МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Волинський національний університет імені Лесі Українки
Кафедра комп’ютерних наук та кібербезпеки

Павленко Ю.С.

ВЕБДИЗАЙН

Навчально-методичний посібник

Для здобувачів освіти спеціальності
122 Комп’ютерні науки
першого (бакалаврського) рівня

Луцьк 2022

УДК 004.774.6(07)
П 12

Рекомендовано до видання науково-методичною радою
Волинського національного університету імені Лесі Українки
(протокол № 4 від 19 грудня 2022 р.)

Рецензенти:
Гембарська С.Б. – к.ф.-м.н., доцент кафедри теорії функцій та методики викладання математики Волинського національного університету імені Лесі Українки;
Здолбіцька Н.В. – к.т.н, доцент кафедри комп'ютерних наук Луцького національного технічного університету.

Павленко Ю.С. Вебдизайн : навчально-методичний посібник [Електронний ресурс] / Ю.С. Павленко; ВНУ імені Лесі Українки. – Електронні текстові дані (1 файл: 2,54 МБ). – Луцьк: ВНУ імені Лесі Українки, 2022. – 81 с.

Навчально-методичний посібник містить теоретичний матеріал до тем першого модуля нормативного освітнього компонента «Вебдизайн», а також приклади практичних завдань та зразки тестових запитань до модульного контролю. Теми укладені згідно силабусу освітнього компонента «Вебдизайн» для здобувачів освіти 1-го курсу спеціальності 122 Комп'ютерні науки. Навчально-методичний посібник є корисним для викладачів, які працюють зі студентами суміжних напрямів та здобувачів освіти, які навчаються за галуззю знань 12 «Інформаційні технології». Буде цікавий для розвитку додаткових вмінь та навичок з вебдизайну та верстки вебсторінок. Для студентів вузів, коледжів та учнів шкіл.

ЗМІСТ

Тема 1. Поняття дизайну, UI/UX дизайну, макету. Історія дизайну. Колористика у вебдизайні 4
Тема 2. Типографіка у вебдизайні 9
Тема 3. Мова гіпертекстової розмітки HTML. Структура HTML-документа. Теги для роботи з текстом 14
Тема 4. Мова гіпертекстової розмітки HTML. Теги для створення таблиць, малюнків та гіперпосилань 23
Тема 5. Теги для створення форм на вебсторінках 30
Тема 6. Використання каскадних таблиць стилів CSS. Поняття CSS, селектора, типи селекторів. Синтаксис написання селекторів. Використання класів та ідентифікаторів. Наслідування та каскадування 36
Тема 7. Блочна структура вебсторінки. Особливості табличної та блочної верстки. Семантична верстка HTML5. Позиціонування 46
Тема 8. Спеціальні селектори. Псевдоелементи, псевдокласи CSS та особливості їх використання при верстці вебсторінок 59
Тема 9. Реалізація переходів та анімацій на CSS3. Google-шрифти. CSS-фігури 67
СПИСОК ДОДАТКОВИХ ДЖЕРЕЛ 72
ДОДАТКИ 73
Додаток А Характеристика елементу <!DOCTYPE> 73
Додаток Б Таблиця спеціальних символів HTML 75
Додаток В Приклади тестових завдань для модульного контролю 77

Тема 1. Поняття дизайну, UI/UX дизайну, макету. Історія дизайну. Колористика у вебдизайні

1. Поняття дизайну.
2. Поняття вебдизайну.
3. Характеристика UI/UX дизайну: особливості та відмінності.
4. Поняття макету.
5. Огляд історії дизайну та вебдизайну.
6. Кольори у вебдизайні.

Слово «design» з'явилося в XVI столітті та вживалося по всій Європі. Італійський вислів «disegno intero» означав народжену у художника і викликану богом ідею – концепцію твору мистецтва. Оксфордський словник 1588 року подає таку інтерпретацію цього слова: «задуманий людиною план або схема чогось, що буде реалізовано, перший начерк майбутнього твору мистецтва».

1849 року в Англії вийшов перший у світі журнал, який мав у назві слово «дизайн» – «Journal of Design», заснований державним діячем, художником-проєктувальником сером Генрі Коулом.

У вересні 1969-го на конгресі Міжнародної ради організацій з дизайну було прийнято наступне визначення: «Під терміном дизайн розуміється творча діяльність, мета якої – визначення формальних якостей предметів, вироблених промисловістю. Ці якості форми відносяться не тільки до зовнішнього вигляду, але головним чином до структурних та функціональних зв'язків, які перетворюють систему в цілісну єдність з точки зору, як виробника, так і споживача».

В широкому розумінні дизайн – це художнє конструювання предметів побуту, формування гармонійного предметного середовища, створеного засобами промислового виробництва, для забезпечення найкращих умов праці побуту та відпочинку людей.

UI (User Interface) – це користувацький інтерфейс. Цим терміном описують те, як продукт виглядає: кольори, дизайн, анімація, контент, форми (кнопки, мітки, поля для вводу) тощо.

UX (User Experience) – це користувацький досвід. Тут мають на увазі емоції, які користувач отримує від час користування інтерфейсом на десктопному додатку чи у смартфоні. UX це рівень того, наскільки користувачу буде просто чи складно зробити те, за чим він прийшов на конкретний інтерфейс (здійснити покупку, прокласти маршрут, подивитися ціну, дізнатися погоду і т.і.).

Основну різницю між UI та UX можна описати так: UI-дизайнер вирішує, як виглядає кермо автомобіля, а UX-дизайнер – як це кермо працюватиме. До UX-дизайну належить функціонал продукту, шлях користувача по інтерфейсу, його взаємодія з ним та ті емоції, які він викликає у користувачів. UI – це робота над графічним втіленням функцій інтерфейсу.

UX-дизайнери аналізують дії користувачів, створюють прототипи поведінки, проводять тестування – щоб користувачеві було легше отримати необхідний йому результат. UI-дизайнер робить все, щоб пояснити користувачеві, як користуватися продуктом.

UX-дизайн – виявлення та розв’язання проблем користувача. UI-дизайн – створення інтуїтивно зрозумілих, естетично приємних, інтерактивних інтерфейсів. Якщо при користуванні інтерфейсом, наприклад, банківського застосунку, користувач не відразу розуміє, як саме потрібно здійснити платіж – отже, UX зроблений погано, і дизайнери не постаралися полегшити користування своїм продуктом.

Дизайнери, зокрема, досліджують різні підходи до розв’язання конкретної проблеми користувача, аби обрати найрелевантніший за швидкістю, зручністю, логікою. Які кроки робить користувач? Які завдання йому потрібно виконати? Наскільки простий шлях від запиту до результату?

UI-дизайнер займається візуалізацією прототипу, який він отримав від UX-дизайнерів. Він створює всі візуальні аспекти шляху користувача: екрани, скрол, типографіка, форми, кнопки, значки, інтервали, колірні схеми,

адаптивність. Дизайнери інтерфейсів фокусуються на всіх деталях, що стосуються шляху користувача. Від обраної кольорової гами залежить те, наскільки інтерфейс буде зручним для людей з порушеннями зору або ж наскільки легко читатиметься текст, коли на екран падає світло.

Колір є одним із найпотужніших засобів візуальної комунікації. Він може впливати на емоції, настрій і поведінку людей. Кольори описують за допомогою колірних моделей, які представляють різні способи (як правило, цифрові) опису кольорів. Виділяють багато колірних моделей – RGB, CMYK, Lab, HSB, HSL, Grayscale та ін.

Колірна модель RGB створена, щоб сприймати, представляти та показувати зображення в електронних системах. У цій моделі колір кодується градаціями складових каналів (Red, Green, Blue). Тому при збільшенні величини градації певного каналу зростає його інтенсивність під час синтезу. Якщо значення усіх каналів дорівнюватимуть 255, результатом буде чистий білий колір, якщо значення дорівнюватимуть 0 – чистий чорний.

Колірне коло (рис. 1) допомагає зрозуміти, як різні кольори співвідносяться один з одним і як їх можна поєднувати. Кольорове коло зазвичай будується з **основних, вторинних і третинних кольорів.** Первинними є ті три пігментні кольори, які не можуть бути утворені жодним поєднанням інших кольорів. Комбінуючи основні кольори, отримують вторинні, а суміш первинних і вторинних кольорів дає третинні кольори, які зазвичай мають двослівні назви, наприклад червоно-фіолетовий. Колірне коло було створено в 1666 році Ісааком Ньютоном у схематичний спосіб і з того часу пройшло через багато трансформацій, але все ще залишається основним інструментом для поєднання кольорів.

Колірна схема – це термін, який дизайнери використовують для опису комбінацій кольорів, які вони використовують у дизайні. Виділяють 5 типів колірних схем.

Монохромна схема складається з різних тонів, відтінків і насиченості одного основного кольору. Вони дуже згуртовані, але ризикують стати одноманітними.

Комплементарні схеми базуються на двох кольорах з протилежних сторін колірного кола. Оскільки ці два відтінки будуть дуже різними, такі схеми можуть бути дуже ефектними та помітними.

Аналогічні колірні схеми містять три кольори, розташовані поруч один з одним на колірному колі. Завдяки тональній подібності ці схеми можуть створити дуже цілісне, єдине відчуття без монотонності монохромної схеми.

Щоб створити триадичну колірну схему, слід намалювати рівносторонній трикутник на колірному колі та вибрати три кольори в точках трикутника. Ця тріада створює різноманітну, але збалансовану схему.

Тетрадна колірна схема включає чотири кольори, які знаходяться на однаковій відстані один від одного на колірному колі. Оскільки чотири кольори можуть утворювати квадрат або прямокутник, деякі ресурси розбивають ці колірні схеми на дві: квадратну і прямокутну.

Рисунок 1 – Колірне коло

Особлива проблема, з якою стикаються дизайнери, коли вони працюють з кольорами, полягає в тому, як люди з дальтонізмом або дефіцитом кольорового зору взаємодіятимуть із продуктом. На рис. 2 кольорові кола показують, як виглядають кольори для людей із нормальним зором (крайній ліворуч) і ті самі кольори, які бачать люди з дефіцитом червоно-зеленого кольору (посередині та справа).

Рисунок 2 – Приклад різного сприйняття кольорів людьми

У вебдизайні рекомендують використоувати правило 60-30-10, яке полягає в тому, щоб використовувати три кольори: основний колір для 60% дизайну, допоміжний колір для 30% дизайну та колір акценту для останніх 10%. Хоча це не можна ввжати точними цифри, вони допомагають надати сайту відчуття пропорції та балансу.

Запитання та завдання до теми

1. Що таке дизайн та вебдизайн?
2. Що таке UI-дизайн?
3. Що таке UX-дизайн?
4. Пояснити особливості та відмінності UI/UX дизайну.
5. Поняття макету.
6. Огляд історії дизайну та вебдизайну.
7. Кольори у вебдизайні.
8. Що таке колірна модель?
9. Що таке колірне коло?

Тема 2. Типографіка у вебдизайні

1. Поняття типографіки та її аспекти. Типографіка у веб.
2. Основні поняття типографіки: шрифт, гарнітура, кегль, інтерліньяж.
3. Поняття трекінгу та кернінгу.
4. Поняття модульної сітки.
5. Основні правила типографіки.

Типографіка – це напрям графічного дизайну і система оформлення друкованого тексту. Вона, зокрема, передбачає вибір гарнітури, кеглю, довжини рядка, міжрядкового і міжсимвольного інтервалу. Мета типографіки – зробити текст легким для сприйняття й привабливим, адже утримати увагу глядача у переповненому візуальним контентом світі може бути непростим завданням.

Головними аспектами типографіки є чіткість, читабельність та естетика. Чіткість визначає, наскільки зручним є текст для сприйняття. Читабельність більше стосується легкості розуміння, а естетика відповідає за загальну привабливість шрифту чи комбінації шрифтів.

Типографіка у веб – це правила оформлення тексту сайту з метою донесення інформації до аудиторії в найлегшій для читання і сприйняття формі. Формат оформлення впливає на показники часу проведеного користувачем на сайті, вчинення цільової дії та загального враження від сайту.

У вебдизайні функціональність завжди важливіше графічних рішень. Дизайнери роблять продукт для людини, і він повинен бути зручним, простим і зрозумілим. За допомогою шрифтів, типографіки формується фірмовий стиль – важливий інструмент реклами. Єдиний фірмовий стиль передбачає сукупність художніх прийомів, що створюють єдиний характер подачі рекламних матеріалів, розроблених на базі оригінального графічного дизайну.

Текст – найголовніший елемент на сторінці, саме він підштовхує користувача до дії, тому для створення зрозумілого, читабельного тексту, в якому легко орієнтуватись слід дотримуватись правил типографіки. Не важливо

наскільки корисна інформація міститься в текстовому блоці, якщо текст неправильно оформлений – він залишиться без уваги.

Шрифт – це набір малих і великих символів, знаків пунктуації, цифр і спецсимволів одного розміру і товщини для окремої гарнітури. Під шрифтом мається на увазі графічне зображення накреслень літер і знаків, які створюють єдину стилістичну й композиційну систему. Шрифти відрізняються за нахилом (прямий, похилий, курсив), насиченістю (контурний, напівжирний, жирний та інші), шириною (надвузький, вузький, нормальний, широкий, надширокий) тощо.

Є багато класифікацій шрифтів, та, згідно з найпоширенішою, їх чотири:

- шрифти із зарубками, або антиква (Times New Roman, Courier New, Baskerville, Georgia). Антикви – класичний вид, посилюється кнтрастність основних і сполучних штрихів, з'являються широкі заокруглені або прямокутні засічки. Використовують в заголовках, текстових блоках, що стилізовані під друковані видання або певний історичний період;

- рубані шрифти без зарубок, або гротески (Arial, Verdana). Гротески – легко читаються на екранах пристроїв через відсутність контрасту основних і сполучних штрихів, відсутність засічок, а також збільшену висоту малих літер. Використовуються у всіх блоках. Arial Inter Roboto PTSans Montserrat OpenSans;

- рукописні шрифти (Snell Roundhand, Vivaldi Regular, AmadeusAP Regular, Hamiltone Signature, Alana Regular);

- акцидентні або декоративні шрифти (Marvin Visions, SK Primo, Postertoaster, Pitcrew).

Дизайнери зазвичай працюють із першими двома видами. Рукописні й акцидентні шрифти використовують для акцентів і заголовків, оскільки вони привертають увагу своєю оригінальністю.

Гарнітура – об'єднання шрифтів, які мають однаковий характер накреслення, але відрізняються за кеглем і нарисом («комплект» шрифтів, що мають спільні стильові ознаки та принципи побудови знаків). Іноді гарнітура має одне накреслення. У випадку, коли їх декілька, існує основне, призначене для набору основного тексту, та додаткові – для заголовків та виділення

фрагментів у тексті. Приклад – шрифт Roboto Regular та варіанти накреслення: Roboto Bold (жирне), Roboto Italic (курсив), Roboto Bold Italic (жирний курсив) (рис. 3).

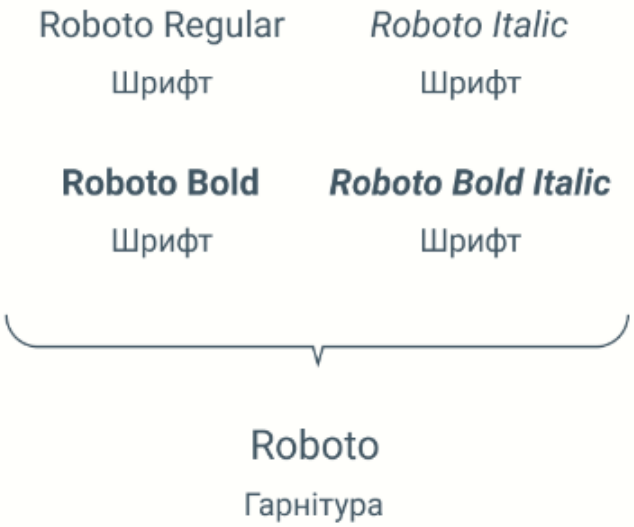


Рисунок 3 – Приклад гарнітури і шрифтів

Кегль визначає розмір шрифту і вимірюється у типографських пунктах. Один пункт становить 0,352777 мм. Шрифти від 3 до 12 пунктів називають текстовими, від 14 до 64 – заголовковими, від 72 – плакатними.

Інтерліньяж (або міжрядковий пробіл) – це вертикальна відстань між двома лініями рядків. Вона також вимірюється у пунктах і залежить від кегля шрифту і міжрядкової відстані.

Міжрядкова відстань – це відстань між сусідніми літерами чи іншими знаками. З цим елементом також пов’язані поняття **трекінгу** та **кернінгу**. Трекінг передбачає налаштування інтервалу між всіма символами у межах одного слова й часто застосовується у заголовках і логотипах. У випадку з кернінгом можна змінювати відстань між окремими парами літер, щоб уникнути неоднорідності тексту.

Модульна сітка є базовою структурою, що визначає розташування тексту, графічних елементів і ілюстрацій, головний інструмент структурування елементів дизайну. Вона зазвичай складається з низки прямих ліній, що створюють своєрідну рамку для впорядкування всіх складових. Сітки дозволяють розташувати елементи рівномірно й пропорційно, мати єдину

форму основним елементам сторінки. Сітка складається з декількох недрукованих горизонтальних і вертикальних ліній, що поділяють сторінку.

Основні правила типографіки:

- використання мінімальної кількості шрифтів. Варто обмежитися двома-трьома варіантами шрифтів залежно від завдання. Більшу кількість буде досить важко узгодити між собою. Доречно використовувати класичні комбінації – антиква у заголовку (наприклад, Garamond, Bodoni або Georgia) та гротеск в основному тексті (наприклад, Helvetica, Roboto, Open Sans) або навпаки. Можна використовувати різні шрифти однієї гарнітури;

- обчислення кеглю шрифта за правилом золотого перетину. За принципом ієрархії: чим важливіша інформація, тим більшим кеглем вона набирається. Оскільки завдання заголовка – привернути увагу до повідомлення, саме він має бути найпомітнішим. Основний текст оформлюють середнім кеглем, а примітки й коментарі – дрібним. Для отримання оптимального співвідношення між розміром заголовку та основним текстом: розмір тексту*1,6, оскільки 1,6 – “золоте” число, що відповідає правилу золотого перетину. Вважається, що воно створює ідеальні пропорції, які виглядають максимально гармонійно й легко сприймаються. Тобто якщо основний текст набраний 18 кеглем, розмір заголовка має бути 28;

- оптимальний інтерліньяж для читабельності тексту. Адже якщо рядки розташовані надто близько один до одного, розібрати їх буде досить складно. Так само незручно читати, коли відстань між лініями велика. Існує стандарт, за яким оптимальний інтерліньяж становить 120% від кегля шрифту. Тобто, якщо кегль 18, інтерліньяж має бути 22. Величина міжрядкового пробілу повинна бути однаковою для всього тексту, щоб не порушувати ритм читання. У випадку з оформленням заголовка інтерліньяж зверху від нього має бути більшим, ніж знизу, аби було зрозуміло, що він стосується конкретного абзацу;

- вибір оптимальної довжини рядка для зручного читання. За Емілем Рудером (відомий швейцарський типограф і графічний дизайнера минулого століття) оптимальна довжина рядка повинна становити 50-60 символів з пробілами. У деяких випадках, зокрема у вебдизайні, вона може бути збільшена

до 80 символів. Для мобільної версії зручна довжина має бути вдвічі меншою. Довжина також визначає і висоту рядка. Тут працює пряма залежність: чим довший рядок, тим більша висота.

