

Методичні вказівки
до виконання лабораторних робіт з дисципліни
«Web-технології та Web-дизайн»
зі спеціальності «Комп'ютерна інженерія»
(освітня програма «Системне програмування»)

Міністерство освіти і науки України
Вінницький національний технічний університет

**Методичні вказівки
до виконання лабораторних робіт з дисципліни
«Web-технології та Web-дизайн»
зі спеціальності «Комп'ютерна інженерія»
(освітня програма «Системне програмування»)**

Вінниця
ВНТУ
2026

Рекомендовано до видання Радою з якості освіти Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 13 від 18.06.2026 р.)

Рецензенти:

О. П. Войтович, кандидат технічних наук, доцент

Р. Ю. Чехместрук, кандидат технічних наук, доцент

Методичні вказівки до виконання лабораторних робіт з дисципліни «Web-технології та Web-дизайн» зі спеціальності «Комп'ютерна інженерія» (освітня програма «Системне програмування») / уклад.: О. В. Войцеховська, В. О. Кривенька. Електрон. текст. дані. Вінниця : ВНТУ, 2026. 50 с.

Методичні вказівки підготовлено відповідно до навчального плану підготовки здобувачів вищої освіти першого (бакалаврського) рівня спеціальності «Комп'ютерна інженерія» освітньої програми «Системне програмування». Призначено для здобувачів, які вивчають освітній компонент «Web-технології та Web-дизайн». Методичні вказівки містять короткі теоретичні відомості для виконання лабораторних робіт та варіанти завдань для їх виконання.

ЗМІСТ

ВСТУП.....	4
ЛАБОРАТОРНА РОБОТА № 1. Ознайомлення з інтерфейсом Figma та базовими інструментами для вебдизайну	5
ЛАБОРАТОРНА РОБОТА № 2. Проектування макета вебсторінки у Figma	10
ЛАБОРАТОРНА РОБОТА № 3. Основи верстки вебсторінки	16
ЛАБОРАТОРНА РОБОТА № 4. Позиціонування, Flexbox та Grid для оформлення вебсторінок	23
ЛАБОРАТОРНА РОБОТА № 5. Реалізація адаптивної вебсторінки за індивідуальним дизайн-макетом	30
ЛАБОРАТОРНА РОБОТА № 6. Реалізація інтерактивних елементів на вебсторінці	36
ЛАБОРАТОРНА РОБОТА № 7. Додавання мультимедійного контенту та анімації до вебсторінки.....	41
ЛАБОРАТОРНА РОБОТА № 8. Публікація вебсайту в мережі «Інтернет»	45
РЕКОМЕНДОВАНА ЛІТЕРАТУРА	48

ВСТУП

Сучасний етап розвитку суспільства характеризується стрімким розвитком інформаційних технологій, що зумовлено глобальною цифровізацією, постійною еволюцією технологій, поширенням штучного інтелекту та зростанням вимог користувачів до якості онлайн-сервісів. При цьому вебдизайн та вебтехнології є одними з найдинамічніших напрямків, які потребують постійного оновлення знань і навичок. Вебплатформи перетворилися на основні інструменти комунікації, маркетингу та продажів. Якість вебпродуктів безпосередньо впливає на довіру клієнтів і фінансові результати компаній.

Освітній компонент «Web-технології та Web-дизайн» є складовою професійної підготовки майбутніх фахівців у галузі інформаційних технологій. Він охоплює широкий спектр знань – від технічних аспектів розробки та структурування вебресурсів до художніх принципів оформлення та побудови зручних користувацьких інтерфейсів. Даний курс орієнтований на комплексне опанування сучасних технологій веброзробки та дизайну – від побудови прототипів і макетів до реалізації готових рішень.

Метою виконання лабораторних робіт є закріплення теоретичних основ створення сучасних вебінтерфейсів, засвоєння основних понять і принципів вебдизайну, вивчення технологій та методів проєктування і верстки вебсторінок, ознайомлення з інструментами створення макетів та прототипів, а також отримання практичних навичок у побудові адаптивних і зручних для користувача інтерфейсів. Особлива увага приділяється роботі з графічним редактором Figma, верстанню за допомогою HTML і CSS, застосуванню сучасних підходів до компонування елементів та впровадженню принципів зручності використання.

