхідності у них стиля батьківського елемента стильові властивості потрібно явно записати.

Псевдоелемент `::selection` призначений для встановлення стилів до виділеного користувачем фрагмента тексту. З ним можна використовувати обмежене коло властивостей: `background`, `color`, `cursor` і `outline`.

Нижче подано приклад, що використовує псевдоелементи `::before`, `::after` і `::selection` для оформлення цитат і зміни стиля виділеного тексту, а його вид у браузері – на рис. 1.29.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Псевдоелементи after і before</title>
  <style>
    *, *::before, *::after {
            box-sizing: border-box;
    }
    html {
      font-family: "Helvetica Neue",
    }
    .container {
        padding: 20px;
    }
    .blockquote {
            margin: 0 auto;
            max-width: 400px;
            position: relative;
            padding: 5px 32px;
            background-color: #e0f2f1;
            color: #004d40;
            border-radius: 4px;
```

```
    }
    .blockquote::before {
            content: "\201e";
            position: absolute;
            top: -16px;
            left: 6px;
            font-family: Georgia, serif;
            font-size: 40px;
            line-height: 1;
            font-weight: bold;
    }
    .blockquote::after {
            content: "\201c";
            position: absolute;
            right: 6px;
            font-family: Georgia, serif;
            font-size: 40px;
            line-height: 1;
            font-weight: bold;
    }
    .blockquote::selection {
            background-color: #9c27b0;
            color: #fff;
    }
  </style>
 </head>
<body>
 <div class="container">
 <div class="blockquote">Псевдоелементи ::before і
::after використовуються разом з властивістю content
і дозволяють вивести необхідні дані відповідно до та
після вмісту елемента. За замовчуванням вони мають
властивість відображення display зі значенням
inline. Якщо потрібно встановити інше відображення,
то його необхідно вказати явно (наприклад, display:
block).</div>
 </div>
</body>
</html>
```

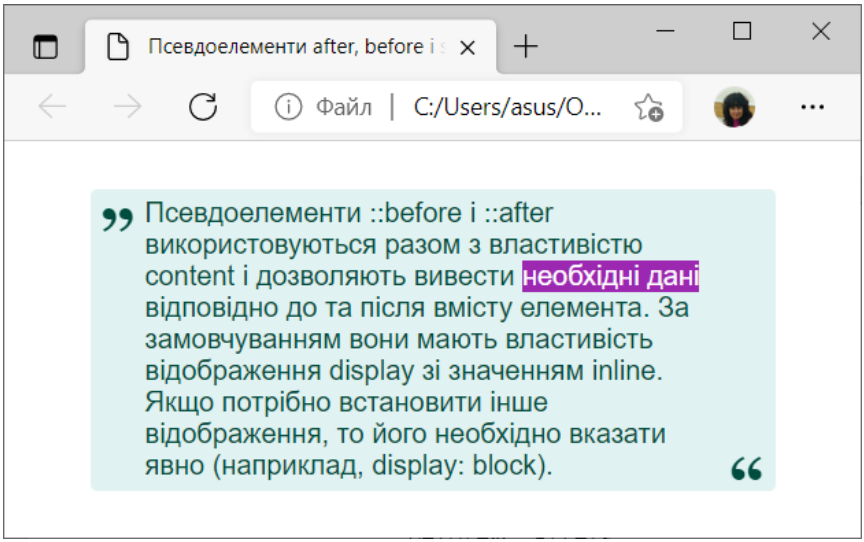

Рис. 1.29. Приклад оформлення цитат з використанням псевдоелементів `::before` і `::after` та зміни стиля фрагмента виділеного тексту

2. ПОСТАНОВКА ЗАВДАННЯ НА ЛАБОРАТОРНІ РОБОТИ

2.1. Форматування сайту за допомогою CSS

Метою лабораторної роботи **є** закріплення знань студентів з технології каскадних таблиць стилів CSS, що використовується для форматування html-документів, створення зовнішньої та внутрішньої таблиць стилів для форматування сайту, а також дослідження правил каскадування та пріоритету стилів.

Порядок виконання роботи:

1. Створити за допомогою редактора Microsoft Visual Studio або Eclipse зовнішню таблицю стилів, яку підключити до всіх сторінок сайту, що розробляється. При цьому зовнішній вигляд сторінок сайту повинен бути ідентичним.

У каскадній таблиці стилів повинні бути визначені:

- правила відображення простих селекторів;
- класи для відображення деяких видів звичайного тексту з відступами і рамками;
- псевдокласи для відображення гіперпосилань та інших html-елементів;
- псевдоелементи для виділення деяких фрагментів html-елементів;