- колір і контраст;
- вирівнювання тексту;
- створення візуальної ієрархії. Візуальна ієрархія – це структурування й підпорядкування елементів у дизайні. Вона визначає, що головне, що другорядне, і як вони об’єднуються в систему. Візуальна ієрархія надважлива, оскільки саме від неї залежить user (customer) journey – шлях користувача, тобто вона спрямовує увагу глядача на ключові моменти. Ієрархія може бути створена, зокрема, завдяки кеглю (важливіший текст більший за розміром), кольору (приклад – колірне виділення слів чи фрагментів тексту, які треба акцентувати) тощо. Варто використовувати просту сітку. Вона дозволить згрупувати й вирівняти всі елементи й отримати в результаті узгоджений дизайн.

Запитання та завдання до теми

1. Що таке типографіка?
2. Перерахувати аспекти типографіки.
3. Що таке шрифт? Скільки різних шрифтів оптимально використовувати на вебсторінці?
4. Що таке гарнітура?
5. Що таке кегль? Які розміри кеглю правильно використовувати на вебсторінках?
6. Що таке модульна сітка і для чого її використовувати?
7. Що таке інтерліньяж? Як правильно вибирати інтерліньяж?
8. Що таке візуальна ієрархія?

Тема 3. Мова гіпертекстової розмітки HTML. Структура HTML-документа. Теги для роботи з текстом

1. Поняття HTML.
2. Поняття тегу та синтаксис написання.
3. Структура HTML-документа.
4. Основні теги для роботи з текстом: створення заголовків, абзаців, цитат, верхніх та нижніх індексів, абревіатур.
5. Теги для створення списків: маркованих, нумерованих, списків визначень.
6. Відображення спеціальних символів на вебсторінках.
7. Способи перенесення тексту на наступний рядок.

HTML (HyperText Markup Language – Мова розмітки гіпертексту) – стандартна мова розмітки документів у WWW; система верстки, яка визначає, як і які елементи повинні розміщатися на вебсторінці.

Мова означає, що вона може бути прочитана як людиною, так і комп’ютером.

Розмітка означає, що код пишеться з допомогою ключових слів.

Гіпертекст означає, що він використовує HTTP як частину Інтернет.

Тег – це елемент мови розмітки гіпертексту, призначений, в основному, для задання того, як буде відображатися текст (сторінка). Розрізняють парні та непарні теги.

Загальний синтаксис написання тегів:

<тег атрибут1="значення" атрибут2="значення">

<тег атрибут1="значення" атрибут2="значення">...</тег>

При написанні тегів взагалі кажучи можна використовувати як великі так і малі літери. Рекомендується назви тегів писати малими літерами.

Вміст тегу можна відображати в декількох рядках. Але довільна кількість пробілів, які написані підряд, в браузері відображається як один. Довільна кількість натискань клавіші ENTER в браузер не переноситься. Для забезпечення такого відображення використовуються інші способи, про що буде розказано нижче.

Для розширення можливостей деяких елементів застосовуються **атрибути**. Є два типи атрибутів: атрибут зі значенням і логічний атрибут, у якого немає значення. Атрибути пишуться всередині відкриваючого тега, декілька атрибутів розділяються пробілом, їх порядок значення не має. Згідно специфікації HTML всі значення атрибутів слід записувати в подвійних або одинарних лапках.

Структура HTML-документа:

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

Наведемо призначення деяких основних та часто використовуваних тегів HTML.

<!DOCTYPE html> – вказує на те, що вебсторінка створена в стандарті HTML5. Елемент <!DOCTYPE> призначений для вказівки типу поточного документа – DTD (document type definition, опис типу документа). Це необхідно, щоб браузер розумів, як слід інтерпретувати поточну вебсторінку.

<html> ... </html> – вказують на те, що вебсторінка створена засобами HTML.

<head> ... </head> – початок і кінець заголовка документа. Крім назви документа в цей розділ може включатися службова інформація.

<title> ... </title> – все, що знаходиться між тегами **<title>** і **</title>**, сприймається браузером як назва документа. Відображається в рядку заголовка браузера. Рекомендується назва не довше 64 символів.

<body> ... </body> – початок і кінець тіла HTML-документа, яке визначає вміст документа.

<h1> ... </h1> – <h6> ... </h6> – описують заголовки шести різних рівнів. Заголовок 1-го рівня – найкрупніший, 6-го рівня – найдрібніший**.**

** ... –** все, що знаходиться між ними, сприймається як один абзац.

**
 –** встановлює переведення рядка в тому місці, де цей тег знаходиться. На відміну від тегу абзацу ** **, використання тегу **
** не додає пустий відступ перед рядком.

_… – перетворює текст між тегами у нижній індекс.

[…] – перетворює текст між тегами у верхній індекс.

<!-- … --> – призначений для коментарів, ігнорується браузерами і не відображається на вебсторінці.

<dfn> … </dfn> – відображає текст курсивом, призначений для позначення тексту як визначення.

<blockquote> … </blockquote> – використовується для позначення цитат виділенням відступами з обох сторін.

<q> … </q> – використовується для позначення цитат, автоматично доставляються лапки перед та після тексту.

<cite> … </cite> – позначає текст, як цитату або зноску, зручно для форматування через CSS. Браузери відображають текст всередині курсивом.

** … –** відображає текст курсивом, призначений для зміни вигляду тексту.

** … ** – відображає текст курсивом, призначений для акцентування тексту.

<var> … </var> – відображає текст курсивом, призначений для позначення змінних у тексті.

** … ** – відображає текст напівжирним, без акценту на його важливість.

** … –** відображає текст напівжирним, призначений для акцентування тексту.

<small> … </small> – зменшує розмір шрифта на одиницю по відношенню до звичайного тексту.

<s> … </s> – відображає текст перекресленим, використовується для вмісту, який вже не є актуальним.

<del> … </del> – відображає текст перекресленим, використовується для виділення тексту, який був видалений в новій версії документа.

<u> … </u> – відображає текст підкресленим.

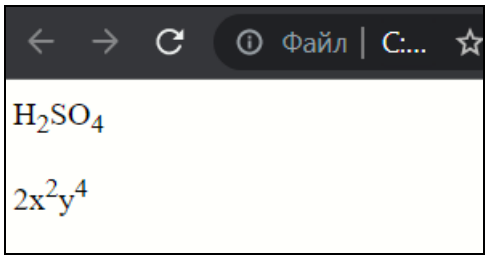
<ins> … </ins> – виділяє текст в новій версії документа, підкреслюючи його.

<mark> … </mark> – виділяє колір фону тексту жовтим, призначений для виділення тексту в довідкових цілях.

<hr /> – дозволяє відобразити горизонтальну лінію на всю ширину вікна браузера.

<wbr> (або ****) – вказує браузеру місце, де можна зробити перенесення слова в тексті. Різниця в тому, що **** додає при переносі дефіс, а **<wbr>** – ні.

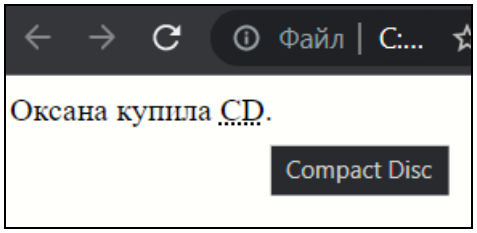
Наприклад:

Код	Результат
 H₂SO₄ 2x²y⁴ 	

<abbr> … </abbr> – вказує, що послідовність символів є абревіатурою. Атрибут **title** забезпечує відображення розшифровки абревіатури, якщо на неї навести вказівник миші. Абревіатура, оформлена таким чином, візуально відображається підкресленою крапками. Крім того, пошукові системи індексують повнотекстовий варіант скорочення, що може використовуватись для підвищення рейтингу документа.

Наприклад:

Код	Результат

 Оксана купила <abbr title="Compact Disc"> CD </abbr>. 	

Буває так, що на вебсторінках потрібно відобразити символи, яких немає на клавіатурі (грецькі букви, знак копірайт «©» тощо) або які не коректно відобразяться на вебсторінці, оскільки є частиною синтаксису HTML (знаки менше «<», більше «>». В такому випадку можна скористатися кодами з додатку 1. Для прикладу наведемо способи задання кількох спеціальних символів.

Ім’я	Код	Вид	Опис
&copy;	&#169	©	знак копірайт
&reg	&#174	®	знак зареєстрованої торгової марки
&lt;	&#60	<	знак «менше»
&gt;	&#62	>	знак «більше»
&xi	&#958	ξ	грецька мала буква ксі

Як і в текстових процесорах, на вебсторінках можна організувати нумеровані та ненумеровані (марковані) списки. А також так звані списки визначень (означень). Розглянемо їх.

Ненумерований список. Синтаксис:

<ul>

**
** перший елемент списку **
**

**
** другий елемент списку **
**

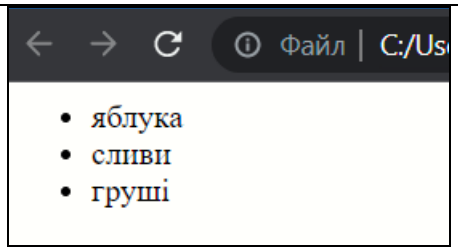
**
** третій елемент списку **
**

...

**
**

Наприклад:

Код	Результат

    - яблука - сливи - груші 	

Нумерований список. Синтаксис:

<ol>

**
** перший елемент списку **
**

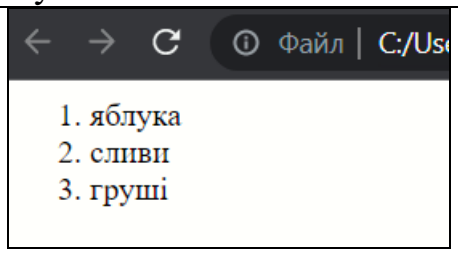
**
** другий елемент списку **
**

**
** третій елемент списку **
**

...

**
**

Наприклад:

Код	Результат
    - яблука - сливи - груші 	

Списки визначень використовуються для опису визначень і термінів. Синтаксис:

<**dl**>

<**dt**>Термін 1</**dt**>

<**dd**>Визначення першого терміну</**dd**>

<**dt**>Термін 2</**dt**>

<**dd**>Визначення другого терміну</**dd**>

...

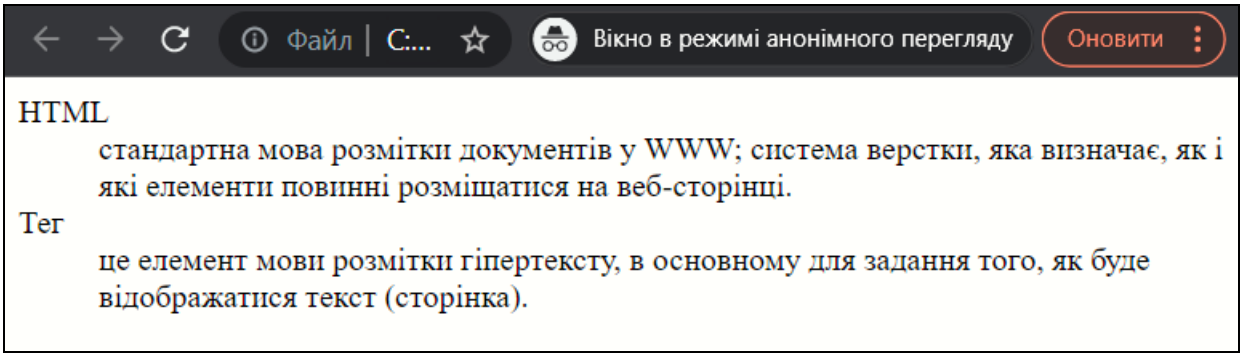
</**dl**>

Наприклад:

Код
<dl> <dt>HTML</dt>

```
    <dd> стандартна мова розмітки документів у WWW; система
верстки, яка визначає, як і які елементи повинні розміщатися на
вебсторінці.</dd>
<dt>Тег</dt>
    <dd> це елемент мови розмітки гіпертексту, в основному для
задання того, як буде відображатися текст (сторінка).</dd>
</dl>
```

Результат



Додатково списки огортати тегом абзацу не потрібно. Можливе створення вкладених списків. Для цього треба правильно організувати вкладеність потрібних тегів.

Важливо пам’ятати: при написанні правильного коду грає роль порядок тегів. Якщо маємо ієрархію вкладених тегів, то теги закриваються у зворотньому порядку до їх відкривання.

Запитання та завдання до теми

1. Що таке HTML?
2. Що таке тег?
3. Навести синтаксис написання тегу.
4. Навести структуру вебсторінки.
5. Які теги використовується для відображення абзацу на вебсторінці?
6. Який тег використовується для задання заголовків на вебсторінці?
7. Чим відрізняються теги, які відображають текст на вебсторінці курсивом?
8. Чим відрізняються теги, які відображають текст на вебсторінці напівжирним шрифтом?
9. Який тег використовується для задання абревіатури на вебсторінці?
10. Як задають списки на вебсторінці?

11. Які теги використовується для задання верхніх та нижніх індексів на вебсторінці?
12. Як можна відобразити спеціальні символи, яких немає на клавіатурі, на вебсторінці?

Завдання 1. Створити вебсторінку за зразком тільки засобами HTML, використовуючи теги лише з теоретичного матеріалу теми 3, за виключенням тегів для створення списків. Приблизний варіант сторінки наведено на рис. 4. **ФРМ** оформити як абревіатуру. Текст для вебсторінки додається у текстовому файлі.

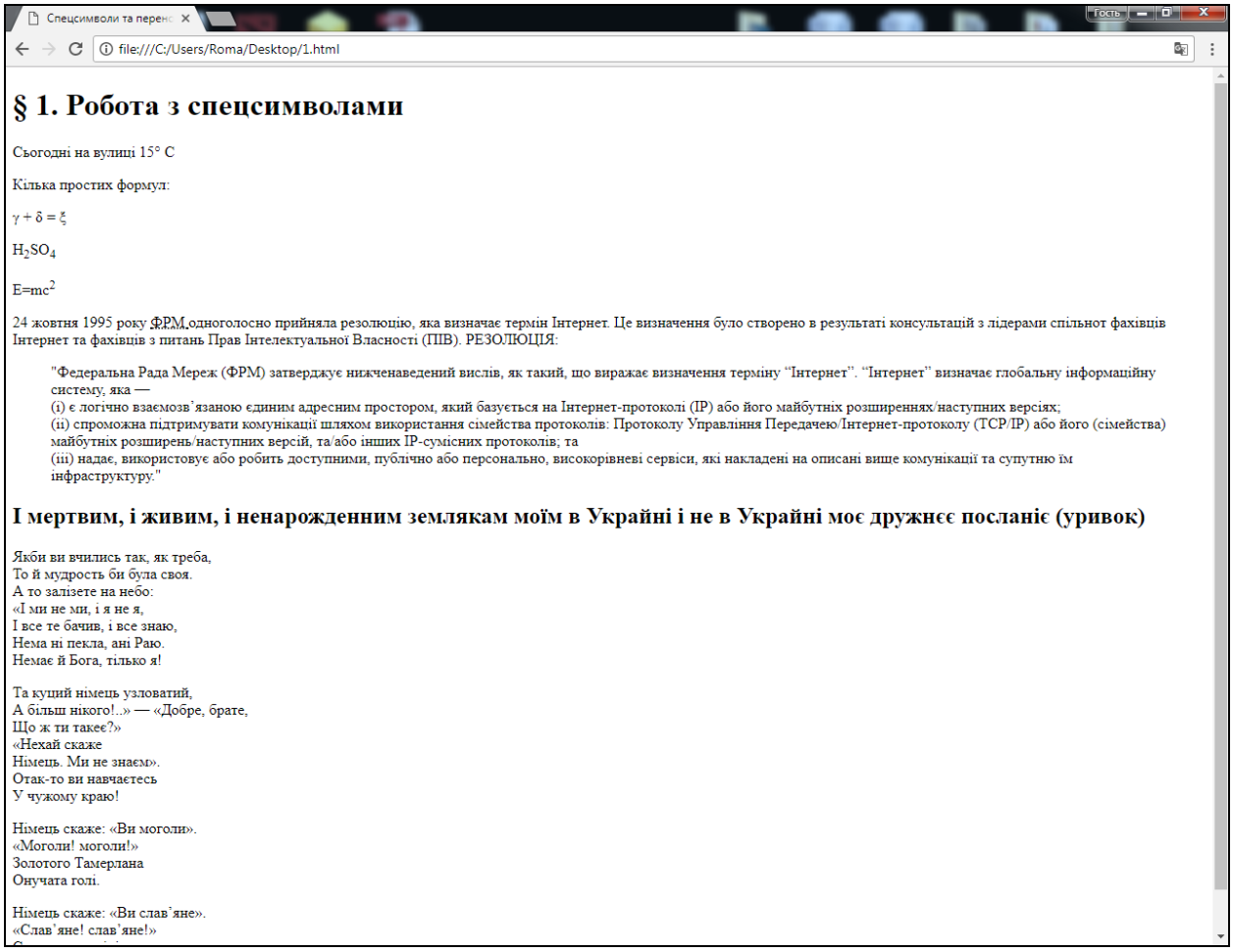


Рисунок 4 – Знімок екрану зразка вебсторінки «Символи»

Завдання 2. Створити вебсторінку за зразком тільки засобами HTML, використовуючи теги лише з теоретичного матеріалу теми 1. Приблизний

варіант сторінки наведено на рис. 5. Текст для вебсторінки додається у текстовому файлі.