Методичні вказівки містять матеріали та завдання для виконання восьми лабораторних робіт, що передбачено робочою програмою навчальної дисципліни. Лабораторні роботи сформовано таким чином, що по завершенню їх виконання здобувачі вищої освіти набувають навички для самостійного створення та оформлення вебсайтів, від розробки макета до реалізації готового проєкту.

ЛАБОРАТОРНА РОБОТА № 1

ОЗНАЙОМЛЕННЯ З ІНТЕРФЕЙСОМ FIGMA ТА БАЗОВИМИ ІНСТРУМЕНТАМИ ДЛЯ ВЕБДИЗАЙНУ

Мета роботи: ознайомитися з інтерфейсом онлайн-редактора Figma, вивчити призначення основних панелей та інструментів, засвоїти принципи роботи з фреймами, фігурами, текстовими елементами, кольоровою палітрою, шарами, групами та вирівнюванням.

Короткі теоретичні відомості

Онлайн-редактор Figma – це сучасний інструмент для створення цифрових інтерфейсів, який працює безпосередньо у веббраузері або як десктопний додаток. На відміну від традиційних програм для дизайну, що потребують складної установки і регулярних оновлень, Figma зберігає всі файли у хмарі, а отже – доступ до проєкту можна отримати з будь-якого місця і з будь-якого пристрою. Вона стала надзвичайно популярною серед дизайнерів завдяки своїй простоті, гнучкості та можливості роботи в команді у режимі реального часу. Сьогодні Figma використовується не лише для створення прототипів і макетів сайтів, але й для розробки мобільних додатків, інфографіки, презентацій та навіть друкованої продукції.

Основне завдання Figma – надати дизайнеру інструменти для створення чітких, візуально привабливих та функціональних інтерфейсів. Вона допомагає на всіх етапах дизайну – від перших ескізів до детально опрацьованого макета. Завдяки потужним інструментам компонування, налаштування стилів, підбору кольорових схем та типографіки, Figma значно скорочує час на розробку та забезпечує узгодженість дизайну між різними сторінками і елементами. Крім того, завдяки інтерактивним прототипам, які можна створювати прямо в Figma, дизайнер має можливість продемонструвати замовнику або команді розробників, як саме буде працювати інтерфейс.

Figma вигідно вирізняється серед конкурентів (таких як Adobe XD, Sketch або Photoshop) тим, що не вимагає встановлення складних пакетів і завжди має актуальну версію, оскільки працює в хмарі. Sketch, наприклад, доступний лише для macOS, тоді як Figma працює у будь-якій операційній системі, включаючи Windows, Linux, а також мобільні пристрої через додатки для перегляду. Ще одна важлива перевага – спільне редагування в реальному часі: кілька дизайнерів або розробників можуть працювати над одним і тим самим файлом одночасно, бачачи зміни колег у момент їх внесення. Це усуває потребу надсилати оновлені версії файлів електронною поштою або зберігати їх у різних папках.

Інтерфейс Figma можна умовно розділити на чотири ключові зони: панель інструментів (Toolbar), панель шарів (Layers), панель властивостей (Properties) та робоча область (Canvas). Кожна зона має своє призначення і допомагає дизайнеру ефективно працювати з макетом. Панель інструментів дає доступ до основних дій – вибір елементів, створення фігур, тексту, масштабування. Панель шарів дозволяє бачити структуру проєкту та управляти окремими елементами. Панель властивостей показує налаштування виділеного об'єкта. Робоча область – це місце, де створюється сам дизайн.

Панель інструментів розташована у верхній частині інтерфейсу та містить кнопки для створення і редагування об'єктів. Тут можна знайти інструмент для вибору елементів (Move), масштабування (Zoom), створення фреймів (Frame), фігур (Rectangle, Ellipse, Line тощо), векторних контурів (Pen), а також інструмент для додавання тексту (Text). Кожен з цих інструментів має свої гарячі клавіші, що значно пришвидшує роботу. Наприклад, натискання клавіші "R" активує інструмент прямокутника, а "T" – інструмент тексту.