- контекстні (дочірні, сусідні та ін.) селектори для тексту всередині таблиці або блоку та інших html-елементів.

2. Добавити до однієї зі сторінок сайту внутрішню таблицю, в якій перевизначити правила форматування будь-яких п’яти елементів. Дослідити правила каскадування стилів та їх пріоритет.

2.2. Управління видимістю, прозорістю та позиціюванням елементів за допомогою CSS

Метою лабораторної роботи **є** закріплення знань студентів з методів управління видимістю, прозорістю, розміщення html-елементів у потоці, а також дослідження відмінностей між різними типами позиціювання елементів.

Порядок виконання роботи:

1. Розробити html-документ, в якому за допомогою абсолютного позиціонування або плаваючих блоків визначити інформаційну область і область меню. У верхній частині документ повинен містити заголовок з ефектом тіні та логотип сайту, а у нижній частині екрана повинна знаходитися інформація про автора і авторські права.

2. Добавити в html-документ блок з абсолютним позиціюванням, що містить дочірній елемент. Для дочірнього елемента змінити у таблиці стилів тип позиціювання та дослідити зміни у його розміщенні.

3. Оформити виділення ключових слів у тексті розробленого html-документа за допомогою тегу `<SPAN>` і задання відповідного класу у таблиці стилів.

4. Передбачити у розробленому html-документі зміну стильового оформлення при відділенні користувачем фрагменту його тексту.

5. Додати у розроблений html-документ цитату, для зображення якої використати псевдоелементи.

6. Додати у розроблений html-документ рисунки з ефектами напівпрозорості та тіні, що частково перекривають один одного.

ТЕСТОВІ ЗАВДАННЯ ДЛЯ ПЕРЕВІРКИ ЗНАНЬ

1. Як розшифровується абревіатура CSS?
   а) Cascading Style Sheets;
   б) Computer Style Sheets;
   в) Creative Style Sheets;
   г) Canvas Styling System;
   д) Complex Style Sheets.
2. Хто створює вебстандарти?
   а) Mozilla;
   б) The World Wide Web Consortium;
   в) Microsoft;
   г) Google;
   д) жоден із перелічених вище.
3. Яким чином можна додати CSS до html-документа?
   а) за допомогою вбудованих стилів (Inline Styling);
   б) за допомогою внутрішньої таблиці стилів (Internal Style Sheet);
   в) за допомогою зовнішню таблиці стилів (External Style Sheet);
   г) за допомогою вбудованої таблиці стилів (Inline Style Sheet);
   д) усіма чотирма способами.
4. Який HTML-тег використовується для встановлення внутрішньої таблиці стилів (Internal style sheet)?
   а) `<css>`
   б) `<script>`
   в) `<style>`
   г) `<head>`
   д) `<link>`
5. Де в HTML-документі слід вставляти посилання на зовнішню таблицю стилів?
   а) `<head>`
   б) `<body>`
   в) `<footer>`
   г) `<aside>`
   д) Можна у будь-якому місці html-документа
6. Який атрибут HTML використовується для встановлення вбудованого стилю?

а) `css`
б) `style`
в) `in-style`
г) `font`
д) `class`

7. Як підключити зовнішню таблицю стилів (External style sheet), що зберігається у файлі `mystyle.css`, до html-документа?
   а) `<link rel="stylesheet" type="text/css" href="mystyle.css">`
   б) `<style src="mystyle.css" />`
   в) `<stylesheet>mystyle.css</stylesheet>`
   г) `<stylesheet src="mystyle.css" />`
   д) `<link rel="stylesheets" type="text/css" src="mystyle.css">`
8. Які твердження вірно характеризують правила пріоритету стилів?