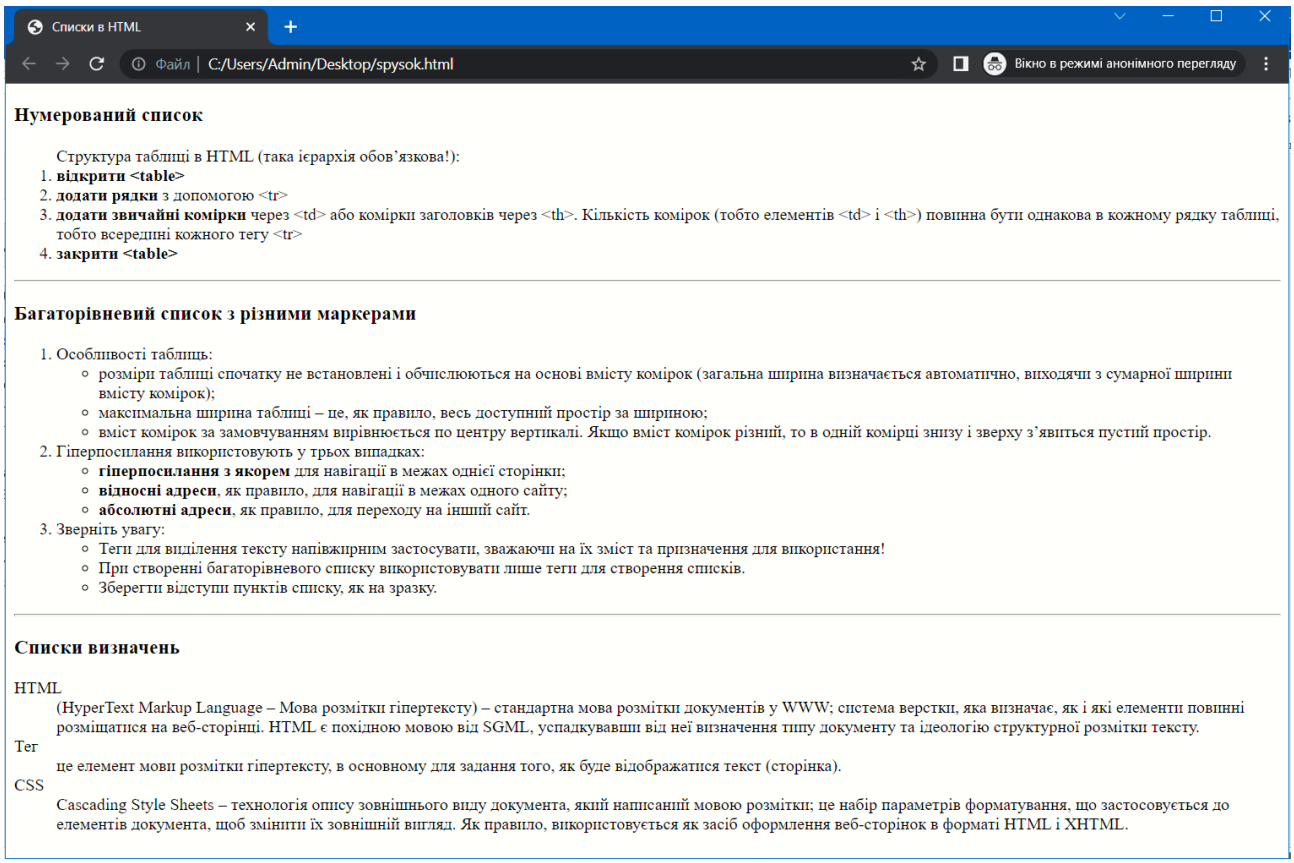


Рисунок 5 – Знімок екрану зразка вебсторінки «Списки в HTML»

Тема 4. Мова гіпертекстової розмітки HTML. Теги для створення таблиць, малюнків та гіперпосилань

1. Теги для створення таблиці: відображення заголовка таблиці, оформлення першого рядка таблиці, об'єднання комірок.
2. Тег для розміщення малюнка на вебсторінці та його основні атрибути.
3. Тег для створення гіперпосилання та його атрибути.
4. Тег для створення горизонтальної розділювальної лінії.

Таблиця в HTML задається з допомогою парного тегу **<table> … </table>** та повинна витримувати певну структуру. Додатково для таблиці тег абзацу писати не потрібно. Структура таблиці в HTML:

1. відкрити таблицю **<table>**;
2. додати рядки з допомогою **<tr>… </tr>**;
3. додати звичайні комірки через **<td>… </td>** або комірки заголовків через **<th>… </th>**. Кількість комірок (тобто елементів **<td>** і **<th>**) повинна бути однакова в кожному рядку таблиці, тобто всередині кожного тегу **<tr>… </tr>**;
4. закрити таблицю **</table>**.

Тег **<table>** має ряд атрибутів, які сьогодні мало використовуються. Але один з них наведемо тут для того, щоб візуально побачити створювану таблицю на вебсторінці. У всіх наступних випадках при створенні таблиці доцільно користуватися властивостями CSS, які будуть розглянуті нижче. Отже,

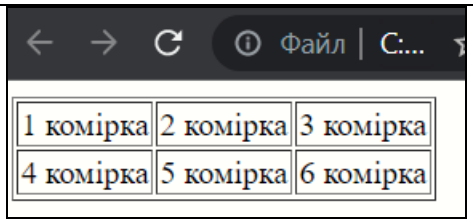
border – атрибут тегу **<table>**, який задає ширину межі таблиці в пікселях. За замовчуванням дорівнює нулю (межі таблиці не відображаються).

Наприклад:

Код	Результат

`<table border="1">`
`<tr>`
`  <td>1 комірка</td>`
`  <td>2 комірка</td>`
`  <td>3 комірка</td>`
`</tr>`	

```
<tr>
     <td>4 комірка</td>
     <td>5 комірка</td>
     <td>6 комірка</td>
</tr>
</table>
```



Для того, щоб над таблицею відображався заголовок слід написати тег **<caption> ... </caption>** відразу після **<table>**. Додатково **<caption> ... </caption>** огортати тегом абзацу не потрібно.

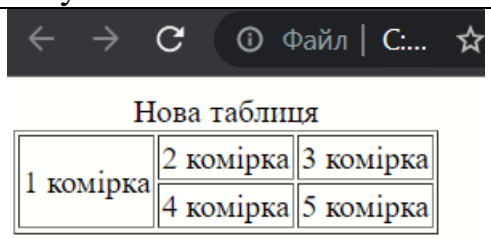
Для об'єднання комірок по горизонталі та вертикалі використовують атрибути тегу **<td>**:

rowspan – об'єднує вказану кількість комірок в одну по вертикалі;

colspan – об'єднує комірки по горизонталі.

Наприклад:

Код	Результат

`<table border="1">`
`<caption>Нова таблиця</caption>`
`<tr>`
`  <td rowspan="2">1 комірка</td>`
`  <td>2 комірка</td>`
`  <td>3 комірка</td>`
`</tr>`
`<tr>`
`  <td>4 комірка</td>`
`  <td>5 комірка</td>`
`</tr>`
`</table>`
	

Елементи розмітки **<tbody>**, **<thead>** і **<tfoot>** призначені для об'єднання рядків таблиці у групи. В межах однієї таблиці ці елементи можна використовувати лише один раз.

<thead> … </thead> створює групу заголовків для рядків таблиці з метою задання єдиного оформлення. Елемент повинен бути використаний в наступному порядку: як дочірній елемент **<table>**, після **<caption>** і **<colgroup>**, і перед **<tbody>**, **<tfoot>** і **<tr>**.

<tbody> … </tbody> групує основний вміст таблиці.

<tfoot> … </tfoot> створює групу рядків для представлення підсумкової інформації, що розташовується в нижній частині таблиці. Розташовується після елемента **<thead>**, перед елементами **<tbody>** і **<tr>**.

Таке групування рядків закладене у стандарті з розрахунку, що браузери при відображенні довгих таблиць забезпечать прокрутку рядків при збереженні над заголовку та підзаголовку нерухомими, а при їх виведенні на принтер зможуть використовувати над заголовки і підзаголовки в якості колонтитулу сторінки. Але сучасні браузери цього не роблять, тому дані елементи можна використовувати в якості логічних, що може бути використане при форматуванні таблиці з допомогою CSS.

Зображення (фото, картинки, схеми, рисунки) на вебсторінках розміщають з допомогою тегу **<img>**. Обов'язковими атрибутами цього тегу є наступні:

src – вказується джерело зображення. Використовується абсолютна або відносна адреса зображення. Як правило – це шлях відносно поточного документа, але можна використовувати і URL в Інтернет;

alt – задає текст, який буде відображатися замість зображення, якщо вона не завантажилася або не відобразилася. Як правило, в цьому тексті вказується короткий опис зображення або посилання.

Відносні адреси вказуються від кореня сайту або поточного документу:

1. **<img src="pic.png">** означає завантажити файл pic.png, який розміщений в тій же папці, що й сама вебсторінка;

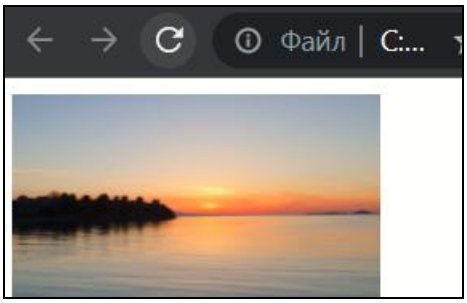
2. **/images/pic.png** – «/» перед адресою означає, що адресація починається від кореня сайту. Файл pic.png знаходиться в папці images, а вона в корені сайту;

3. **../images/pic.png** – дві крапки перед іменем вказують на те, що потрібно перейти на рівень вище в списку папок сайту і в папці images знайти файл pic.png;

4. **images/pic.png** – якщо перед іменем папки немає ніяких додаткових символів, то папка розміщена всередині поточної папки, а в ній знаходиться потрібний файл.

Для того, щоб зображення на вебсторінці відображалося з нового рядка, потрібне додаткове використання тегу абзацу або тегу переходу на новий рядок.

Наприклад:

Код	Результат
 <img src="image1.png" alt="Фото"> 	

Вебсторінки, як правило, завжди містять гіперпосилання для забезпечення можливості переходу на інші вебсторінки. Гіперпосилання використовують у трьох випадках:

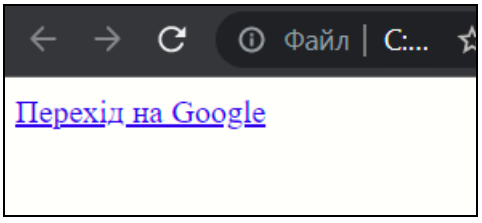
- відносні адреси, як правило, для навігації в межах одного сайту;
- абсолютні адреси, як правило, для переходу на інший сайт;
- гіперпосилання з якорем для навігації в межах однієї сторінки.

Синтаксис:

** текст гіперпосилання **

Для забезпечення відображалося гіперпосилання з нового рядка потрібне додаткове використання тегу абзацу або тегу переходу на новий рядок.

Наприклад:

Код	Результат
  Перехід на Google 	

Гіперпосилання з якорем призначені для навігації в межах однієї сторінки. Якорем називають закладку з унікальним іменем в певному місці вебсторінки, яка призначена для переходу на неї за гіперпосиланням. Якорі

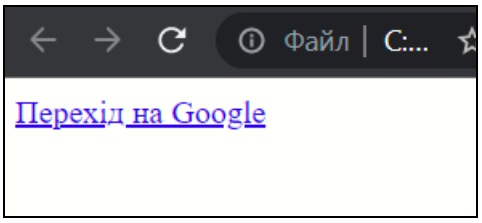
зручно застосовувати в великих документах для швидкого переходу до певного розділу.

Створення якоря:

1. Зробити закладку у відповідному місці і дати їй унікальне ім’я з допомогою атрибута **id**.

2. При створенні гіперпосилання в якості значення **href** для переходу до цього якоря використовується значення **id** з символом решітки (#) попереду.

Наприклад:

Код	Результат
  Перехід на Google 	

Запитання та завдання до теми

1. Який тегом позначають початок таблиці на вебсторінці?
2. Які теги використовують для позначення рядків та стовпців таблиці?
3. Які атрибути призначені для об’єднання рядків та стовпців таблиці?
4. Яким тегом позначають гіперпосилання?
5. Який тег використовують для відображення малюнків на вебсторінці?
6. Назвати обов’язковий атрибут тегу **<a>**.
7. Назвати обов’язковий атрибут тегу **<img>**.
8. Що таке якір і як його створити?

Завдання 1. Створити таблиці лише з допомогою тегів та атрибутів HTML, наведених вище. Приблизний варіант сторінки наведено на рис. 6.

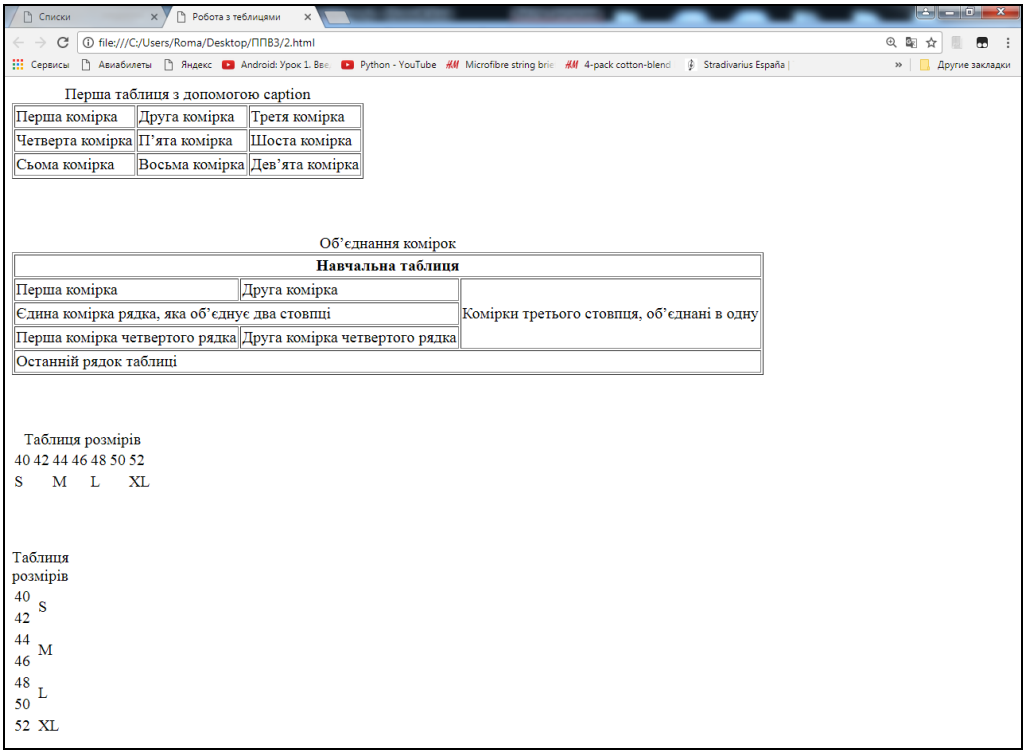


Рисунок 6 – Знімок екрану зразка вебсторінки «Таблиці»

Завдання 2. Створити вебсторінку за зразком з допомогою тегів та атрибутів HTML, наведених вище (малюнки можуть бути інші). Обов'язкова наявність закладок (якорів). Приблизний варіант сторінки наведено на рис. 7-8. Текст для вебсторінки додається у текстовому файлі.

Рисунок 7 – Знімок екрану зразка вебсторінки «Малюнки і гіперпосилання»

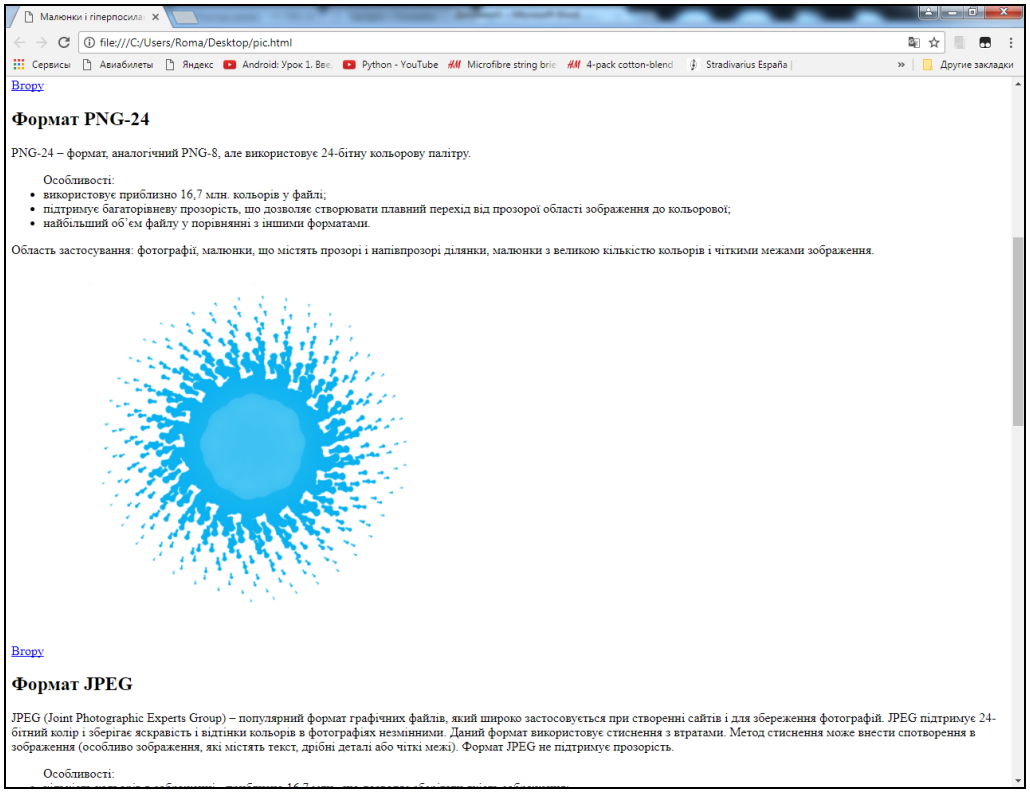


Рисунок 8 – Знімок екрану зразка вебсторінки «Малюнки і гіперпосилання» (продовження)

Завдання 3.

1.Створити вебсторінку зі списком гіперпосилань на свої попередні роботи. На кожній із попередніх робіт (веб-сторінок) додати посилання для повернення на головну сторінку «Мої виконані завдання».