Панель шарів у лівій частині інтерфейсу відображає усі елементи макета у вигляді ієрархічного списку. Це допомагає впорядковувати об'єкти, групувати їх, змінювати порядок накладання або приховувати непотрібні елементи. У великих проєктах панель шарів є особливо корисною, оскільки дозволяє швидко знайти потрібний об'єкт без необхідності його візуального пошуку на полотні.

Панель властивостей знаходиться праворуч і змінюється залежно від того, який об'єкт вибрано. Якщо виділено текст, тут можна змінити шрифт, розмір, інтерліньяж, колір. Якщо виділено фігуру – налаштувати її розміри, колір заливки, обведення, тіні, кути заокруглення. У випадку фреймів можна змінювати розміри контейнера, сітки та параметри авто-компонування (Auto Layout).

Робоча область – це центральна частина інтерфейсу, де дизайнер створює і розташовує об'єкти. Canvas у Figma практично необмежений за розміром, що дозволяє розміщувати кілька екранів або сторінок одного проєкту в межах одного файлу. Це зручно для створення повного набору сторінок сайту або послідовності екранів мобільного додатку.

Фрейми у Figma виконують роль контейнерів, які можуть містити інші об'єкти, у тому числі інші фрейми. Вони використовуються для організації макетів сторінок та екранів додатків. На відміну від простої групи об'єктів, фрейм має власні параметри, включно з розміром, сіткою та властивостями адаптивності.

Figma пропонує широкий набір базових геометричних фігур: прямокутники, кола, лінії, трикутники, багатокутники. Кожна з них може бути змінена – у прямокутника можна скруглювати кути, у кола – регулювати сектор, а багатокутник можна перетворювати на зірку.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ РОБОТИ

1. Ознайомлення з інтерфейсом

Почніть роботу з Figma із входу у свій обліковий запис або реєстрації на сайті figma.com. Після авторизації створіть новий документ, обравши пункт New design file. Перед вами відкриється робоча область програми, поділена на кілька панелей: зверху розташована панель інструментів, ліворуч – панель шарів, праворуч – панель властивостей. Ретельно розгляньте кожну з них: зверніть увагу, як змінюється панель властивостей залежно від обраного елемента, і як панель шарів відображає ієрархію об'єктів. Далі створіть новий фрейм (клавіша F) розміром 1440×1024 пікселі. На ньому розмістіть кілька простих фігур: квадрат, коло, лінію, стрілку та зірку. У панелі "Layers" дайте кожному об'єкту зрозумілу назву. Спробуйте поєднати 2–3 фігури в єдину композицію, наприклад, створивши стилізований будиночок або смайлик.

2. Робота з текстом

Текст у Figma створюється за допомогою інструменту Text (клавіша T). Додайте у свій макет заголовок великим, жирним шрифтом розміром 32 px. Нижче розмістіть абзац пояснювального тексту шрифтом розміром 14–16 px у звичайному (regular) накресленні. Відрегулюйте вирівнювання, інтерліньяж і відступи так, щоб текст легко читався та виглядав охайно. Для закріплення навичок створіть власну міні-візитку: намалюйте прямокутник, який слугуватиме фоном, і розмістіть на ньому своє ім'я, статус (наприклад, "Студент групи 1KI-266") та адресу електронної пошти. Вирівняйте всі елементи по центру, використовуйте Auto Layout для розташування тексту, а готову візитку перетворіть на компонент.

3. Кольори і заливка

Виділіть будь-яку фігуру та змініть її колір у панелі властивостей, у розділі Fill. Додайте обведення (Stroke), експериментуйте з його товщиною та кольором. Спробуйте застосувати ефекти, такі як тінь (Drop shadow) або розмиття (Blur). Для практики створіть серію із п'яти-семи кіл різних кольорів, під кожним напишіть назву емоції або настрою, з яким асоціюється колір (наприклад, жовтий – “енергія”, блакитний – “спокій”). Розмістіть кола з підписами в один рядок, використовуючи Auto Layout для рівномірного вирівнювання.