   а) зовнішня таблиця стилів, посилання на яку зустрічається в html-коді сторінки раніше, має пріоритет перед тією, посилання на яку зустрілося пізніше;
   б) внутрішня таблиця стилів має пріоритет перед зовнішньою;
   в) вбудовані стилі мають пріоритет перед будь-якими стилями, заданими у таблицях стилів;
   г) більш конкретні стилі мають пріоритет перед менш конкретними;
   д) якщо до тегу прив'язані кілька стильових класів, то ті, що вказані лівіше, мають пріоритет перед зазначеними правіше.
9. Як вставити коментар у файл CSS?
   а) `<!-- це коментар -->`
   б) `// це коментар`
   в) `' це коментар`
   г) `/* це коментар */`
   д) `{ це коментар }`
10. Як додати колір фону для всіх елементів `<h2>`?
    а) `all.h2 {background-color:#FFFFFF;}`
    б) `h2 {background-color:#FFFFFF;}`
    в) `h2.all {background-color:#FFFFFF;}`
    г) `#h2 {background-color:#FFFFFF;}`
    д) `.h2 {background-color:#FFFFFF;}`

11. Виберіть правильний CSS синтаксис, що змінює шрифт абзаців тексту на напівжирний.
    а) `p style="text-size:bold"`
    б) `p {text-size:bold}`
    в) `p {text-weight:bold}`
    г) `p style="font-size:bold"`
    д) `p {font-weight: bold}`
12. Яка властивість використовується для зміни розміру межі html-елемента?
    а) `border-size:3px;`
    б) `border:width:3px;`
    в) `border-width:3px;`
    г) `border-width-size:3px;`
    д) `border-size-width:3px;`
13. Яка CSS властивість контролює розмір тексту?
    а) `font-size`
    б) `text-size`
    в) `css-size`
    г) `text-height`
    д) `text-style`
14. Який селектор використовується для встановлення стилю кількох елементів?
    а) `id`
    б) `class`
    в) `style`
    г) `text`
    д) `css`
15. Виберіть правильний CSS синтаксис
    а) `a {font-weight: bold;}`
    б) `{a:font-weight: bold;}`
    в) `a {font:weight: bold;}`
    г) `a:font:weight=bold;`
    д) `a:font-weight: {bold;}`
16. Як створити список без маркерів?
    а) `list-style-type: no-bullet`
    б) `list-style-type: none`
    в) `list: none`

г) `list-style-type: nobullet`
д) це засобами CSS зробити неможливо.

17. .Як створити список, в якому маркери елементів представлені у вигляді однотонних квадратів?
    а) `list-type: square;`
    б) `list-style-type: square;`
    в) `list-type-style: square;`
    г) `list: square;`
    д) це засобами CSS зробити неможливо.
18. Виберіть правильний CSS синтаксис, який визначає розмір шрифту всіх тегів `<a>` у 14pt
    а) `a{font-size:14pt;}`
    б) `a{font:14pt;}`
    в) `a{size: 14pt;}`
    г) `a{text:14pt;}`
    д) `a{text-size:14pt;}`
19. Як прибрати підкреслення гіперпосилання?
    а) `a {decoration:no-underline;}`
    б) `a {underline:none;}`
    в) `a {underline:no-underline;}`
    г) `a {text-decoration:no-underline;}`
    д) `a {text-decoration:none;}`
20. Яка властивість дозволяє вивести кожне слово у тексті з великої літери?
    а) `text-transform:capitalize;`
    б) `capitalize: word;`
    в) `text-style:capitalize;`
    г) `transform:capitalize;`
    д) це засобами CSS зробити неможливо.
21. Яка властивість CSS використовується для зміни кольору тексту елемента?
    а) `text-color;`
    б) `fgcolor;`
    в) `color;`
    г) `font-color;`
    д) `color-text.`
22. У чому полягає помилка у наступному коді?:

```
font-family: Arial, Times New Roman, Helvetica,
sans-serif;
```

а) відсутні лапки для шрифту Times New Roman;
б) не можна вказувати більше трьох різних шрифтів;
в) замість властивості `font-family` необхідно використовувати властивість `font`;
г) не існує шрифту Helvetica.
д) помилка відсутня.

23. Є такий CSS-код:

```
body {font-size: 14pt;} p {font-size: 2em;}.
```

Який розмір тексту буде в абзаці?
а) 7pt;
б) 12pt;
в) 14pt;
г) 16pt;
д) 28pt.

24. Яке з цих значень НЕ може бути значенням для властивості `font-size`?
а) `50%`;
б) `2mm`;
в) `2ex`;
г) `2`;
д) `xx-large`.

25. Як називається функціональна нотація (функція), що дозволяє у CSS виконувати математичні вирази?
а) `calc()`;
б) `calculate()`;
в) `exec()`;
г) execute();
д) `math()`.

26. CSS-властивість `background-image` може містити декілька зображень?
а) ні;
б) так;
в) так, якщо попередньо встановлено директиву `@background: multiple`;

г) так, якщо попередньо встановлено директиву @image: multiple;
д) так, якщо попередньо встановлено директиву @background: many.

27. Який рядок задає зображення для фону та повторює його по вертикалі?
    а) background: url(images/back.png) repeat-y;
    б) background: url(images/back.png) repeat-x;
    в) background-image: url(images/back.png) repeat-y;
    г) background: images/back.png; repeat-y;
    д) background: images/back.png; y-repeat.
28. Яка властивість потрібно додати у тег <img>, щоб отримати тінь усередині картинки?
    а) box-shadow: 0 0 5px #000;
    б) img-shadow: inset 0 0 5px #000;
    в) img-shadow: inset 5px #000;
    г) box-shadow: inset 0 0 5px #000;
    д) shadow: inset 0 0 5px #000.
29. Яке значення НЕ може мати властивість border-style?
    а) dotted;
    б) color;
    в) inset;
    г) outset;
    д) groove.
30. Як правильно групувати селектори?
    а) розділити кожен селектор пробілом;
    б) розділити кожен селектор комою;
    в) розділити кожен селектор точкою з комою;
    г) розділити кожен селектор знаком +;
    д) розділити кожен селектор знаком /.
31. Який елемент у таблиці стилів відповідає html-елементу із id="block"?
    а) *block
    б) block
    в) #block
    г) .block
    д) ~block

32. Є наступний html-код:

```
<div>Text 1</div>
<div id="block">Text 2</div>
<p class="test-1">Text 3</p>
<p id="block"></p>
<p class="test-1">Text 4</p>
```

У чому допущено помилку?

а) помилка відсутня;
б) неправильно записана назва класу `test-1`;
в) клас `test-1` не можна використовувати для кількох тегів;
г) не можна використовувати `id` для блокових елементів;
д) ідентифікатор `block` був присвоєний різним елементам, але його повторення не допускається.

33. Який колір буде мати напівжирний курсивний текст у коді: `<p>Колір цього <b><i>тексту</i></b></p>` при використанні наступного стилю?

```
P {color: green;}
B {color: blue;}
I {color: orange;}
B > I {color: olive;}
P > I {color: yellow;}
```

а) помаранчевий;
б) жовтий;
в) синій;
г) оливковий;
д) зелений.

34. Який стиль потрібно використати, щоб змінити колір тексту лише для другого абзацу?

```
<p class="text text1-count1-text">Перший абзац</p>
<p class="text text2-count2-text">Другий абзац</p>
<p class="text text3-count3-text">Третій абзац</p>
```

а) `P[class^="text2"] { color: red;}`
б) `P[class*="text2"] { color: red;}`
в) `P[class|="text2"] { color: red;}`
г) `P[class~="text2"] { color: red;}`
д) `P[class$="text2"] { color: red;}`

35. До якого елемента буде застосовуватись наступний стиль?

```
[class~="lorem"] {background: #333;}
```

а) `<p class="ipsum-lorem">Lorem ipsum dolor sit amet</p>`

б) `<div class="lorem-ipsum dolor">Lorem ipsum dolor sit amet</div>`

в) `<p class="lorem-ipsum">Lorem ipsum dolor sit amet</p>`

г) `<p><span class="2orem ipsum">Lorem ipsum dolor sit amet</span></p>`

д) `<div class="lorem ipsum">Lorem ipsum dolor sit amet</div>`

36. Який стиль задає колір фону для текстового поля?

а) `INPUT[type="textarea"] {background: #acdacc;}`

б) `INPUT[type="texts"] {background: #acdacc;}`

в) `INPUT[type="textfield"] {background: #acdacc;}`

г) `INPUT[type="textinput"] {background: #acdacc;}`

д) `INPUT[type="text"] {background: #acdacc;}`

37. Як вибрати всі елементи `<p>` всередині елемента `<div>`?

а) `div > p;`

б) `div.p;`

в) `div p;`

г) `div + p;`

д) `div~p.`

38. Який стиль встановить червоний колір тексту в абзаці?

а) `BODY * P { color: red;}`

б) `HTML * P { color: red;}`

в) `HTML P { color: red;}`

г) `BODY P * {color: red;}`

д) `P * {color: red;}`

39. Який буде колір у слова `word`, якщо у таблиці стилів задано `ul li em {color: red;}` та є наступний html-код?