Наприклад.

Мої виконані завдання:

1. Робота з текстом
2. Списки в HTML
3. Таблиці в HTML
4. Графіка в HTML

2. Зробити гіперпосилання на зовнішній ресурс за Вашим бажанням.

3. Зробити гіперпосилання у вигляді зображення на довільний ресурс.

Тема 5. Теги для створення форм на вебсторінках

1. Поняття форми та типи елементів управління.
2. Тег для додавання форми на вебсторінку.
3. Призначення тегу <input> та його атрибути.
4. Теги для створення випадаючих списків на вебсторінці.
5. Тег для створення багаторядкових коментарів.

Тег **<form> … </form>** є контейнером для елементів управління і дає можливість вводити інформацію і відправляти її на сервер. Атрибути тегу:

- **name** – визначає ім’я форми, унікальне для даного документа. Використовується, якщо в документі є кілька форм;

- **action** – обов’язковий атрибут. Визначає URL-адресу, на яку буде направлений вміст форми – шлях до скрипта сервера, який обслуговує дану форму;

- **method** – визначає спосіб відправлення вмісту форми. Можливі значення **get** (за замовчуванням) і **post**. При значенні **get** інформація з форми додається в кінець URL, яка вказується вище. **post**: передає інформацію про форму відразу після звертання до вказаної URL-адреси;

- **enctype** – визначає спосіб кодування вмісту форми при відправленні. За замовчуванням використовується "application/x-www-form-urlencoded“;

- **target** – визначає ім’я вікна, в яке буде повернуто результат обробки відправленої форми.

<input> – призначений для створення елементів управління і завжди використовується разом з атрибутом **type**, який визначає тип елемента управління. Атрибути тегу **<input>**:

- **value** – значення елемента;
- **placeholder** – виводить текст-підказку;
- **size** – встановлює кількість видимих символів в text;
- **name** – ім’я елементу;

- **maxlength** – встановлює максимально допустиме число символів, які можна ввести в **text**;

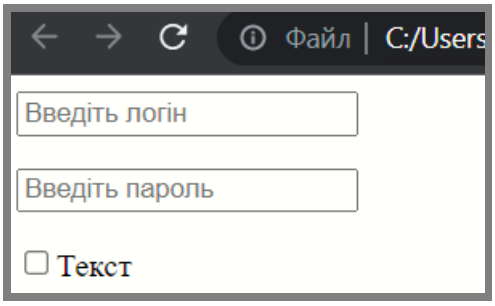
- **checked (**false або true) – приймає початковий стан для checkbox і radio (за замовчуванням false);

- **type** – обов'язковий атрибут тегу **<input>,** описує тип інтерфейсного елемента. Деякі значення атрибуту **type**:

- **type = "text"** – створює рядок для введення тексту;
- **type = "password" –** створює рядок для введення тексту, при цьому відображає символи, які вводяться у вигляді зірочок;
- **type = "checkbox"** – створює перемикач "прапорець";
- **type = "radio"** – створює перемикач, який дозволяє вибрати одне значення з кількох альтернативних. Оскільки перемикачі є груповими елементами, то ім'я (name) у всіх елементів групи повинно бути однаковим;
- **type = "file"** – створює елемент для вибору локальних файлів;
- **type = "hidden"** – створює невидимий для користувача елемент. Може використовуватись для відправки додаткової службової інформації (наприклад, пароль);
- **type = "button"** – створює стандартну кнопку;
- **type = "submit"** – створює кнопку «Надіслати»;
- **type = "reset"** – створює кнопку «Скинути». При натисканні всі поля форми приймуть значення, дані їм за замовчуванням.
- **type = "color"** – віджет для вибору кольору;
- **type = "number"** – для введення чисел;
- **type = "range"** – повзунок для вибору чисел у вказаному діапазоні;
- **type = "date"** – поле для вибору календарної дати;
- **type = "datetime"** – дата і час;
- **type = "datetime-loca**l" – місцеві дата та час;
- **type = "time"** – для часу;
- **type = "month"** – вибір місяця;
- **type = "week"** – вибір тижня;
- **type = "url"** – для веб-адрес;

- **type = “email”** – для адреси електронної пошти, виглядає, як поле для введення тексту, але має параметри перевірки правильності вводу;
- **type = “search”** – поле для пошуку;
- **type = “tel”** – для телефонних номерів;
- **type = “image”** – створює графічну кнопку відправки форми, відображає картинку, що визначається атрибутом src або значення атрибута alt, якщо зображення відсутнє.

Наприклад.

Код	Результат
<form> <input type="text" placeholder="Введіть логін"/> <input type="password" placeholder="Введіть пароль" /> ☐ Текст </form>	

Поля зі списком (випадаючі списки) дозволяють користувачу вибрати одне значення з фіксованого списку значень, які представлені тегами <**option**> **…** </**option**>. За замовчуванням перший елемент відображається в рядку вибору**.** Синтаксис:

<select>
<option> пункт 1</option>
<option> пункт 2</option>
…
</select>

Атрибути тегу **<select> … </select>**:

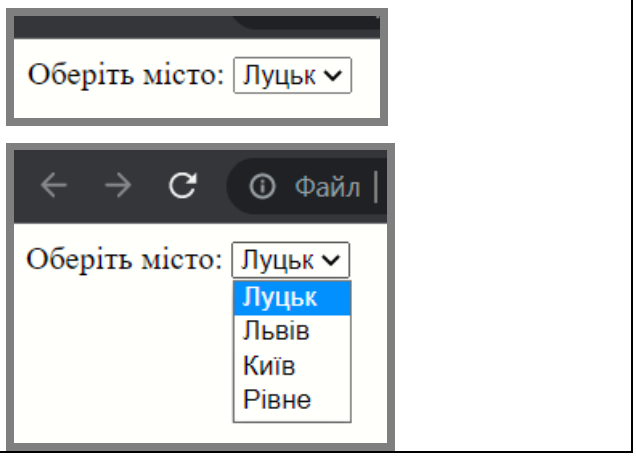
- **name** – задає ім’я списку;
- **size** – встановлює висоту меню в рядках, якщо є можливість вибору декількох елементів;
- **multiple** – вказує на можливість вибору декількох елементів меню.

<option> … </option> – використовується тільки з елементом <select> і описує окремі пункти меню. Атрибути:

- **selected** – визначає пункт списку, який буде вибраний першим при завантаженні документа;

- **value** – задає даному пункту унікальне ім’я, яке буде використовуватись разом із іншими відомостями про вміст заповненої форми.

Наприклад.

Код	Результат
<form> Оберіть місто: <select> <option>Луцьк</option> <option>Львів</option> <option>Київ</option> <option>Рівне</option> </select> </form>	

<optgroup> … **</optgroup>** дає змогу об’єднати групу елементів <**option> … </option>** із загальним надписом, яка, як правило, виділяється напівжирним шрифтом.

<textarea> … **</textarea>** – елемент, який використовується для того, щоб дозволити користувачу вводити більше одного рядка інформації (вільний текст).

Атрибути:

- **name** – ім’я поля введення;
- **rows** – висота поля введення в символах;
- **cols** – ширина поля введення в символах.

Якщо потрібно, щоб за замовчуванням в полі виводився якийсь текст, його необхідно вставити всередину тегів <**textarea**> і </**textarea**>.

<fieldset> … </fieldset> призначений для групування елементів форми, що візуально полегшує роботу з формами, які містять багато елементів

управління. Браузери, як правило, відображають використання цього тегу у вигляді рамки. Синтаксис:

<fieldset>
<legend> Текст **</legend>**
</fieldset>

<legend> … </legend> призначений для створення заголовка групи елементів форми, яка визначається з допомогою **<fieldset> … </fieldset>**.

<button> … **</button>** – створює стандартну кнопку, але пропонує розширені можливості в порівнянні з **<input type="button">**; обов'язково вказувати атрибут type.

<label> … </label> – мітка – зв'язує текст з відповідним елементом форми. Як правило використовується в наступних випадках:

- текст стає активним і при клікові по ньому змінюється значення пов'язаного з текстом перемикача або прапорця. Це зручно у використанні, бо непотрібно прицільно наводити вказівник миші на невеликий за розмірами елемент;

- стилі для **<label>** дозволяють задавати положення тексту та інші характеристики оформлення.

Існує два способи зв'язування об'єкта і мітки. Перший полягає у використанні ідентифікатора **id** всередині елемента форми і вказуванні його імені в якості атрибута for елемента **<label>**. Поля форми і текст для них можуть візуально знаходитися поряд, але в коді документа міститися всередині різних елементів. При другому способі елемент форми поміщається всередину контейнера **<label>**. Синтаксис:

1 спосіб
<input id="<ідентифікатор>">
<label for="<ідентифікатор>"> Текст </label>

2 спосіб
<label><input> Текст </label>

Запитання та завдання до теми

1. Яким тегом задають форму на вебсторінці?
2. Як відобразити рядок для введення на вебсторінці?
3. Як відобразити рядок для введення паролю на вебсторінці?
4. Як відобразити прапорець на вебсторінці?
5. Як відобразити перемикач на вебсторінці?
6. Як відобразити кнопку на вебсторінці?
7. Як відобразити випадаючий список на вебсторінці?
8. Як відобразити елемент для вибору локальних файлів на вебсторінці?
9. Як відобразити кнопку-зображення на вебсторінці?
10. Як відобразити область для введення багаторядкових коментарів на вебсторінці?
11. Які є типи випадаючих списків?

Завдання. Створити вебсторінку за поданим зразком (рис. 6).

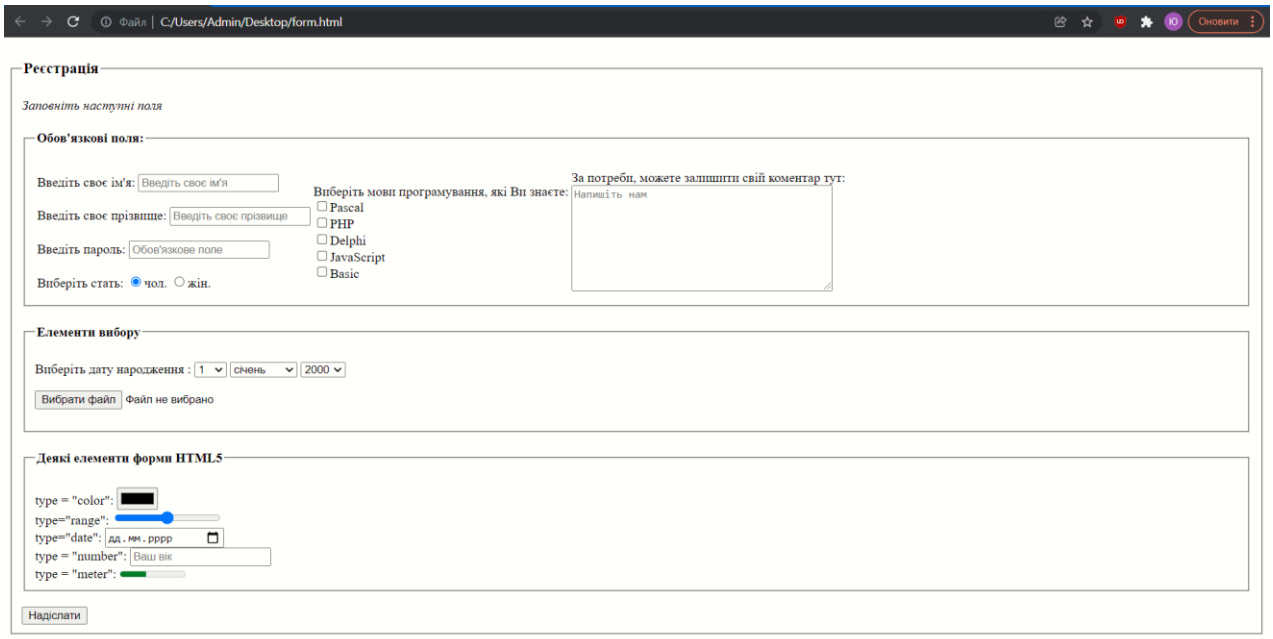


Рисунок 9 – Знімок екрану прикладу форми

Тема 6. Використання каскадних таблиць стилів CSS. Поняття CSS, селектора, типи селекторів. Синтаксис написання селекторів. Використання класів та ідентифікаторів. Наслідування та каскадування

1. Поняття CSS, їх призначення та переваги.
2. Поняття селектора, типи селекторів, синтаксис написання селекторів.
3. Способи та типи підключення каскадних таблиць стилів до вебсторінки.
4. Поняття та призначення класів CSS.
5. Синтаксис написання класів.
6. Синтаксис написання ідентифікаторів.
7. Механізм наслідування в CSS.

CSS – Cascading Style Sheets – технологія опису зовнішнього виду документа, який написаний мовою розмітки; це набір параметрів форматування, що застосовується до елементів документа, щоб змінити їх зовнішній вигляд.

Переваги стилів:

- розмежування коду і оформлення;
- різне оформлення для різних пристроїв;
- розширені в порівнянні з HTML способи оформлення елементів;
- прискорення завантаження сайту;
- єдине стильове оформлення великої кількості документів;
- централізоване зберігання.

Розрізняють декілька типів стилів, які можуть спільно застосовуватися до одного документа. Це стиль **браузера**, стиль **автора** і стиль **користувача**.

Стиль браузера – це оформлення, яке за замовчуванням застосовується до елементів вебсторінки браузером. Це оформлення можна побачити у випадку HTML-тексту, коли до документа не додається ніяких стилів.

Стиль автора – це стиль, який додає до документа його розробник.

Стиль користувача – це стиль, який може включити користувач сайту через налаштування браузера. Такий стиль має більш високий пріоритет і перевизначає вихідне оформлення документа.

Вказані типи стилів можуть існувати один з одним, якщо вони не намагаються змінити вид одного і того ж елемента. У разі виникнення суперечності спочатку має пріоритет **стиль користувача, потім стиль автора, далі – стиль браузера**.

Стильові правила записуються в форматі, відмінному від HTML. Основним поняттям виступає **селектор** – це деяке ім'я стилю, до якого додаються параметри форматування. В якості селектора можуть бути теги, класи і ідентифікатори. Синтаксис написання селекторів:

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

CSS не чутливі до регістра, перенесення рядків, пропусків і символів табуляції, тому форма запису залежить від бажання розробника.

Існує багато властивостей CSS, всі тут описати неможливо і недоречно, наведемо деякі з найпоширеніших:

color: колір тексту;

text-align: горизонтальне вирівнювання тексту;

background-color: встановлення фону елемента;

width: ширина елементу у пікселях або відсотках від сторінки;

height: висота елементу у пікселях або відсотках сторінки;

padding: внутрішній відступ елемента;

margin: зовнішній відступ елемента;

font-family: тип шрифту для тексту на сторінці;

font-size: кегль шрифту у пікселях;

border: властивість характеристик межі елемента.

Коли браузер зчитує таблицю стилів, він форматує документ згідно цієї таблиці. Існує три способи підключення таблиці стилів.

1. Підключення зовнішньої таблиці стилів (зв'язані стилі) застосовується у випадках, коли один стиль визначається для багатьох сторінок:

```
<head>
<link rel="stylesheet" type="text/css" href="first.css">
</head>
```

Зовнішню таблицю стилів можна створити в довільному текстовому редакторі. Файл зовнішньої таблиці стилів не може містити ніяких тегів HTML, немає заголовку і закінчення, як в HTML (див. рис. 10).

Розширення файла таблиці стилів – .css. Значення параметрів rel і type залишаються незмінними незалежно від коду.

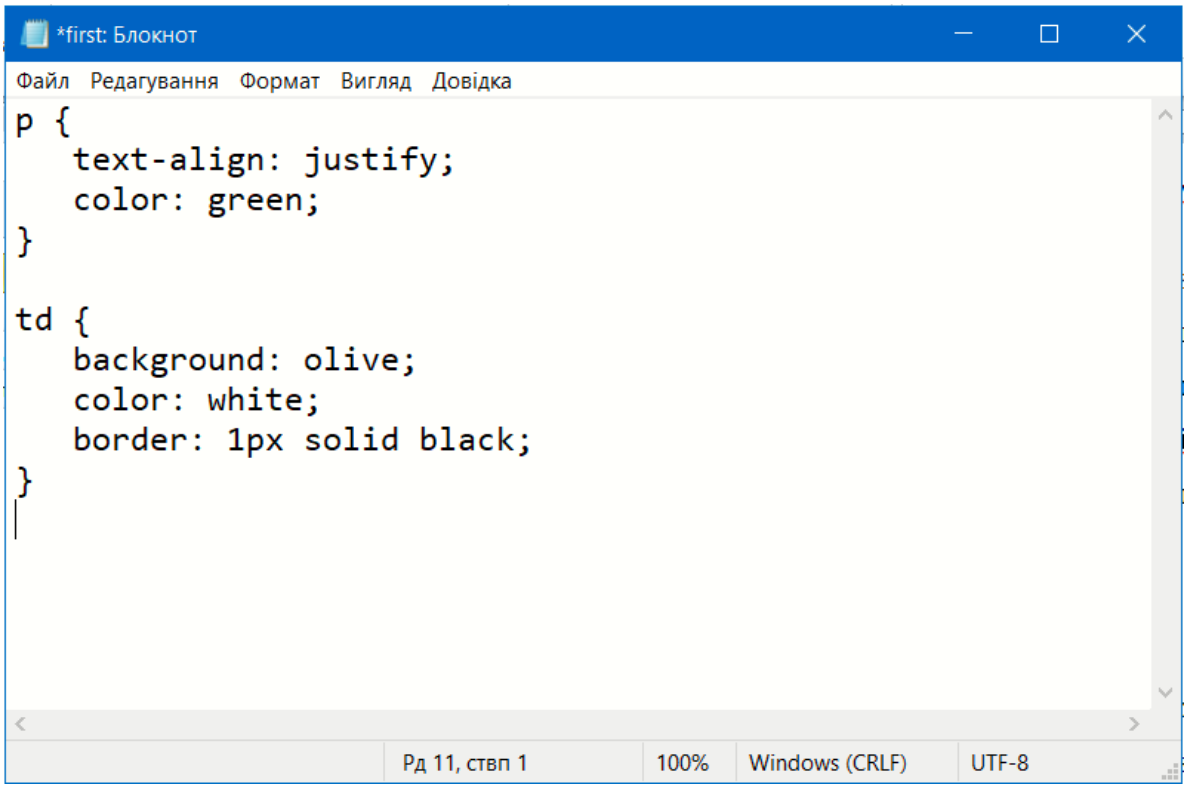


Рисунок 10 – Знімок екрана редактору Блокнот з фрагментом коду CSS

2. Підключення внутрішньої таблиці стилів (глобальні стилі) застосовується тоді, коли один документ має унікальний стиль. Для визначення внутрішніх стилів використовують тег **<style>…</style>**, який розміщається в розділі заголовка.