4. Шари, групи, вирівнювання

Працюючи у Figma, приділяйте увагу структурі вашого проєкту. У панелі Layers перейменовуйте об'єкти, щоб легко знаходити потрібні елементи. Об'єднуйте кілька пов'язаних елементів у групи (Ctrl+G) та використовуйте інструменти вирівнювання для точного розташування. Зверніть увагу на різницю між групою та фреймом: група просто з'єднує об'єкти, тоді як фрейм може виступати контейнером зі своїми параметрами. Для вправи створіть три іконки з простих форм – наприклад, іконку "Домашня сторінка", "Налаштування" та "Кошик". Додайте під кожною іконкою підпис, згрупуйте пари "іконка + текст", а потім розташуйте їх в одному рядку, використавши Auto Layout.

5. Auto Layout

Auto Layout —інструмент, що автоматично впорядковує елементи. Створіть кнопку, що складається з прямокутника та напису "Натисни мене". Виділіть обидва елементи та застосуйте Auto Layout (Shift+A). Додайте внутрішні відступи, скругліть кути прямокутника, змініть колір заливки та додайте тінь. Перетворіть кнопку на компонент, а потім створіть ще два варіанти для станів "Hover" та "Active". Таким чином ви отримаєте інтерактивний набір кнопок, готових до використання в макетах.

6. Компоненти

Компоненти дозволяють створювати елементи, які можна багаторазово використовувати та редагувати у всіх місцях одночасно. Виділіть будь-який створений вами об'єкт і перетворіть його на компонент (Ctrl+Alt+K). Створіть кілька копій (Instances) і змініть у них вміст – наприклад, текст або зображення. Створіть картку товару, що містить зображення (заливка у фігуру), назву, ціну та кнопку "Купити". Використайте Auto Layout для охайного розташування елементів, а потім перетворіть картку на компонент. Створіть ще кілька карток із різними товарами, щоб побачити, як швидко можна оновлювати дизайн.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке Figma і для чого вона використовується?
2. Які основні панелі інтерфейсу Figma ви знаєте та для чого вони призначені?
3. Чим відрізняється фрейм від групи елементів?
4. Як створити новий фрейм заданого розміру?
5. Які інструменти Figma дозволяють створювати прості геометричні фігури?
6. Як змінити назву шару в панелі Layers і чому це важливо?

7. Що таке Fill та Stroke у Figma і як ними користуватися?
8. Які ефекти можна застосувати до об'єктів у Figma? Наведіть приклади.
9. Як створити текстовий блок і змінити його шрифт, розмір та міжрядковий інтервал?
10. Що таке Auto Layout і які можливості він надає дизайнеру?
11. Як задати внутрішні та зовнішні відступи для елементів у Auto Layout?
12. Що таке компонент у Figma та які його переваги?
13. Чим відрізняється Component від Instance?
14. Як швидко змінити колір і обведення кількох об'єктів одночасно?
15. Які типи сіток можна додати у Figma та для чого вони потрібні?
16. Що таке 12-колонкова сітка і чому вона є стандартом у вебдизайні?
17. Як вирівняти кілька елементів між собою у Figma?
18. Яким чином можна створити кілька станів кнопки (Normal, Hover, Active) у Figma?
19. Як правильно експортувати готовий макет або окремі елементи з Figma?
20. Які переваги Figma у порівнянні з іншими графічними редакторами для вебдизайну?

ЛАБОРАТОРНА РОБОТА № 2

ПРОЄКТУВАННЯ МАКЕТА ВЕБСТОРІНКИ У FIGMA

Мета роботи: навчитися проєктувати макет вебсторінки у середовищі Figma, застосовуючи базові інструменти для створення структурованого та зручного інтерфейсу. Закріпити навички роботи з фреймами, сітками, текстовими блоками, кольоровими схемами та компонентами. Розвинути вміння поєднувати елементи дизайну у єдиний макет, який відображає логіку розташування контенту та відповідає принципам сучасного вебдизайну.