```
<ul>
<li>blah</li>
</ul>
```

а) фіолетовий;

б) колір за замовчуванням;
в) червоний;
г) чорний;
д) білий.

40. Як встановити окремий стиль для посилань на домени української частини Інтернет?
    а) `a[href$=".ua"] { /* style */ };`
    б) `a.ua { /* style */ };`
    в) `if (href="*.ua"} a { /* style */ };`
    г) `.ua { /* style */ };`
    д) `a[href^=".ua"] { /* style */ }.`
41. Як за допомогою CSS можна створити таку рамку навколо елемента?

```
Верхня границя = 1px
Нижня границя = 2px
Ліва границя = 3px
Права границя = 4px
```

    а) `border-width:1px 3px 2px 4px;`
    б) `border-width:2px 3px 1px 4px;`
    в) `border-width:1px 2px 3px 4px;`
    г) `border-width:1px 4px 2px 3px;`
    д) `border-width:4px 1px 3px 2px.`
42. Як можна змінити правий зовнішній відступ елемента?
    а) `indent`;
    б) `box-right`;
    в) `border-right`;
    г) `padding-right`;
    д) `margin-right`.
43. Як буде оформлені слова `link 1`, `link 2`, `link 3`, якщо у таблиці стилів та html-документі задані відповідні коди?

```
li {
display: inline;
margin: 0px 25px 0px 25px;
padding: 10px 50px 10px 50px;
border: thin solid #000000;
}

<ul>
```

```
<li>link 1</li>
<li>link 2</li>
<li>link 3</li>
</ul>
```

а) на одному рядку у тонких чорних рамках та розміром по ширині 50 px та висоті 10 px;
б) на одному рядку у тонких чорних рамках та розміром по ширині 75 px та висоті 10 px;
в) на одному рядку у тонких чорних рамках та розміром по ширині 100 px та висоті 20 px;
г) на трьох рядках у тонких чорних рамках та розміром по ширині 50 px та висоті 25 px;
д) на трьох рядках у тонких чорних рамках та розміром по ширині 100 px, висоті 50 px.

44. Як визначається значення `padding-top: 10%`?
    а) 10 % висоти вмісту блоку;
    б) 10 % ширини вмісту блоку;
    в) 10 % ширини вмісту батьківського елемента;
    г) 10 % від висоти блоку;
    д) 10 % від ширини блоку.
45. Що таке схлопування `margin`?
    а) об'єднання двох вертикальних `margin` в один;
    б) обнуління `margin`;
    в) присвоєння `margin` від'ємного значення;
    г) об'єднання двох горизонтальних `margin` в один;
    д) коли `margin` не враховується у розмірі блоку.
46. Для яких елементів працює схлопування `margin`?
    а) для рядково-блочних;
    б) для рядкових;
    в) для блокових;
    г) для елементів з обтіканням;
    д) для будь-яких.
47. Для яких сусідніх елементів не працює схлопування `margin`?
    а) для елементів із відносним позиціонуванням;
    б) для елементів з `padding`;
    в) для абсолютно позиціонованих елементів;
    г) для рядково-блочних елементів;
    д) для елементів з обтіканням.

48. Яка помилка міститься у наступному коді?

```
div {
 background: #fc0;
 padding: 10px;
 padding-left: -10px;
}
```

а) помилка відсутня;
б) не можна одночасно використовувати `padding` та `padding-left`;
в) `padding` не можна комбінувати із `background`;
г) `padding` може бути від'ємним;
д) `padding-left` повинен знаходитись перед `padding`.

49. Яка ширина блоку буде за наступного CSS?