Синтаксис:

<style>

селектор1 {
властивість1: значення;
властивість2: значення;
... }
селектор2 {
властивість1: значення;
властивість2: значення;
... }
…

</style>

<style> застосовується для визначення стилів елементів вебсторінки. Записується всередині контейнера <head>. Можна задавати більше, ніж один тег <style>. Атрибути:

media – визначає пристрій виведення, для роботи з яким призначена таблиця стилів. Можливі значення: all, braille, handheld, print, screen, speech, projection, tty, tv;

type – повідомляє браузеру, який синтаксис використовувати, щоб правильно інтерпретувати стилі.

Наприклад.

```
<head>
  <style>
     li {font-family: arial}
     p {font-size: 25%}
     h1 {margin-right: 10pt}
  </style>
</head>
```

3. Вбудовані стилі (**внутрішні стилі**) використовуються в тому випадку, коли необхідно застосувати стиль для одного елемента. Недолік: він змішує вміст документа з його представленням і втрачає багато переваг таблиць стилів.

Наприклад.

```
<body style="background-color: blue">
  <h1 style="color: yellow" >Заголовок H1</h1>
</body>
```

Теоретично всі описані способи використання CSS можуть застосовуватися як окремо, так і сумісно один з одним. При цьому першим завжди застосовується внутрішній стиль, потім глобальний, а в останню чергу – зв'язаний. Але треба пам'ятати, що на практиці слід користуватися першим способом підключення каскадної таблиці стилів для успішної валідації сторінок сайту.

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.і'мя_класу {
    властивість1: значення;
    властивість2: значення;
... }
```

Всередині стилю спочатку пишеться потрібний тег, а потім, через крапку, призначене для користувача ім'я класу. Імена класів повинні починатися з латинського символу і можуть містити в собі символ дефіса (-) і підкреслення (_). Щоб вказати в коді HTML, що тег використовується з певним класом, до тегу додається параметр class="і'мя_класу".

Можна, також, використовувати класи і без вказівки тега. Синтаксис в цьому випадку буде наступний.

```
.ім'я_класу {
    властивість1: значення;
    властивість2: значення;
... }
```

При такому записі, клас можна застосовувати до будь-кого тегу.

Ідентифікатор (ID селектор) визначає унікальне ім'я елемента, яке використовується для зміни його стилю і звернення до нього через скрипти. Синтаксис:

```
#ім'я_ідентифікатора {
    властивість1: значення;
    властивість2: значення;
... }
```

Ім’я ідентифікатора повинне починатися з латинського символу і може містити в собі символ дефіса (-) і підкреслення (_). На відміну від класів ідентифікатори повинні бути унікальними, тобто зустрічатися в коді документа тільки один раз.

Звернення до ідентифікатора відбувається аналогічно класам, але як ключове слово у тезі використовується параметр id, значенням якого виступає ім’я ідентифікатора. Символ решітки при цьому вже не вказується.

Як і при використанні класів, ідентифікатори можна застосовувати до конкретного тегу. Синтаксис при цьому буде наступний:

тег#ім’я_ідентифікатора {
властивість1: значення;
властивість2: значення;
... }

Каскадування таблиці стилів означає, що порядок слідування правил в CSS має значення. Коли застосовані два правила CSS, що мають однакову специфічність, буде застосоване те, яке в CSS написане останнім.

Специфічність означає, яке правило буде застосоване в тому випадку, коли кілька правил мають різні селектори, але можуть бути застосованими до одного і того ж елемента. Різні типи селекторів (селектори тегів, селектори класів, селектори ідентифікаторів тощо) мають різний вплив на елементи сторінки. Чим більш загальний вплив здійснює селектор на елементи сторінки, тим менша його специфічність:

- селектор тегів менш специфічний (наприклад, h1 вибере всі заголовки на сторінці);
- селектор класу більше специфічний, бо він вибере лише ті елементи на вебсторінці, які мають конкретне значення атрибута class. Селектор класу застосовується після селектора елемента і тому перекриє його стилі.

Наслідування в CSS – це механізм, за допомогою якого значення властивостей елемента-батька передаються його елементам-нащадкам. Стилі, присвоєні деякому елементу, успадковуються всіма нащадками (вкладеними елементами), якщо вони не перевизначені явно. Наприклад, розмір шрифту і його колір часто застосовують до **<body>**, щоб всі елементи всередині цього

тегу мали ті ж властивості. Наслідування дозволяє скоротити розмір таблиці стилів, але якщо стилів багато, то відстежити, який батьківський елемент встановив певну властивість, стає складніше.

Не наслідуються наступні властивості: width, height, background-color, background-image, background-position, background-repeat, background, border-width, border-style, border-color, border, float, margin, padding.

Для примусового наслідування значень властивостей CSS від батьківського елемента використовується значення **inherit**. Це значення зазвичай застосовується для того, щоб властивості, які не успадковують значення, це робили.

Якщо в різних стилях до одного і того ж елемента були застосовані одні і ті ж властивості, то важливо визначити, яке буде остаточне значення цих властивостей.

Для цього потрібно враховувати специфічність селекторів та порядок правил (при однаковій специфічності).

Ключове слово !important грає роль в тому випадку, коли користувачі підключають свою власну таблицю стилів. Якщо виникає суперечність, коли стиль автора сторінки і користувача для одного і того ж елемента не збігається, то !important дозволяє підвищити пріоритет стилю.

При використанні користувацької таблиці стилів або одночасному застосуванні різного стилю автора і користувача до одного і того ж селектора, браузер керується наступним алгоритмом:

- якщо !important доданий в авторський стиль – буде застосовуватися стиль автора;
- якщо !important доданий в призначений для користувача стиль – буде застосовуватися стиль користувача.
- якщо !important немає як в авторському стилі, так і стилі користувача – буде застосовуватися стиль користувача.
- якщо !important міститься в авторському стилі і стилі користувача – буде застосовуватися стиль користувача.

Синтаксис:

властивість: значення !important

Запитання та завдання до теми

1. Що таке CSS?
2. Перерахувати переваги використання CSS.
3. Що таке селектор? Навести синтаксис написання селекторів.
4. Перерахувати типи каскадних таблиць стилів.
5. Як можна підключити каскадну таблицю стилів?
6. Що таке клас? Навести синтаксис написання класів.
7. Що таке ідентифікатор?
8. Навести синтаксис написання ідентифікаторів.
9. Що таке наслідування в CSS? Які властивості не наслідуються?

Завдання 1. Використовуючи засоби CSS для форматування тексту (підключити зовнішню таблицю стилів), створити сторінки за поданим зразком. Приблизний варіант сторінки наведено на рис. 11, 12. Текст для вебсторінки додається у текстовому файлі. На сторінках встановити однаковий стиль для заголовків двох різних рівнів (колір і шрифт); вирівнювання, колір шрифту та відступ першого рядка абзацу; фон сторінок.

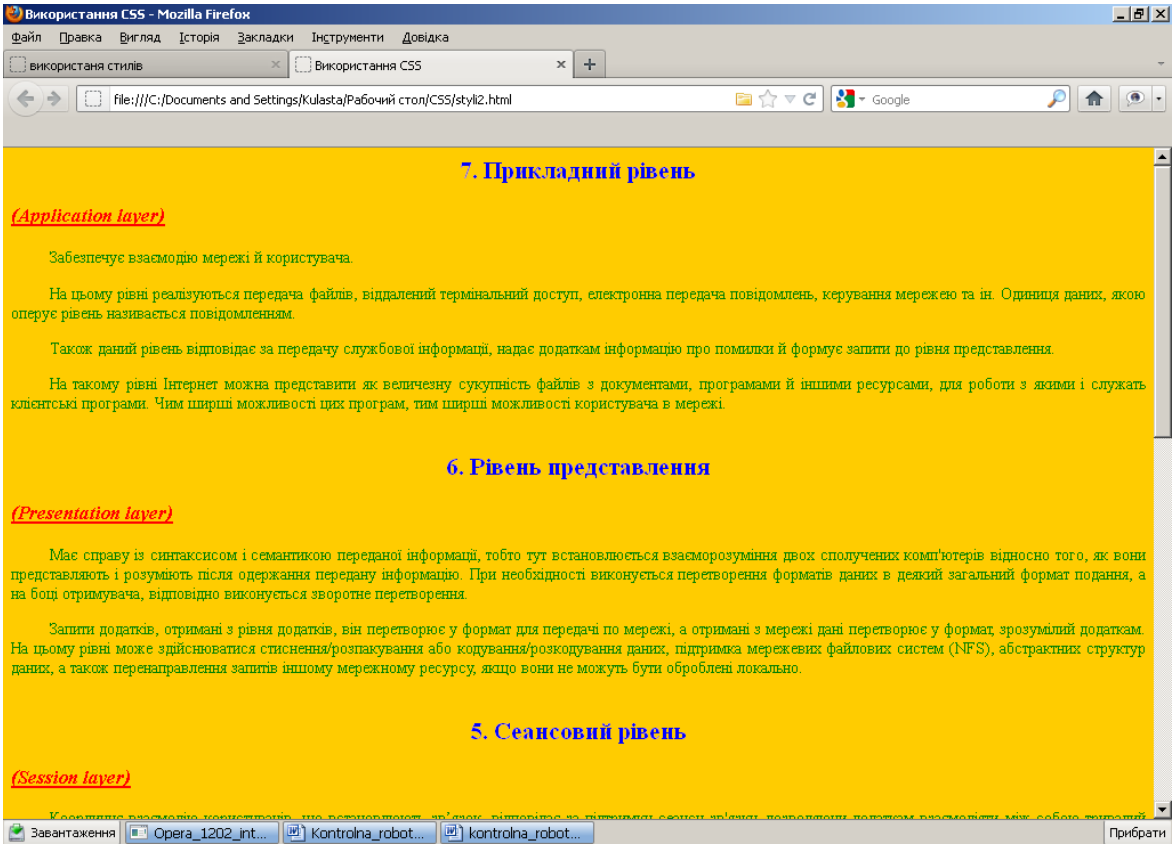


Рисунок 11 – Знімок екрану зразка вебсторінки «CSS»

Рисунок 12 – Знімок екрану зразка вебсторінки «CSS» (продовження)

Завдання 2. Використовуючи засоби CSS для форматування тексту та класи (підключити зовнішню таблицю стилів), створити сторінку за поданим зразком (рис. 13).

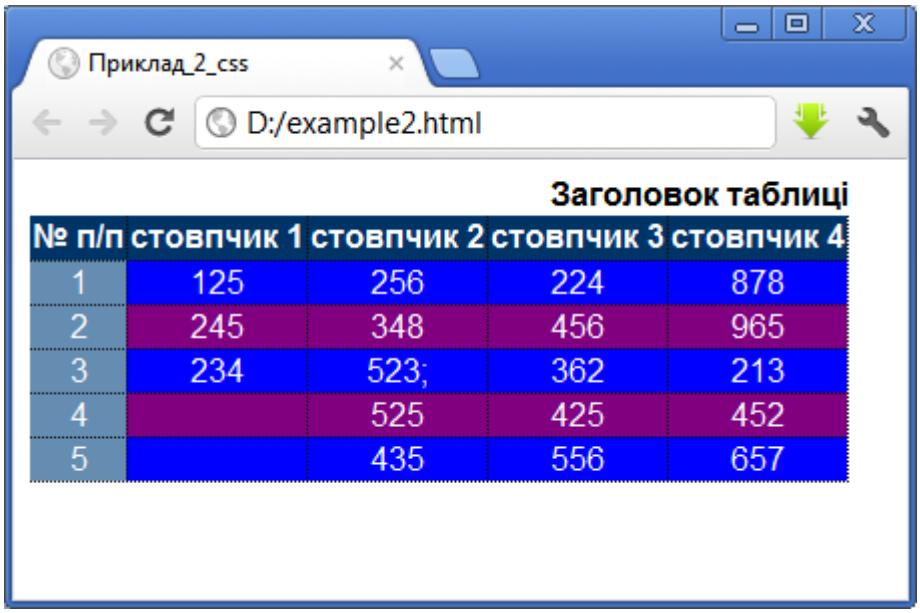


Рисунок 13 – Знімок екрану зразка вебсторінки «Класи»

Тема 7. Блочна структура вебсторінки. Особливості табличної та блочної верстоk. Семантична верстка HTML5. Позиціонування

1. Поняття та різниця між блочними та рядковими HTML-елементами.
2. Поняття верстки, типи верстоk.
3. Характеристика та особливості табличної верстки.
4. Характеристика та особливості блочної верстки.
5. Поняття стандарту HTML5.
6. Поняття семантичної верстки.
7. Семантичні елементи HTML5.
8. Блочні та текстові елементи HTML5.
9. Позиціонування елементів з допопомогою CSS.

В HTML елементи поділяються на блочні та рядкові.

Блочні елементи призначені для структурування осоновних частин вебсторінки шляхом розділення вмісту (контенту) на логічно пов'язані блоки (<p>; ; ; <h1>-<h6>; <article>; <section>; <blockquote> та ін.). Всі блочні елементи містять відкриваючі і закриваючі теги.

Рядкові елементи призначені для розмежування частини тексту і надання йому певної функції або змісту. Рядкові елементи, як правило, містять одне або кілька слів (<a>; <abbr>, <input> та ін.).

Блочні елементи можуть містити блочні або рядкові елементи. Рядкові елементи можуть містити тількі інші рядкові елементи (за виключенням елемента <a>).

Коли жоден семантичний елемент не підходить для вмісту або з метою групування чи стилізації, використовують один з двох загальних елементів:

** … ** – для блочних елементів;

** … ** – для рядкових елементів.

Верстка – це процес перетворення макета (малюнка) у вебсторінку. Критерії якості верстки:

- співпадання з дизайном;

- кросбраузерність (це властивість сайту однаково відображатися та функціонувати у відповідності до поставленого завдання в усіх браузерах);

- адаптація під різні роздільні здатності;

- чистий і валідний код.

Розрізняють верстку табличну, блочну, семантичну та адаптивну.

При табличній верстці для задання структури сайту використовується тег **<table>**. Таблична верстка раніше була основним способом верстки веб-сторінок.

Переваги та недоліки верстки з допомогою таблиць:

- таблиці не перекриваються одна з одною при малих роздільних здатностях (дозволяє найбільш пропорційно розташувати всі елементи сторінки відносно один одного);

- легко забезпечити кросбраузерність;

- код виходить громіздким (багато лишніх рядків і стовпців);

- не всі дизайни можна створити з допомогою таблиць;

- довге завантаження сторінки та погана індексація вмісту пошуковими системами. Вміст сторінки, яка використовує табличну верстку, не буде відображатися до того часу, поки не завантажаться всі дані.

Блочна верстка на відміну від табличної має ряд переваг:

- значно менше HTML-коду – менша вага сторінки;

- відокремлення стиля елементу від коду html;

- підходить для будь-якого дизайну;

- можливість накладання шарів;

- краща індексація пошуковими системами;

- висока швидкість завантаження сторінки, яка складається із взаємно незалежних елементів.

Кожен блочний елемент є прямокутною областю, яка має декілька властивостей. Основою блока є його контент (вміст) – текст, зображення, відео тощо, ширина якої задається властивістю **width**, а висота – властивістю **height**. Навкруги контенту розміщені поля (**padding**), що створюють порожній простір від вмісту до внутрішнього краю межі; далі слідують межі (**border**), а навколо

меж відступи (**margin**) – невидимий пустий простір від зовнішнього краю меж. Порядок цих властивостей є чітко визначеним і не може бути порушеним (рис. 14).

Рисунок 14 – Блочна модель елементу

Властивість **padding** задає відстань від внутрішнього краю межі до уявного прямокутника, що обмежує вміст блоку. Основне призначення цієї властивості – створити пустий простір навколо вмісту блока, що покращує читабельність та вигляд сторінки в цілому.

В якості значення не можна вказувати від’ємні числа (вони ігноруються браузером).

Властивість **border** додає навколо елемента рамку заданої товщини, стилю та кольору. Для створення ліній на окремих сторонах елемента використовуються властивості **border-top**, **border-right**, **border-bottom** і **border-left** відповідно. Також ці властивості можуть бути використані, щоб забрати лінію з певного боку блоку. В такому випадку задається значення **none**

але спочатку встановлюється рамка, а потім забираються лінії з потрібного боку.

Властивість **margin** як правило використовується для створення вертикальних і горизонтальних відступів між елементами. Властивість **margin** дозволяє вирівнювати елемент за центром горизонталі. Тоді треба встановити значення **auto** і задати ширину елемента через **width**.

Для **margin** можуть бути встановлені від'ємні значення, тоді елемент зміщається в протилежному напрямі.

Для деяких елементів браузер додає **margin** автоматично:

- для **<body>** за замовчуванням **margin: 8px;**
- для списків **<ul>**, **<ol>**, **<dl>** **margin-top** і **margin-bottom** задані по **1em**;
- для **<blockquote>** і **<figure>** **margin-left** і **margin-right** задані по **40px**.

Такі відступи, згідно макету, не завжди потрібні, тому їх можна обнулити (**margin: 0**).

Вертикальні **margin** у сусідніх елементів об'єднуються між собою – схлопуються. Саме схлопування діє на два сусідні блоки або батьківський і дочірній блоки. При цьому відступи, що знаходяться поряд, не сумуються, а комбінуються в один. Для **margin** зліва і справа схлопування ніколи не відбувається.

Схлопування задумувалось в першу чергу для коректного відображення тексту в абзацах. Відстань між абзацами **<p>** без схлопування збільшиться в два рази, тоді як верхній margin в першому абзаці і нижній margin у останньому абзаці залишаться незмінними. Схлопування гарантує, що відстань в абзацах кругом буде одинаковою.

Якщо обидва значення **margin** додатні, то з них вибирається найбільше і воно встановлюється як відстань між блоками.

Якщо одне із значень **margin** від'ємне, то вони сумуються (додаються) за правилами математики. Якщо отримане значення виявиться від'ємним, то воно буде діяти на нижній блок і, відповідно, він зсунеться вверх на вказане значення. При цьому можливе накладання одного блоку на інший.

Якщо обидва значення **margin** від’ємні, то з двох значень вибирається більше за модулем, і воно ж буде від’ємним відступом між елементами.

Для пустих елементів, всередині яких немає ніякого вмісту, **margin-top** і **margin-bottom** також комбінуються в один за тими ж правилами, що й для сусідніх блоків. При цьому потрібне виконання наступних умов:

- для елемента не повинен бути заданий **padding** зверху або знизу;
- для елемента не повинен бути заданий **border** зверху або знизу;
- висота елемента не повинна бути вказана через властивість **height** або **min-height**.

Розміри блоку – це комплексна величина, що складається із значень різних властивостей.

Ширина блоку є сумою значень наступних властивостей (рис. 15):

margin-left + border-left + padding-left + width +
+ padding-right + border-right + margin-right.

Рисунок 15 – Розмір блочного елемента по горизонталі

Висота блоку є сумою значень наступних властивостей (рис. 16):

margin-top + border-top + padding-top + height +

+ padding-bottom + border-bottom + margin-bottom.

Якщо властивість **height** не вказана, то вона вважається **auto**, і в такому випадку висота вмісту обчислюється автоматично.

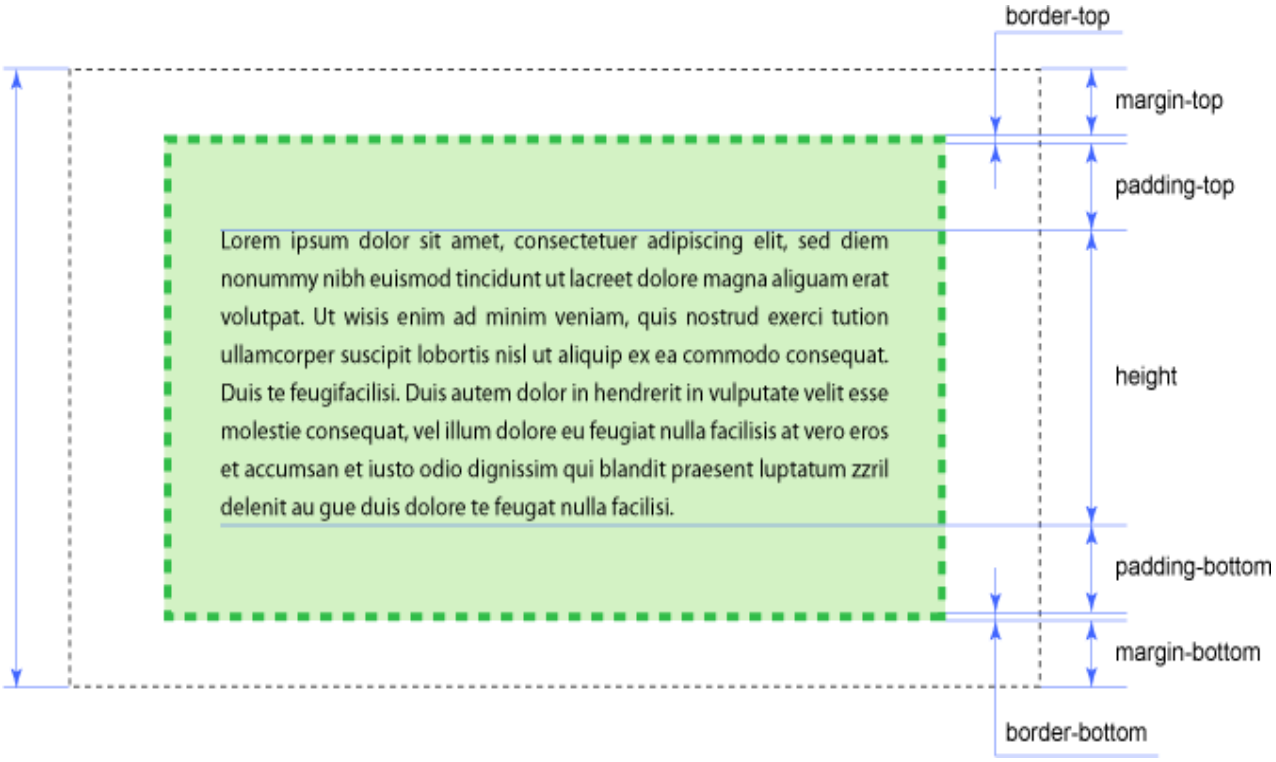


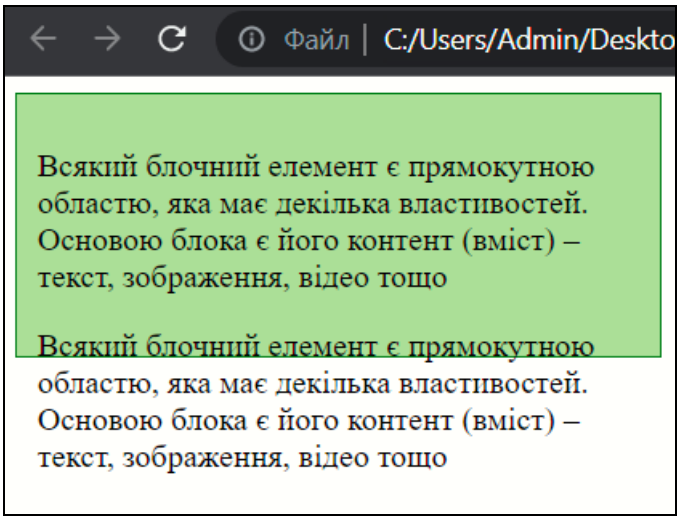
Рисунок 16 – Розмір блочного елемента по вертикалі

Якщо вміст перевищує розмір блоку при заданому **height**, то вміст відображається зверху блоку. Щоб уникнути такого властивість **height** краще не задавати, тоді висота блоку буде обчислюватися автоматично. Але коли є необхідність чітко вказати висоту блоку, рекомендується додати властивість **overflow**, значення **auto** якої встановлює полоси прокрутки (за необхідимості), а значення **hidden** приховує все, що не вмістилося в задані розміри.

Наприклад.

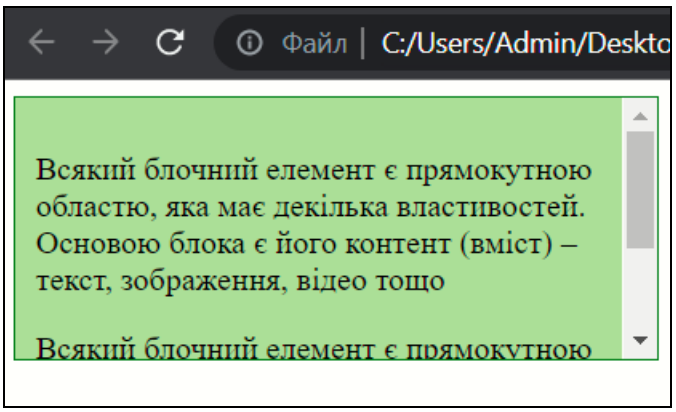
Код	Результат