Короткі теоретичні відомості

Проєктування макета вебсторінки є одним із ключових етапів створення сучасних вебсайтів. Макет дозволяє наочно представити майбутній вигляд сайту ще до написання коду. У ньому визначаються структура, стиль, розташування елементів і загальний настрій інтерфейсу. Це своєрідна «карта», за якою рухається і дизайнер, і розробник, і навіть замовник, щоб зрозуміти, чи правильно реалізується задум.

Розпочинаючи створення макета, важливо усвідомлювати, що він має поєднувати два напрями: UI (User Interface) – зовнішній вигляд інтерфейсу та UX (User Experience) – досвід користувача. Якщо перший відповідає за естетику, гармонію кольорів і візуальну привабливість, то другий – за зручність використання, логіку переходів і зрозуміле розташування кнопок чи текстів.

Макетування починається з розуміння цілей сайту. Наприклад, лендинг має швидко привернути увагу користувача й підштовхнути його до дії – купівлі товару, реєстрації чи підписки. Інтернет-магазин має зручно презентувати товари, забезпечити легку навігацію й просту форму оформлення замовлення. Інформаційний сайт акцентує увагу на зручному читанні текстів і чіткому розподілі матеріалів.

Будь-яка вебсторінка складається з окремих логічних блоків. Зазвичай це header (шапка) з логотипом і меню, hero-блок – перший екран із великим заголовком і кнопкою дії, контентні секції з текстами, зображеннями чи картками товарів, і footer (підвал) з контактами та додатковими посиланнями. Вміння правильно компоувати ці частини формує професійний макет.

Для створення зручного й структурованого макета дизайнери застосовують сітки (layout grid). Найчастіше це система з 12 колонок, яка дозволяє рівномірно розташовувати об'єкти та забезпечує їхню адаптивність під різні розміри екранів. Завдяки сітці користувачеві легко сприймати інформацію, а верстальнику – швидко переносити макет у HTML та CSS.

Крім сітки, у Figma є інструменти для вирівнювання й відступів. Вони допомагають уникнути «хаосу» на сторінці, коли елементи розташовані нерівномірно. Правильне вирівнювання формує відчуття гармонії, а відступи дозволяють «дихати» кожному блоку, щоб сайт не виглядав перевантаженим.

Favicon – це невелика іконка, яка асоціюється з вебсайтом. Вона відображається у вкладках браузера, закладках і списку історії переглядів. Favicon допомагає користувачам швидко орієнтуватися серед багатьох відкритих вкладок та легко знаходити сайт серед закладок, додаючи при цьому професійний вигляд вебресурсу. Стандартний розмір favicon зазвичай становить 16×16 або 32×32 пікселів. Найпоширеніші формати – .ico, .png та .svg. Формат .svg зручний тим, що він масштабований, тому іконка залишається чіткою на екранах з різною роздільною здатністю.

Дизайн favicon зазвичай відображає логотип або символ бренду. Він має бути простим і легко впізнаваним навіть у маленькому розмірі, а також відповідати загальному стилю сайту. Контрастні кольори та мінімалістичні форми допомагають зробити іконку помітною та зрозумілою користувачам.

На практиці favicon можна створити у Figma, використовуючи Shape Tool, Text Tool або готовий SVG. Після завершення його експортують у потрібному форматі для сайту та підключають через тег `<link rel="icon" href="favicon.ico">` у HTML-документі. Це дозволяє браузерам правильно відображати іконку на всіх пристроях.

Важливою складовою макета є типографіка. Шрифти на сайті повинні бути читабельними й відповідати стилю проєкту. Наприклад, для корпоративного сайту підійдуть строгі геометричні шрифти, для блогу – більш легкі й нейтральні. Як правило, використовують два шрифти: один для заголовків (великий і виразний), інший для основного тексту (простий і зручний для читання).

Окрім шрифтів, значну роль відіграє кольорова палітра. Вебдизайнери рекомендують використовувати обмежену кількість основних кольорів – зазвичай 2–4. Важливо, щоб вони гармоніювали між собою й відповідали змісту сайту. Наприклад, для медичного порталу доречні блакитні й зелені тони, а для сайту ресторану – теплі відтінки червоного та коричневого.