```
div { width:100px; padding:10px; margin:10px; }
```

а) 100px;
б) 110px;
в) 120px;
г) 130px;
д) 140px.

50. Яке значення `padding` є некоректним?
а) 10px;
б) 10 %;
в) 1em;
г) auto;
д) 0.

51. Яке значення за замовчуванням задається для `width`?
а) 300px;
б) 16em;
в) auto;
г) 0;
д) 100 %.

52. Чому дорівнює висота рядково-блочного елемента за наявності `height`?
а) `height` плюс `margin`, `border` та `padding`;
б) `height` плюс `border` та `padding`;
в) `height` плюс `border` та `margin`;

г) висота вмісту плюс `border` та `padding`;

д) висота вмісту плюс `margin`, `border` та `padding`.

53. Яка властивість змінює алгоритм обчислення розмірів блоку?

а) `display`;

б) `box-sizing`;

в) `resize`;

г) `writing-mode`;

д) `background`.

54. Які властивості формують блок?

а) `outline`;

б) `padding`;

в) `margin`;

г) `border`;

д) `background`;

е) `font-size`.

55. Яка властивість зрушить елемент праворуч?

а) `left: 10px`;

б) `bottom: 10px`;

в) `right: 10px`;

г) `top: 10px`;

д) `left: -10px`.

56. Яка властивість зрушить елемент нагору?

а) `bottom: 10px`;

б) `top: 10px`;

в) `left: 10px`;

г) `left: -10px`;

д) `right: 10px`.

57. Є наступний html-код:

```
<section>
   <div class="c1" style="float: left">…</div>
   <div class="c2">…</div>
</section>
<p>…</p>
```

До якого елемента слід додати `overflow: hidden`, щоб скасувати дію `float` після `<section>`?

а) `<p>`

б) `<div>`
в) `<div class="c1">`
г) `<div class="c2">`
д) `<section>`

58. Яке значення властивості `overflow` не скасовує обтікання?
    а) `visible`;
    б) `auto`;
    в) `hidden`;
    г) `scroll`;
    д) `all`.
59. Які значення допустимі у властивості `clear`?
    а) `left`;
    б) `none`;
    в) `both`;
    г) `right`;
    д) `all`.
60. До яких елементів застосовується властивість `clear`?
    а) до будь-яких;
    б) до нижченаведених;
    в) до вищенаведених;
    г) до дочірніх;
    д) до батьківського.
61. Яке значення `display` прибирає елемент зі сторінки?
    а) `block`;
    б) `list-item`;
    в) `inline`;
    г) `none`;
    д) `transparent`.
62. Яке значення `border` дозволяє прибрати межу?
    а) `invisible`;
    б) `none`;
    в) `block`;
    г) `transparent`;
    д) `hidden`.
63. Що таке потік html-документа?
    а) орієнтація макету для смартфонів, планшетів та інших пристроїв;
    б) порядок накладання елементів з осі Z;

в) порядок накладання елементів з осі X;
г) природний стан елементів один за одним;
д) спосіб розташування елементів на вебсторінці.

64. Які властивості можуть міняти природний порядок потоку?
    а) `display`;
    б) `z-index`;
    в) `margin`;
    г) `float`;
    д) `position`.
65. Яка властивість виводить елемент із потоку?
    а) `display`;
    б) `clear`;
    в) `z-index`;
    г) `margin`;
    д) `float`.
66. Яке значення властивості `position` задає статичне позиціонування?
    а) `static`;
    б) `absolute`;
    в) `relative`;
    г) `fixed`;
    д) `allstatic`.
67. Яка властивість скасовує дію властивості `float`?
    а) `display`;
    б) `visible`;
    в) `position`;
    г) `z-index`;
    д) `clear`.
68. Яка властивість дозволяє змінювати порядок накладання елементів?
    а) `float`;
    б) `display`;
    в) `visible`;
    г) `z-index`;
    д) `position`.
69. Які значення `position` видаляють елемент із потоку?
    а) `static`;
    б) `relative`;

в) `fixed`;
г) `none`;
д) `absolute`.

70. Яке значення за замовчуванням властивість `position`?
    а) `relative`;
    б) `fixed`;
    в) `absolute`;
    г) `none`;
    д) `static`.
71. Як за допомогою стилів встановити, щоб елементи `<div>` розташовувалися поруч один з одним по горизонталі справа наліво?
    а) `div {float: right;};`
    б) `div {position: left;};`
    в) `div {item-align: right;};`
    г) `div {float: left;};`
    д) `div {text-align: right;}.`
72. Якщо абсолютно позиціонованого елемента не задані властивості `left`, `right`, `top`, `bottom`, куди зміститься елемент?
    а) залишиться на вихідному місці;
    б) у верхній лівий кут батьківського елемента;
    в) у нижній правий кут батьківського елемента;
    г) у верхній лівий кут вікна браузера;
    д) у центр вікна браузера.
73. За якого значення `position` не працює властивість `z-index`?
    а) `absolute`;
    б) `fixed`;
    в) `static`;
    г) `relative`;
    д) властивість `z-index` не залежить від значення `position`.
74. Який стиль написати, щоб елемент займав всю висоту?
    а) `div {position: absolute; left: 0; right: 0;};`
    б) `div {position: absolute; margin: auto;};`
    в) `div {position: absolute; height: 100%;};`
    г) `div {position: absolute; height: auto};`
    д) `div {position: absolute; top: 0; bottom: 0;}.`
75. Від чого буде відраховуватись значення властивості `top` для `<section>` за наступного стилю?