```
// Випадок, коли текст
займає більше місця, ніж
задана висота блоку

<style>
  .block {
   background: #C5DF94;
   border: 1px solid green;
   height: 110px;
   padding: 10px;
   width: 300px;
  }
</style>
```



Код	Результат

```
// Те саме, але додана
полоса прокрутки

<style>
  .block {
   background: #C5DF94;
   border: 1px solid green;
   height: 110px;
   padding: 10px;
   width: 300px;
   overflow: auto;
  }
</style>
```



HTML5 – це нова версія мови HTML, прийнята як рекомендований стандарт 28 жовтня 2014 року W3C.

Деякі можливості **HTML5:**

- підтримка геолокації;
- відтворення відео та аудіо;
- нові елементи форм;
- можливість малювання на вебсторінці та ін.

В **HTML5** з’явилися ряд нових елементів та тегів, які забезпечують семантичну верстку веб-сторінок.

Семантична верстка – це підхід до створення веб-сторінок мовою HTML, який базується на використанні HTML-тегів відповідно до їх семантики

(призначення), та забезпечує логічну та послідовну ієрархію вебсторінки. Тобто назви HTML-тегів повинні співпадати з їх смисловим значенням.

При семантичній верстці чітко повинні виділятися заголовки різних рівнів, списки, таблиці, малюнки, абзаци та ін. Таким чином семантична верстка дає можливість "бачити" вебсторінку, читаючи HTML-код, швидше вносити зміни в код, економити час.

Семантичний HTML-код чистіший і менший за об’ємом, легше читається і простіше шукати помилку.

Нові структурні елементи HTML5:

<section> … </section> – визначає секції (розділи). Використовується для групування логічно пов’язаного вмісту вебсторінки. Використовується для опису певного блоку тексту;

<header> … </header> – позначає шапку сайту або розділу (верхня частина сторінки);

<footer> … <footer> – позначає підвал сайту або розділу (нижня частина сторінки);

<nav> … </nav> – позначає набір гіперпосилань на інші сторінки (використовується для створення навігації по сайту);

<article> … <article> – використовується для позначення тексту новини, статті та ін.

Блочні елементи HTML5:

<aside> … </aside> – визначає блок збоку від контенту для розміщення рубрик, посилань на архів, міток та ін. Такий блок, як правило, називається "сайдбар" або "бічна панель";

<dialog> … </dialog> – призначений для створення діалогу між співрозмовниками;

<figure> … </figure> – дозволяє згрупувати зображення разом із підписом до нього. Разом використовується тег **<figcaption>** … **</figcaption>**, який містить підпис зображення. **<figcaption>** … **</figcaption>** повинен бути першим або останнім тегом в групі.

Елементи текстового рівня:

mark (m) – позначає текст, як виділений. Такий текст нічим не відрізняється від іншого тексту, але його вигляд може бути змінений з допомогою стилів.

time – позначає текст всередині як дата, час або обидва значення.

Порядок відображення елементів на сторінці називається **потоком документа**. Блочний елемент займає 100% ширини батьківського елементу. Тому, блочні елементи відображаються один під іншим відповідно до розмітки сторінки. Рядковий елемент займає ширину, яка відповідає ширині вмісту всередині нього. Тому, рядкові елементи відображаються поруч один з одним.

За допомогою CSS можна порушити звичайний порядок відображення елементів на сторінці, використавши властивість **position**, яка дозволяє встановити один із трьох типів позиціювання елементів: **статичне**, **відносне** та **абсолютне**. При позиціонуванні елементів можна використовувати як додатні, так і від'ємні значення.

Нормальне позиціонування. Якщо для елемента властивість **position** не задана або має значення **static**, то елемент виводиться в потоці документа в порядку слідування елементів в коді HTML. Властивості **left**, **top**, **right**, **bottom**, якщо вони визначені, ігноруються.

При абсолютному позиціонуванні (**position: absolute**) елемент не існує в потоці документа і його положення задається відносно країв вікна браузера. Крім абсолютного позиціонування для елементів слід задавати **overflow: auto** – при перевищенні контентом висоти видимої області з'являється полоса прокрутки.

Фіксоване положення (**position: fixed**). Елемент прив'язується до вказаної властивостями **left**, **top**, **right** і **bottom** точки на екрані та не змінює свого положення при прокручуванні вебсторінки. При виході фіксованого шару за межі видимої області праворуч або знизу від неї не виникає полос прокручування. Застосовується для створення меню, вкладок та ін.

Відносне позиціонування (position: relative) – положення елемента встановлюється відносно його початкового положення додаванням властивостей **left**, **top**, **right** і **bottom**.

Позиціоновані елементи можна нашаровувати один на одного з допомогою CSS-властивості **z-index**, імітуючи таким чином третій вимір (вісь Z, яка перпендикулярна до екрану).

Запитання та завдання до теми

1. Що таке блочний елемент та яким тегом його задати?
2. Назвати відомі вам блочні елементи.
3. Що таке рядковий елемент та яким тегом його задати?
4. Назвати відомі вам рядкові елементи.
5. Що таке верстка?
6. Перерахувати критерії якості верстки.
7. Що таке блочна верстка, її характеристики.
8. Що таке таблична верстка, її характеристики.
9. Що таке HTML5?
10. Які нові можливості присутні в HTML5?
11. Що таке семантична верстка?
12. Перерахувати нові структурні елементи HTML5.
13. Перерахувати нові блочні елементи HTML5.
14. Перерахувати нові рядкові елементи HTML5.

Завдання 1. Створити блоки з текстом за поданим зразком (рис. 17). Звернути увагу на кольорове оформлення, відступи.

Основним елементом, який застосовується при блочній верстці є тег ** **. Фрагмент коду, який позначений цим тегом, називається шаром. З допомогою стилів блоки позиціонуються на сторінці, формуючи шаблон, який потім наповнюється контентом.

Верстка – це процес перетворення малюнка у вебсторінку.

> Критерії якості верстки:
> співпадання з дизайном;
> кросбраузерність;
> адаптація під різні роздільні здатності;
> чистий і валідний код.

Розрізняють верстку табличну, блочну та семантичну.

Поля (**padding**) – це **відстань** від внутрішнього краю межі або краю блоку до уявного прямокутника, який обмежує вміст блоку. Значення полів не можуть бути від'ємними.

Межі (**border**) – це лінії кругом полів елемента на одній, двох, трьох або чотирьох його сторонах. В кожної лінії є товщина, стиль та колір.

Відступ (**margin**) – це порожній простір від зовнішнього краю межі, полів або вмісту блоку. **Межі і поля не є обов'язковими і можуть бути відсутніми.**

Рисунок 17 – Зразки блочних елементів

Завдання 2. Використовуючи властивості заокруглення кутів блоку, тіні для тексту, блочні та рядкові елементи, створити вебсторінку за зразком (рис.18).

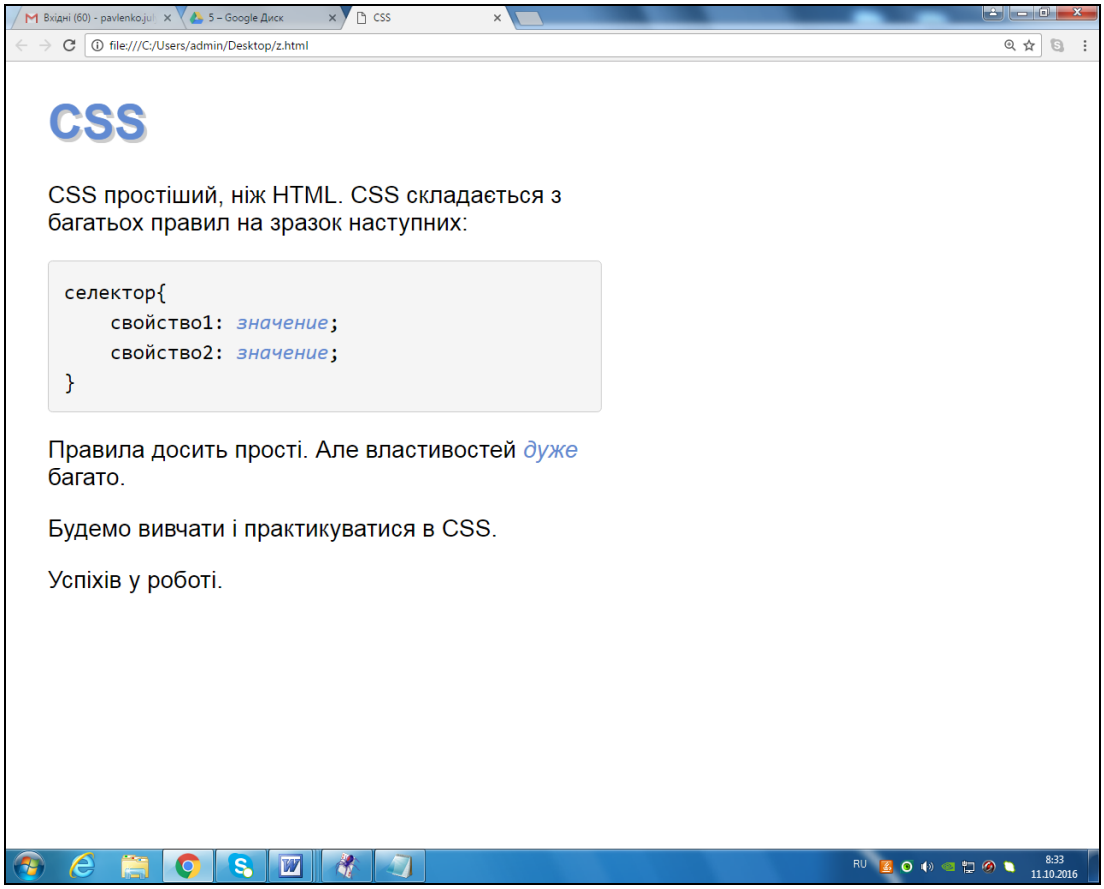


Рисунок 18 – Зразок вебсторінки

Завдання. З допомогою HTML5 та CSS зверстати наступний макет (див. рис. 19).

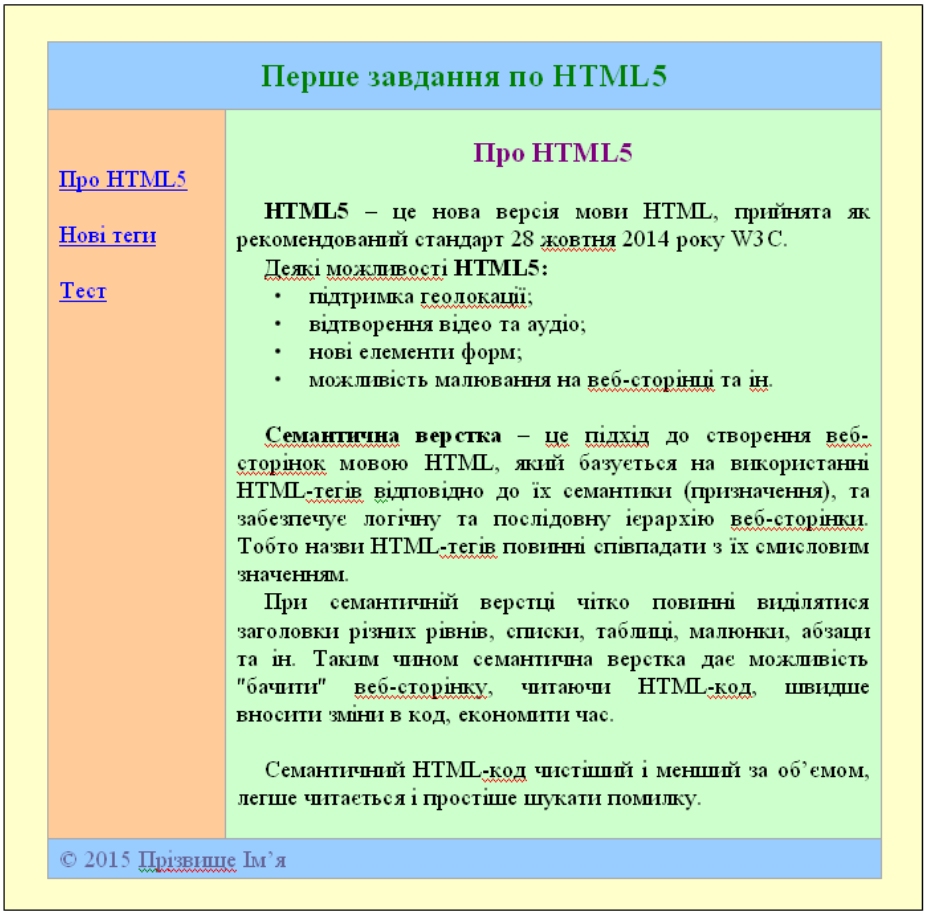


Рисунок 19 – Приклад верстки

Тема 8. Спеціальні селектори. Псевдоелементи, псевдокласи CSS та особливості їх використання при верстці вебсторінок

1. Поняття контекстних селекторів, синтаксис.
2. Поняття сусідніх селекторів, синтаксис.
3. Поняття дочірніх селекторів, синтаксис.
4. Поняття псевдоелементу CSS, синтаксис.
5. Поняття псевдокласу CSS, синтаксис.
6. Класифікація псевдокласів.

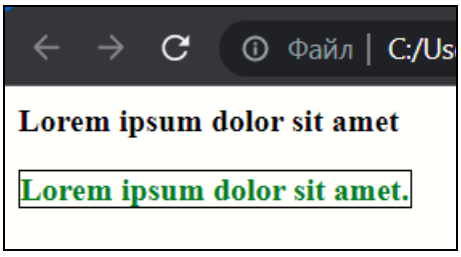
Контекстні селектори. При створенні вебсторінок часто доводиться вкладати одні теги в інші. Для коректного використання стилів цих тегів застосовують селектори, які працюють тільки в певному контексті. Таким чином можна встановити стиль для окремого тегу, а також для тегу, який знаходиться всередині іншого. Такі селектори називають контекстними. Вони складаються з кількох частин, розділених пробілом. Синтаксис:

селектор1 селектор2 {правила стилю}

В цьому випадку стиль буде застосовуватись до **селектора2** тоді, коли він розміщений всередині **селектора1**.

Наприклад:

Код	Результат

<style>
p b {
border: 1px solid black;
color: green;
}
</style> </head>
<body>
 Lorem ipsum dolor sit amet
 	
 Lorem ipsum dolor sit amet.
 	
</body>
	

Сусідні селектори використовуються для розташованих поряд елементів.

Наприклад, теги <li> в списку є сусідніми по відношенню один до одного і вкладеними в тег <ul>.

Сусідні селектори записуються з допомогою знаку +. Синтаксис:

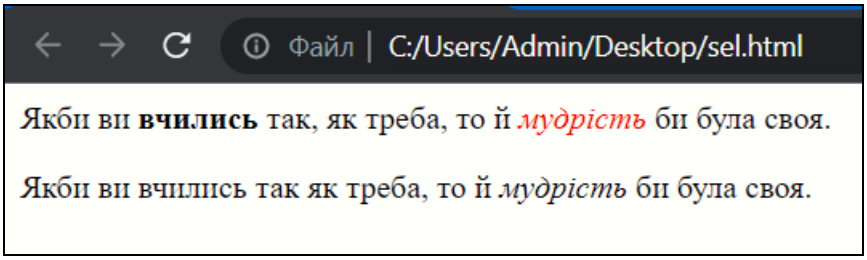
селектор1 + селектор2 {правила стилю}

Стиль при такому запису застосовується до **селектора2**, але тільки в тому випадку, якщо він є сусіднім для **селектора1** і слідує відразу після нього.

Код