У Figma можна зберігати кольорові стилі, щоб забезпечити єдність дизайну. Якщо дизайнер вирішить змінити відтінок головного кольору, то зміна автоматично застосовується до всіх елементів макета. Це суттєво економить час і робить дизайн гнучким.

Ще одним важливим інструментом є компоненти. Це готові елементи (наприклад, кнопки, картки товарів чи іконки), які можна багаторазово використовувати в різних місцях. Якщо змінити головний компонент, то оновляться всі його копії. Це забезпечує консистентність дизайну й полегшує підтримку проєкту.

Створюючи макет, необхідно дотримуватись принципів візуальної ієрархії. Найважливіші елементи повинні виділятися – бути більшими, яскравішими або розташовуватися у верхній частині сторінки. Другорядні елементи можуть бути меншими й спокійнішими. Це допомагає користувачу інтуїтивно орієнтуватися в інформації.

Крім візуальної ієрархії, велике значення має контраст. Він може бути не лише кольоровим, а й за формою, розміром чи товщиною шрифту. Наприклад, темний текст на світлому фоні завжди читається краще. Контраст дозволяє швидко привертати увагу до потрібних блоків.

У сучасному дизайні активно застосовується Auto Layout – інструмент Figma, що допомагає автоматично вирівнювати елементи. Завдяки йому створення списків, карток чи кнопок стає значно простішим. Якщо додати новий елемент у список, усі інші автоматично посунуться, зберігаючи правильні відступи.

Проектуючи макет, не можна забувати про адаптивність. Сайт має добре виглядати як на великому моніторі, так і на екрані смартфона. Тому в Figma можна створювати кілька варіантів одного макета під різні розміри екрану: desktop, tablet, mobile. Це дозволяє передбачити, як змінюватиметься структура.

Дуже важливо, щоб макет був зрозумілим не лише дизайнеру, а й верстальнику. Тому об'єкти необхідно правильно називати у панелі «Layers», групувати їх і організовувати у фрейми. Акуратність у роботі з шарами економить час усіх членів команди.

Бібліотека стилів у Figma – це інструмент, який дозволяє зберігати стандартизовані елементи дизайну, такі як кольори, шрифти, відступи та ефекти. Вона створюється для того, щоб дизайн залишався узгодженим і послідовним у межах одного проєкту або всієї організації.

Основна мета бібліотеки стилів – забезпечити єдиний візуальний стиль. Коли всі елементи використовують однакові кольори, шрифти та ефекти, інтерфейс виглядає гармонійно, а користувачі швидше звикають до дизайну. Колірні стилі в бібліотеці дозволяють зберігати палітру кольорів, яка використовується у проєкті. Це можуть бути основні кольори бренду, кольори кнопок або тексту. Зміна кольору в стилі автоматично оновлює всі елементи, до яких він застосований.

Стилі тексту включають шрифти, розмір, товщину, міжрядковий інтервал та колір тексту. Завдяки цьому можна швидко змінювати оформлення заголовків, підзаголовків і основного тексту по всьому макету без ручного редагування кожного елемента. В бібліотеці стилів можна також зберігати ефекти, наприклад тіні, розмиття або підсвітку. Це дозволяє використовувати однакові візуальні ефекти для кнопок, карток і інших елементів інтерфейсу, створюючи відчуття єдності. Стилі відступів і сітки допомагають дотримуватися пропорційності та правильного розташування елементів на сторінці. Збереження цих параметрів у бібліотеці робить макет більш структурованим і легким для редагування.

Використання бібліотеки стилів значно економить час дизайнерів. Не потрібно повторно налаштовувати кольори або шрифти для кожного нового елемента – достатньо застосувати готовий стиль. Бібліотека стилів також корисна для командної роботи. Всі учасники проєкту використовують одні й ті ж стандарти, що зменшує ризик помилок і дублювання стилів. Крім того, бібліотека стилів спрощує масштабування проєкту. Якщо потрібно додати нові елементи або розширити інтерфейс, можна швидко застосувати вже існуючі стилі, зберігаючи цілісність дизайну. На практиці рекомендується створювати бібліотеку стилів на самому початку проєкту, визначити основні кольори, шрифти і ефекти, а потім постійно її оновлювати. Це дозволяє швидко адаптувати дизайн під нові вимоги і підтримувати його сучасний вигляд.