```
.parent {position: relative;}
.parent section {position: absolute; top: 100px;}
```

а) від верхнього краю вихідного положення елемента;
б) від верхнього краю вікна браузера;
в) від центра вікна браузера;
г) від верхнього краю батьківського елемента;
д) від верхнього краю видимої області.

76. Який псевдоклас використовується для опису стилю, який активується при знаходженні курсору у межах елемента, але без натискання на цей елемент?
    а) `:active`;
    б) `:focus`;
    в) `:lang`;
    г) `:optional`;
    д) `:hover`.
77. Який псевдоклас дозволяє вибрати останній елемент у групі?
    а) `:last-line`;
    б) `:last-letter`;
    в) `:last-child`;
    г) `:last-item`;
    д) `:last-type`.
78. Код `ul li::first-letter {font-size: 200%;}`
    а) нічого не робить, тому що у коді помилка;
    б) робить першу літеру першого елемента у нумерованому списку розміром 200%;
    в) робить першу літеру першого елемента у ненумерованому списку розміром 200%.
    г) робить першу літеру кожного елемента нумерованого списку розміром 200%;
    д) робить першу літеру кожного елемента ненумерованого списку розміром 200%.
79. Яка властивість є обов'язковою для роботи псевдоелемента `::after`?
    а) `position`;
    б) `text-align`;
    в) `content`;
    г) `width`;
    д) `overflow`.

СПИСОК ЛІТЕРАТУРИ

1. Дакетт Дж. HTML и CSS. Разработка и создание веб-сайтов / Дж. Дакетт. – Москва: Эскимо, 2021. – 480 с.
2. Кириченко А. В. Web на практике. CSS, HTML, JavaScript, MySQL, PHP для fillstack-разработчиков / А. В. Кириченко, А. П. Никольский, Е. В. Дубовик. – Москва: Наука и техника, 2021. – 431 с.
3. Зубик Л. В. Основи сучасних web-технологій. Ч.1 : навч. посіб. / Л. В. Зубик, І. М. Карпович, О. М. Степанченко. – Рівне : НУВГП, 2016. – 290 с. URL: http://ep3.nuwm.edu.ua/id/eprint/3686 (дата звернення 22.08.2021).
4. HTML, CSS, JavaScript и jQuery [Електронний ресурс]. – Режим доступу: https://html5book.ru/ (дата звернення: 22.08.2021).

ЗМІСТ

Навчальне видання

Марченко Наталя Андріївна
МАЛЬКО Максим Миколайович
СИДОРЕНКО Ганна Юріївна

ТЕХНОЛОГІЯ CSS

Навчально-методичний посібник
для студентів спеціальностей 124 «Системний аналіз»,
186 «Видавництво і поліграфія»

Відповідальний за випуск проф. Дорофєєв Ю.І.
Роботу до видання рекомендував проф. Безменов М. І.

У авторській редакції

План 2022 р., поз. 115

Підписано до друку 10.01.2023. Формат 60x84 1/16. Папір офсетний.
Друк – цифровий. Гарнітура Таймс. Ум. друк. арк. 6,3.
Наклад 50 прим. Ціна договірна. Замовл. № 61К

Видавничий центр НТУ «ХПІ»
Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.
61002, Харків, вул. Кирпичова, 2

Віддруковано з наданого оригінал-макету:
ТОВ «РІК-У», 88000, м. Ужгород, вул. Гагаріна, 36
Свідоцтво: Серія ДК № 5040 від 21 січня 2016 року