```
<style>
  b + i {
    color: red;
  }
</style>
</head>
<body>
  <p>Якби ви <b>вчились </b> так, як треба, то й
<i>мудрість</i> би була своя.</p>
  <p>Якби ви вчились так як треба, то й <i>мудрість</i> би
була своя.</p>
</body>
```

Результат



Дочірні селектори. Контекстні селектори впливають на всіх нащадків, що не завжди є зручним. Іноді необхідно задати стилі лише для дочірніх елементів. Особливо корисно це при роботі з багаторівневими списками. Для цього використовують дочірні селектори. Синтаксис:

селектор 1 > селектор 2 {правила стилю}

Стиль застосується до **селектора 2**, але тільки в тому випадку, якщо він є дочірнім для **селектора 1**.

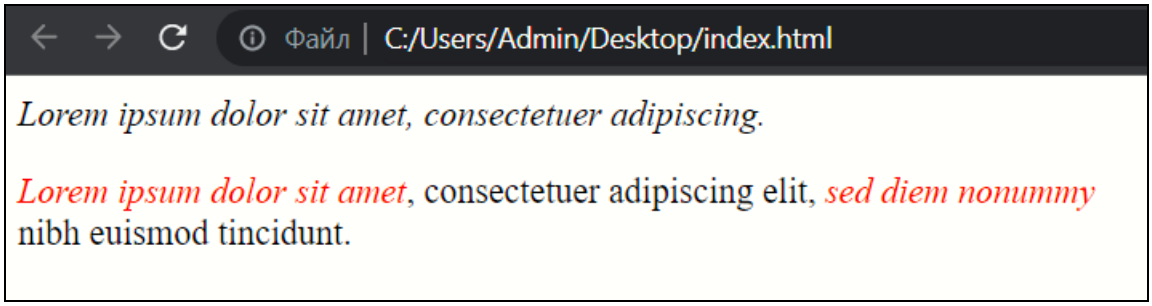
Наприклад.

Код

```
<style>
       P > I {
    color: red;
   }
```

```
  </style>
 </head>
 <body>
  <div>
   <i>Lorem ipsum dolor sit amet, consectetuer adipiscing.</i>
   <p><i>Lorem ipsum dolor sit amet</i>, consectetuer
adipiscing
   elit, <i>sed diem nonummy</i> nibh euismod tincidunt.</p>
  </div>
</body>
```

Результат



Споріднені селектори схожі на сусідні селектори, але, на відміну від них, стильові правила застосовуються до всіх елементів, що розташовані поряд. Наприклад, для селектора h1~p стиль буде застосовуватись до всіх елементів <p>, які знаходяться після заголовка <h1>. При цьому <h1> і <p> повинні мати спільного батьківського елемента. Тому якщо <p> помістити всередину , то стилі застосовуватися вже не будуть. Синтаксис:

E ~ F { опис правил стилю }

Стиль застосовується до елемента F в тому випадку, якщо він має того самого батьківського елемента, що й елемент E, і слідує після нього. Наприклад.

```
h1 ~ p { color: red; }
<h1>Заголовок</h1>
<p>Абзац 1</p>
<p>Абзац 2</p>
```

(червоний колір тексту буде встановлений для всіх абзаців)

```
h1 ~ p { color: red; }
<h1>Заголовок</h1>
<p>Абзац 1</p>
 <div><p>Абзац 2</p></div>
<p> Абзац 3</p>
```

Тут червоний колір тексту буде встановлений для 1-го і 3-го абзаців. До другого не буде, бо <h1> і <p> не мають спільного батьківського елемента.

Псевдоелементи дозволяють задати стиль елементів, не визначених в дереві елементів документа, а також генерувати вміст, якого немає у вихідному тексті. Синтаксис:

селектор::псевдоелемент {правила стилю}

Спочатку пишуть ім'я селектора, потім двокрапка, далі ім'я псевдоелемента. Кожен псевдоелемент може застосовуватись тільки до одного селектора. Якщо потрібно відразу встановити кілька псевдоелементів для одного селектора, правила стилю повинні додаватися до них окремо:

Псевдоелементи не можуть застосовуватись до внутрішніх стилів, тільки до таблиці зв'язаних або глобальних стилів. Наведемо кілька псевдоелементів:

::after застосовується для вставки призначеного контенту після вмісту елемента. Стильова властивість content визначає вміст для вставки;

::before – вставляє контент до вмісту елемента;

::first-letter – визначає стиль першого символа в тексті елемента, до якого додається (задає буквицю);

::first-line – визначає стиль першого рядка блочного тексту. До нього можна застосовувати не всі стильові властивості. Можна: властивості, що відносяться до шрифта, зміни кольору тексту і фону, а також clear, line-height,letter-spacing, text-decoration, text-transform, vertical-align і word-spacing.

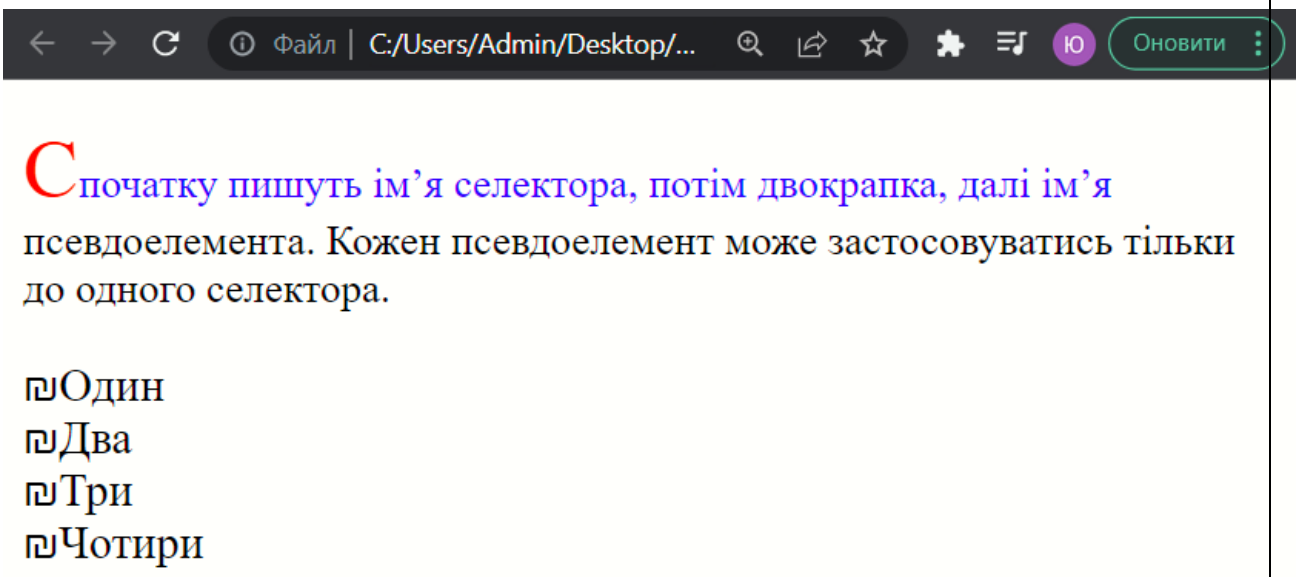
Наприклад.

Код

```
<head>
  <style>
   ul {
      padding-left: 0;
      list-style-type: none; }
   li::before {
      content: "\20aa ";  }
   p {
      font-size: 90%;
      color: black; }
   p::first-letter {
      font-family: 'Times New Roman', Times, serif;
      font-size: 200%;
      color: red; }
```