Ще одна важлива складова – іконки та ілюстрації. Вони роблять дизайн живим і сучасним. Проте важливо, щоб вони були єдиного стилю. Використання різних наборів ікон може зруйнувати цілісність візуальної мови сайту. У Figma можна підключати плагіни для швидкого пошуку готових ікон.

У процесі створення макета доцільно користуватися принципом модульності. Це означає, що дизайн має складатися з окремих блоків, які легко можна переставити чи замінити. Такий підхід робить макет більш гнучким і готовим до змін у майбутньому.

Після завершення створення макета його варто протестувати. У Figma є режим презентації (Prototype), який дозволяє натиснути на кнопки й переходити між екранами, імітуючи роботу сайту. Це дає можливість оцінити зручність навігації та перевірити, чи все працює інтуїтивно.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ РОБОТИ

Створіть у Figma макет простої вебсторінки, наприклад «Про себе» (Landing Page / Homepage). Ваша сторінка має презентувати вас як особистість. (Тематика для розробки вебсторінки змінюється щорічно).

Обов'язкові вимоги до макета:

1. Використовуйте 12-колонкову **Layout Grid** у фреймі сторінки. Задайте відступи та поля (margins, gutters) для правильної композиції. Кожна секція сторінки має бути оформлена у фреймі (Frame).

2. У Header (шапка сайту) створіть **власний логотип** (у вигляді тексту або іконки, зроблений через Text Tool або Shape), додайте меню навігації, зроблене у вигляді Auto Layout (горизонтальне вирівнювання). Окрім цього, додайте фавікон (іконку сайту) у верхній частині макета.

3. У Hero section (головний блок) додайте заголовок (Text Tool, великий шрифт), підзаголовок (менший шрифт), **кнопку СТА** (створіть компонент кнопки), а також фото або ілюстрація (вставка через Place Image).

4. Додаткові секції (мінімум 3). Використовуйте різні прийоми:

- Auto Layout для списків або карток.

- **Компоненти** для повторюваних елементів (картки товарів/проектів, відгуки).

- Icons / Illustrations (<https://iconscout.com/icons>).

Приклад секцій: «Про мене», «Навички», «Мої проекти», «Відгуки», «Контакти», «Моє хобі», «Моє дитинство», «Моя родина», «Домашні улюбленці».

5. Зробіть власну **бібліотеку стилів**. Для цього підберіть основні кольори, які будуть використані у макеті, і збережіть їх у вигляді Color Styles. У бібліотеці мають бути визначені стилі для основного акцентного кольору (Primary), додаткового (Secondary), фону (Background) та тексту (Text). Оберіть не менше 2–3 кольорів, які добре поєднуються між собою. Далі налаштуйте Text Styles: окремо для заголовків (наприклад, h1, h2) та для основного тексту (параграфи, підписи чи кнопки). Використайте не менше двох різних шрифтових стилів, які відрізнятимуться розміром, товщиною чи гарнітурою. Оформіть бібліотеку так, щоб будь-які зміни у стилях автоматично оновлювали всі елементи на сторінці.

6. Організація макета у Figma. Усі секції мають бути **згруповані** у фрейми з назвами: Header, Hero, About, Skills, Contacts. Шари у Layers мають зрозумілі назви (не залишати «Rectangle 123»). Використовуйте **Constraints** для правильного позиціонування елементів відносно сторін фрейму.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке сітка (layout grid) у Figma та для чого вона використовується?
2. Які елементи зазвичай входять до шапки сайту (header)?
3. Що таке hero section і яку функцію він виконує у вебдизайні?
4. Які інформаційні секції найчастіше можна зустріти на вебсторінках?
5. У чому полягає різниця між компонентом і копією об'єкта у Figma?
6. Для чого використовують Auto Layout і які його основні можливості?
7. Чому важливо дотримуватись єдиної кольорової палітри та шрифтової пари у макеті?
8. Які принципи вирівнювання елементів у макеті вважаються правильними?
9. Чим відрізняється групування об'єктів від створення фреймів у Figma?
10. Які переваги має робота з компонентами під час проектування вебінтерфейсу?