```
   p::first-line{
      color: blue; }
  </style>
</head>
<body>
<p> Спочатку пишуть ім’я селектора, потім двокрапка, далі ім’я
псевдоелемента. Кожен псевдоелемент може застосовуватись тільки
до одного селектора. Якщо потрібно відразу встановити кілька
псевдоелементів для одного селектора, правила стилю повинні
додаватися до них окремо:
</p>
<p>Спочатку пишуть ім’я селектора, потім двокрапка, далі ім’я
псевдоелемента. Кожен псевдоелемент може застосовуватись тільки
до одного селектора. Якщо потрібно відразу встановити кілька
псевдоелементів для одного селектора, правила стилю повинні
додаватися до них окремо:
</p>
<ul>
   <li>Один</li>
   <li>Два</li>
   <li>Три</li>
   <li>Чотири</li>
</ul>
</body>
```

Результат



Псевдокласи визначають динамічний стан елементів, який змінюється з допомогою дій користувача, а також положення в дереві документа.

Прикладом такого стану є гіперпосилання, яке змінює свій колір при наведенні на нього курсора миші. При використанні псевдокласів браузер не перезавантажує поточний документ. Тому з їх допомогою можна отримати динамічні ефекти на сторінці. Синтаксис:

селектор:псевдоклас {правила стилю}

Спочатку вказують селектор, до якого додається псевдоклас, а далі ім’я псевдокласу. Допускається застосовувати псевдокласи до імен ідентифікаторів або класів, до контекстних селекторів. Якщо псевдоклас вказується без селектора спереду, то він буде застосовуватись для всіх елементів документа.

Умовно всі псевдокласи діляться на три групи:

- ті, які визначають стан елементів;
- ті, які відносяться до дерева елементів;
- ті, які вказують на мову тексту.

Псевдокласи, які визначають стан елементів – такі, які розпізнають поточний стан елемента і застосовують стиль тільки для цього стану:

:active – відбувається при активації користувачем елемента (наведення вказівника миші на елемент і клацання). Але використовується переважно для гіперпосилань;

:link – вибирає невідвідані гіперпосилання;

:focus – застосовується до елемента при отриманні ним фокусу;

:hover – активізується, коли вказівник миші знаходиться в межах елемента, але клік мишею по ньому не відбувається;

:visited – застосовується до відвіданих гіперпосилань. Звичайно таке гіперпосилання змінює свій колір за замовчуванням на фіолетовий, але з допомогою стилів колір та інші параметри можна задавати самостійно.

Псевдокласи, які відносяться до дерева елементів;

:first-child – застосовується до першого дочірнього елемента селектора, який розташований в дереві елементів документа. Найзручніше використовувати в тих випадках, коли потрібно задати різні стилі для першого та інших однотипних елементів;

:first-of-type (:last-of-type) – задає правила стилів для першого (останнього) елемента в списку дочірніх елементів свого батьківського елемента;

:nth-child – використовується для додавання стилю до елементів на основі нумерації в дереві елементів. Відлік ведеться від першого елемента. Синтаксис:

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

- **odd** – всі непарні номери елементів;
- **even** – всі парні номери;
- **<число>** – порядковий номер елемента відносно свого батьківського елемента. Нумерація починається з 1;
- **<вираз>** – задається у вигляді $an + b$, де a і b – цілі числа, а n – автоматично приймає значення 0, 1, 2, ...

:nth-last-child – використовується для додавання стилю до елементів на основі нумерації в дереві елементів. Але відлік ведеться від останнього елемента;

:nth-last-of-type – використовується для додавання стилю до елементів вказаного типу на основі нумерації в дереві. Нумерація починається з кінця;

:nth-of-type – використовується для додавання стилю до елементів вказаного типу на основі нумерації в дереві. Нумерація починається з початку.

Атрибути, які вказують на мову тексту. Для документів, які одночасно містять текст на декількох мовах, має значення дотримання правил синтаксису, характерних для тієї чи іншої мови. З допомогою псевдокласів можна змінювати стиль оформлення іншомовних текстів, а також деякі налаштування.

:lang – визначає мову, яка використовується в документі або його фрагменті.

Запитання та завдання до теми

1. Що таке контекстні селектори, їх синтаксис.
2. Що таке сусідні селектори, їх синтаксис.
3. Що таке дочірні селектори, їх синтаксис.
4. Що таке псевдоелемент?
5. Що таке псевдоклас?
6. Навести синтаксис написання псевдоелементів.
7. Навести синтаксис написання псевдокласів.
8. Перерахувати псевдоелементи, які Ви знаєте.
9. Як поділяються псевдокласи?

10. Перерахувати псевдокласи, які визначають стан елементів.

11. Перерахувати псевдокласи, які відносяться до дерева елементів.

12. Перерахувати псевдокласи, які вказують на мову тексту.

Завдання.

1. В абзаці зробити першу букву більшою і кольоровою з допомогою псевдоелементу **:first-letter**.

2. Надати першому рядку абзацу іншого форматування (розмір, колір, шрифт) з допомогою псевдоелементу **:first-line**.

3. Є кілька абзаців тексту. З допомогою **:first-child** забезпечити інше відображення першого абзацу (виділення напівжирним шрифтом).

4. При наведенні вказівника миші на рядок таблиці, рядок змінює колір/товщину рамки (псевдоклас **:hover**).

5. Зробити гіперпосилання спочатку синього кольору, в момент натискання кнопки миші – зеленими, відвідані – помаранчеві.

6. Зробити таблицю зеброю з різними псевдокласами (без додаткових класів).

7. Зробити гіперпосилання кнопкою: забрати підкреслення, задати колір фону, рамку. При наведенні вказівника миші, змінюється заливка і т.д. і т.п.

8. Зробити форму з двох елементів: поле для введення і перемикач (checkbox). При отриманні курсору поле для введення повинне підсвітитися червоною рамкою. При виборі checkbox, його значення замальовується жовтим кольором.

9. Створити форму, в якій обов’язкові поля відмічені хрестиком, а необов’язкові – галочкою. При правильному заповненні обов’язкових полів хрестик повинен змінюватися на галочку.

Тема 9. Реалізація переходів та анімацій на CSS3. Google-шрифти. CSS-фігури

1. Поняття переходу та спосіб його реалізації.
2. Властивості переходів.
3. Поняття анімації та реалізація анімацій в CSS3.
4. Властивості анімації.
5. Робота з GoogleFonts.
6. Створення фігур з допомогою CSS.

Переходи в CSS дозволяють плавно перейти від одного стану елемента до іншого (окремі властивості анімуються від початкового стану до кінцевого); це специфічний вид анімації, де є тільки початковий та кінцевий стан.

При переходах можна визначити наступні властивості:

- **transition-property:** які властивості анімуються. Дозволяє реалізувати анімацію для вибраних окремих властивостей. Лише третя частина CSS-властивостей може бути анімована. За замовчуванням **transition-property: all**, тобто будуть анімуватися всі можливі властивості;

- **transition-duration:** як довго триває анімація. Може бути встановлене в секундах (s) або в мілісекундах (ms);

- **transition-delay:** визначає, як довго перехід повинен чекати перед початком. Для задання використовують секунди (s) або мілісекунди (ms);

- **transition-timing-function:** функція часу, що визначає, як обчислюється значення кожної властивості під час переходу. За замовчуванням **transition-timing-function: ease** (прискорення спочатку і сповільнення в кінці). Інші значення властивості:

  • **linear:** постійна швидкість;

  • **ease-in:** повільний початок, швидке завершення;

  • **ease-out:** швидкий початок, повільне завершення;

  • **ease-in-out:** схоже на ease, але з більш вираженим прискоренням/сповільненням;

• **cubic-bezier:** якщо ці функції часу не влаштовують, можна написати свою власну, використовуючи криві Без'є.

Можна використовувати скорочену версію властивості **transition:**. В цьому випадку, тільки **transition-duration** є **обов'язковим**.

Анімація в CSS

Для задання анімації можна використовувати скорочений варіант властивості **animation**:, де перераховувати через пробіл потрібні значення, або прописувати кожну окрему властивість, яка впливає на перебіг анімації:

- **animation-name**: назва анімації;
- **animation-duration**: як довго триває анімація;
- **animation-timing-function**: як обчислюються проміжні стани;
- **animation-delay**: анімація починається через деякий час;
- **animation-iteration-count**: скільки разів повинна виконатися анімація;
- **animation-direction**: повинен виконуватися рух в зворотній бік чи ні;
- **animation-fill-mode**: які стилі застосовуються до початку анімації і після її завершення.

Перед застосуванням анімації до HTML-елементів треба **написати анімацію з допомогою ключових кадрів**, використовуючи правило **@keyframes**. Ключові кадри – це кожен проміжний крок в анімації. Вони визначаються з допомогою відсотків. Наприклад:

- 0% – перший крок анімацій;
- 50% – крок на половині анімації;
- 100% – останній крок.

Можна використовувати ключові слова **from** і **to** замість 0% і 100%, відповідно. Кількість ключових кадрів можна визначати довільно, навіть в дробовому форматі. Кожен ключовий кадр – це правило CSS.

Для того, щоб задати анімацію слід написати:

@keyframes назва { ... }

animation-name – назва анімації використовується щонайменше 2 рази:

- при написанні анімації з допомогою **@keframes**;

- при використанні анімації з допомогою властивості **animation-name** (або **animation**).

Назва анімації не може починатися з цифри або двох дефісів та може включати наступні символи:

- букви (a-z);
- цифри (0-9);
- підкреслення (_);
- дефіс (-).

animation-duration – тривалість анімації встановлюється в секундах або мілісекундах. Значення за замовчуванням 0s.

animation-timing-function – функції часу для анімації можуть використовувати ключові слова, як в переходах, а можуть бути визначені з допомогою довільних кривих Без'є. Значення за замовчуванням: **ease**.

animation-delay – затримка анімації встановлюється в **секундах** (1s) або **мілісекундах** (200ms). За замовчуванням 0s.

Корисно використовувати, коли включається кілька анімацій в серії.

animation-iteration-count

За замовчуванням анімація відтворюється тільки один раз (значення 1). Для забезпечення циклічності використовують три типи значень:

- цілі числа – стільки разів відбудеться повторення анімації;
- дробові числа, які будуть відтворювати тільки частину анімації;
- ключове слово **infinite**, яке буде повторювати анімацію нескінченно.

animation-direction – визначає, в якому порядку читаються ключові кадри:

- **normal:** починається з 0%, закінчується на 100%, починається з 0% знову;
- **reverse:** починається з 100%, закінчується на 0%, починається з 100% знову;
- **alternate:** починається з 0%, іде до 100%, повертається на 0%;
- **alternate-reverse:** починається з 100%, іде до 0%, повертається на 100%.

animation-fill-mode – визначає, що відбувається перед початком анімації і після її завершення; повідомляє браузеру, якщо стилі анімації також повинні застосуватися за межами анімації.

Значення:

- **none:** стилі анімації не впливають на стиль за замовчуванням;
- **forwards:** останні стилі, які застосовуються в кінці анімації, зберігаються надалі;
- **backwards:** стилі анімації будуть застосовуватись до того, як почнеться анімація;
- **both:** стилі застосовуються до і після відтворення анімації.

З огляду на останню тенденцію використовувати типографіку в якості основного елемента дизайну, цінність сервісу Google Fonts зросла багаторазово. Його інтерфейс і система завантаження інтуїтивно зрозумілі і зручні. Він дозволяє порівнювати всі доступні шрифти і стилі, щоб легше було відповідати творчому задуму, в якому, наприклад, можуть поєднуватися жирна типографіка, шрифти із зарубками і акцидентні шрифти, великі абзаци, декоративні стилі і багато іншого.

Доступ до шрифтів за покликанням https://fonts.google.com/.

З допомогою властивостей CSS можна створити багато геометричних фігур, приклади яких можна переглянути за покликанням https://easycss.top/elements/geometricheskie-figury-na-chistom-css/.

Запитання та завдання до теми

1. Для чого використовують переходи?
2. Яка властивість призначена для реалізації переходів?
3. Які параметри впливають на перехід?
4. Що таке анімація?
5. Як реалізовується анімація в CSS?
6. Які символи допустимо вводити в назві анімації?
7. Які параметри впливають на перебіг анімації?
8. Що таке ключові кадри анімації?

9. Як додати шрифт бібілотеки Google Fonts на сторінку?

Завдання 1. За допомогою переходів та псевдокласів забезпечити зміну та колір розміру блоку та тексту у ньому при наведенні на блок вказівника миші. У випадку, коли вказівник миші забирають із області блоку, він повертається до попередніх параметрів. Застосувати різні швидкості зміни та зробити висновки.

Завдання 2. Створити вебсторінку "Рух сонця по небосхилу". За допомогою CSS-анімацій забезпечити рух по кривій від нижнього лівого кутка через верх до нижнього правого кутка екрану. При цьому змінюється колір фону екрану, колір та розмір сонця.

СПИСОК ДОДАТКОВИХ ДЖЕРЕЛ

1. HTML Підручник. Початок. Уроки для початківців. W3Schools українською [Електронний ресурс] – Режим доступу до ресурсу: https://w3schoolsua.github.io/html/index.html
2. HTML і CSS довідник українською [Електронний ресурс] – Режим доступу до ресурсу: https://html-css.co.ua/
3. Український веб-довідник [Електронний ресурс] – Режим доступу до ресурсу: https://css.in.ua/
4. HTML For Beginners The Easy Way: Start Learning HTML & CSS Today [Електронний ресурс] – Режим доступу до ресурсу: https://html.com/
5. HTML Tutorial: Learn HTML For Free | Codecademy [Електронний ресурс] – Режим доступу до ресурсу: https://www.codecademy.com/learn/learn-html

ДОДАТКИ

Додаток А

Характеристика елементу <!DOCTYPE>

Елемент <!DOCTYPE> призначений для вказівки типу поточного документа – DTD (document type definition, опис типу документа). Це необхідно, щоб браузер розумів, як слід інтерпретувати поточну вебсторінку.

Існує кілька видів <!DOCTYPE>:

HTML 4.01

- <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
- <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
- <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">

HTML 5

- <!DOCTYPE HTML>

XHTML 1.0

- <!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
- <!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
- <!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

XHTML 1.1

<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

Синтаксис

<!DOCTYPE [Елемент верхнього рівня] [Публічність] "[Реєстрація]//[Організація]//[Тип] [Ім’я]//[Мова]" "[URL]">

Елемент верхнього рівня – вказує елемент верхнього рівня в документі, для HTML це тег <html>.

Публічність – об’єкт є публічним (значення PUBLIC) або системним ресурсом (значення SYSTEM), наприклад, таким як локальний файл. Для HTML/XHTML вказується значення PUBLIC.

Реєстрація – повідомляє, що розробник DTD зареєстрований у міжнародній організації зі стандартизації (International Organization for Standardization, ISO). Приймає одне з двох значень: плюс (+) – розробник зареєстрований в ISO і мінус (-) – розробник не зареєстрований. Для W3C значення ставиться «-».

Організація – унікальна назва організації, що розробила DTD. Офіційно HTML/XHTML публікує W3C, ця назва і пишеться в <!DOCTYPE>.

Тип – тип описуваного документа. Для HTML/XHTML значення вказується DTD.

Ім’я – унікальне ім’я документа для опису DTD.

Мова – мова, якою написано текст для опису об’єкту. Містить дві літери, пишеться у верхньому регістрі. Для документа HTML/XHTML вказується англійська мова (EN).

URL – адреса документа з DTD.

Додаток Б

Таблиця спеціальних символів HTML

Ім'я	Код	Вид	Опис
			нерозривний пробіл
&pound	&#163	£	фунт стерлінгів
&euro	&#8364	€	знак євро
&para	&#182	¶	символ параграфа
&sect	&#167	§	параграф
&copy	&#169	©	знак copyright
&reg	&#174	®	знак зареєстрованої торгової марки
&trade	&#8482	™	знак торгової марки
&deg	&#176	°	градус
&plusmn	&#177	±	плюс-мінус
&frac14	&#188	¼	дріб – одна четверта
&frac12	&#189	½	дріб – одна друга
&frac34	&#190	¾	дріб – три четвертих
&times	&#215	×	знак множення
&divide	&#247	÷	знак ділення
&fnof	&#402	ƒ	знак функції
Грецькі букви			
&Alpha	&#913	Α	грецька велика буква альфа
&Beta	&#914	Β	грецька велика буква бета
&Gamma	&#915	Γ	грецька велика буква гамма
&Delta	&#916	Δ	грецька велика буква дельта
&Epsilon	&#917	Ε	грецька велика буква епсилон
&Zeta	&#918	Ζ	грецька велика буква дзета
&Eta	&#919	Η	грецька велика буква ета
&Theta	&#920	Θ	грецька велика буква тета
&Iota	&#921	Ι	грецька велика буква йота
&Kappa	&#922	Κ	грецька велика буква каппа
&Lambda	&#923	Λ	грецька велика буква лямбда
&Mu	&#924	Μ	грецька велика буква мю
&Nu	&#925	Ν	грецька велика буква ню
&Xi	&#926	Ξ	грецька велика буква ксі
&Omicron	&#927	Ο	грецька велика буква омікрон
&Pi	&#928	Π	грецька велика буква пі
&Rho	&#929	Ρ	грецька велика буква ро
&Sigma	&#931	Σ	грецька велика буква сигма
&Tau	&#932	Τ	грецька велика буква тау
&Upsilon	&#933	Υ	грецька велика буква іпсилон
&Phi	&#934	Φ	грецька велика буква фі
&Chi	&#935	Χ	грецька велика буква хі
&Psi	&#936	Ψ	грецька велика буква псі
&Omega	&#937	Ω	грецька велика буква омега
&alpha	&#945	α	грецька мала буква альфа
&beta	&#946	β	грецька мала буква бета
&gamma	&#947	γ	грецька мала буква гамма
&delta	&#948	δ	грецька мала буква дельта
&epsilon	&#949	ε	грецька мала буква епсилон
&zeta	&#950	ζ	грецька мала буква дзета

&eta	&#951	η	грецька мала буква ета
&theta	&#952	θ	грецька мала буква тета
&iota	&#953	ι	грецька мала буква йота
&kappa	&#954	κ	грецька мала буква каппа
&lambda	&#955	λ	грецька мала буква лямбда
&mu	&#956	μ	грецька мала буква мю
&nu	&#957	ν	грецька мала буква ню
&xi	&#958	ξ	грецька мала буква ксі
&omicron	&#959	ο	грецька мала буква омікрон
&pi	&#960	π	грецька мала буква пі
&rho	&#961	ρ	грецька мала буква ро
&sigmaf	&#962	ς	грецька мала буква сигма
&sigma	&#963	σ	грецька мала буква сигма
&tau	&#964	τ	грецька мала буква тау
&upsilon	&#965	υ	грецька мала буква іпсилон
&phi	&#966	φ	грецька мала буква фі
&chi	&#967	χ	грецька мала буква хі
&psi	&#968	ψ	грецька мала буква псі
&omega	&#969	ω	грецька мала буква омега
Стрілки			
&larr	&#8592	←	стрілка вліво
&uarr	&#8593	↑	стрілка вверх
&rarr	&#8594	→	стрілка вправо
&darr	&#8595	↓	стрілка вниз
&harr	&#8596	↔	стрілка вліво-вправо
Інші символи			
&spades	&#9824	♠	
&clubs	&#9827	♣	
&hearts	&#9829	♥	
&diams	&#9830	♦	
&quot	&#34	"	подвійна лапка
&amp	&#38	&	амперсанд
&lt	&#60	<	знак "менше"
&gt	&#62	>	знак "більше"
Знаки пунктуації			
&hellip	&#8230	…	...
&prime	&#8242	′	мінути і фути
&Prime	&#8243	″	секунди і дюйми
Загальна пунктуація			
&ndash	&#8211	–	тире
&mdash	&#8212	—	довге тире
&lsquo	&#8216	‘	
&rsquo	&#8217	’	
&sbquo	&#8218	‚	
&ldquo	&#8220	“	
&rdquo	&#8221	”	
&bdquo	&#8222	„	
&laquo	&#171	«	
&raquo	&#187	»	

Додаток В

Приклади тестових завдань для модульного контролю

З допомогою чого один атрибут HTML відділяється від іншого?
- пробілу
- лапок
- коми
- дефісу

На які два основні типи діляться html-елементи?
- рядкові і дворядкові
- прості і складені
- блокові і рядкові
- інформаційні і декоративні

Як розшифровується CSS?
- Colorful Style Sheets
- Cascading Style Sheets
- Computer Style Sheets
- Common Style Sheets

Виберіть правильний базовий синтаксис css:
- селектор (властивість: значення;)
- селектор { властивість – значення; }
- селектор { властивість(значення); }
- селектор { властивість: значення; }

Який символ ставиться перед селектором для вказання, що це назва класу:
- #
- .
- :
- @

Який символ ставиться перед селектором для вказання, що це ідентифікатор:
- #
- .
- :
- !

Яка принципова різниця між селекторами ID і CLASS?
- ID повинен бути унікальним на сторінці, а однаковий CLASS може бути в декількох елементів

- різниця лише в наборі властивостей, який може бути використаний для цих селекторів. Наприклад, для ID не можна задавати властивість border, а для CLASS можна

- ніякої різниці між ними немає

- CLASS повинен бути унікальним на сторінці, а однаковий ID може бути у декількох елементів

Які види позиціонування елементів існують в CSS?

- flow, none, show, shift
- absolute, relative, static, fixed
- slip, relating, attached, static
- absolute, show, static, fixed

Що може бути селектором CSS?

- теги, класи, ідентифікатори
- класи, поля, елементи
- теги, класи, поля
- елементи, теги, класи

Властивість background задає...

- фон сторінки
- фон комірки
- фон таблиці
- всі відповіді вірні

Серед значень властивості border може бути...

- green
- left
- top
- 2

Що задає властивість margin?

- межу елемента
- поле між вмістом і межею елемента
- відстань між елементом та іншими елементами
- ширину вмісту елемента

Псевдокласи призначені для...

- визначення динамічного стану елементів, який змінюється з допомогою дій користувача, а також положення в дереві документа

- задання стилю елементів, не визначених в дереві елементів документа, а також генерують вміст, якого немає у вихідному тексті

- зміни звичайних класів з використанням префіксу psevdo

- задання псевдокласів

Псевдоелементи призначені для...

- задання стилю елементів, не визначених в дереві елементів документа, а також генерують вміст, якого немає у вихідному тексті

- визначення динамічного стану елементів, який змінюється з допомогою дій користувача, а також положення в дереві документа

- зміни звичайних елементів з використанням префіксу psevdo

- задання псевдоелементів

Яке визначення є правильним?

- HTML (HyperText Markup Language) – стандартна мова програмування у WWW; система верстки, яка визначає, як і які елементи повинні розміщатися на вебсторінці

- HTML (HyperText Markup Language) – стандартна мова розмітки документів у WWW; система верстки, яка визначає, як і які елементи повинні розміщатися на вебсторінці

- HTML (HyperText Markup Language) – стандартна мова у WWW; система верстки, яка визначає, чому елементи повинні розміщатися на вебсторінці

- HTML (HyperText Markup Language) – стандартна мова розмітки документів у WWW; система верстки, яка визначає, як і які елементи повинні розміщатися і виглядати у програмі

Що таке тег?

- це елемент мови розмітки гіпертексту, в основному для задання того, як буде відображатися контент

- це атрибут мови розмітки гіпертексту, в основному для задання того, як буде відображатися контент

- це елемент мови розмітки гіпертексту для задання того, де будуть розміщуватись елементи на сторінці

- це елемент мови розмітки гіпертексту для задання контейнерів

При написанні тегів можна використовувати...

- тільки малі літери

- тільки великі літери

- як великі так і малі літери

- в залежності від тегу

Для чого потрібен елемент !DOCTYPE?

- вказує браузеру, яку версію HTML використовувати для аналізування і відображення HTML сторінки

- вказує браузеру, яку версію CSS використовувати для аналізування і відображення HTML сторінки

- вказує браузеру, яку версію HTML та CSS використовувати для аналізування і відображення HTML сторінки

- вказує браузеру, хто створив вебсторінку

Який атрибут задає текст, що буде відображатися замість картинки, якщо вона не завантажилася або не відобразилася?

- src
- img
- href
- alt

Виберіть правильно написане CSS-правило:

- p \{ text-align: right; \}
- p \{ text-align: 'right'; \}
- p \{ text-align: 100%; \}
- p \{ text_align: right; \}

В якому з наведених нижче варіантів міститься явна помилка?

- p span \{font-size: 150%;\}
- p span\#text \{font-size: 150%;\}
- p \{font-size: 150%;\}
- p text (font-size: 150%;\}

Яке розширення файлу використовується для вебсторінок?

- .html
- .doc
- .web
- .txt

Вміст якого елемента видно в браузері?

- <!doctype>
- <title>
- <meta>
- <head>

З допомогою чого один атрибут відділяється від іншого?

- пробілу
- лапок
- коми
- дефісу

Який елемент створює маркований список?

-
-
-
- <dl>

Який елемент виводить зображення?

- <input>
- <pic>
- <image>

- <img>

Який елемент додає комірку таблиці?
- <tr>
- <th>
- <col>
- <td>

Який атрибут об’єднує рядки таблиці?
- rowspan
- colspan
- col
- span

Яке CSS-правило написано коректно?
- a [color: red;]
- a { color: red; }
- { a:color = red; }
- a { color = red; }

Що задає властивість padding?
- межу елемента
- поле між вмістом і межею елемента
- відстань між елементом та іншими елементами
- ширину вмісту елемента