11. Створіть hero section із заголовком, кнопкою та зображенням. Покажіть, як додати зображення через панель «Assets» або вставку «Place Image».
12. Створіть компонент кнопки та додайте його копії на макет. Покажіть, де у верхній панелі знаходиться кнопка «Create Component» і як застосувати зміни до всіх копій.
13. Створіть фрейм з трьома кнопками та застосуйте Auto Layout. Покажіть, де у панелі властивостей активувати «Auto Layout» і налаштувати відступи між елементами.
14. Застосуйте кольоровий стиль і стиль тексту з бібліотеки стилів. Покажіть, де в панелі «Assets» або «Styles» знаходяться збережені кольори та текстові стилі.
15. Створіть групу з 3 елементів і окремий фрейм з 3 інших елементів. Покажіть, де кнопки «Group» і «Frame» у верхній панелі та порівняйте поведінку елементів.
16. Що таке бібліотека стилів у Figma і яку роль вона відіграє у підтримці єдиного дизайну проєкту?
17. Які типи стилів можна зберігати в бібліотеці (наприклад, кольори, текст, ефекти) і як вони впливають на узгодженість макета?
18. Як застосування стилів з бібліотеки спрощує редагування макета та роботу в команді?
19. Що таке Constraints у Figma і яку функцію вони виконують під час створення адаптивного дизайну?
20. Які типи Constraints існують і як вони впливають на поведінку об'єкта при зміні розміру фрейму?

ЛАБОРАТОРНА РОБОТА № 3

ОСНОВИ ВЕРСТКИ ВЕБСТОРІНКИ

Мета роботи: Ознайомитися з базовою структурою HTML-документа. Навчитися створювати основні елементи вебсторінки: заголовки, абзаци, списки, таблиці, зображення, посилання, форми. Освоїти принципи підключення та застосування CSS для оформлення вебсторінки.

Короткі теоретичні відомості

HTML (HyperText Markup Language) – це основа будь-якої вебсторінки. Його роль полягає не у візуальному оформленні, а у визначенні структури та змісту. Браузер читає HTML-код і відображає його як сайт.

Будь-який HTML-документ починається з декларації `<!DOCTYPE html>`. Вона потрібна, щоб браузер працював у «сучасному режимі» й коректно відображав сторінку.

Кореневим елементом сторінки є `<html>`. Він охоплює увесь документ і зазвичай має атрибут `lang="uk"` або іншу мову, що важливо для пошукових систем і доступності. У середині `<html>` є два основні блоки: `<head>` і `<body>`. У `<head>` міститься службова інформація (метадані), у `<body>` – контент, який бачить користувач. У `<head>` можна розмістити тег `<meta charset="UTF-8">`, що задає кодування символів. Також тут додаються стилі CSS, скрипти JavaScript і SEO-дані через теги `<meta name="description">` чи `<meta name="keywords">`.

Заголовки `<h1>`–`<h6>` формують ієрархію документа. `<h1>` використовується для головного заголовку сторінки, а інші – для підрозділів. Правильна ієрархія важлива для пошукових систем і читабельності. Для абзців використовується тег `<p>`. Він дозволяє структурувати текст, розбиваючи його на логічні частини. Виділення важливого тексту можна зробити через `<strong>` (сильний акцент), `<em>` (курсив, акцент), `<mark>` (підсвічування кольором). Також є `<u>` (підкреслення) і `<small>` (дрібний текст).

Для цитат застосовується `<blockquote>`, а короткі цитати всередині рядка можна робити через `<q>`. Є також тег `<cite>`, яким позначають автора чи джерело цитати. Горизонтальна лінія `<hr>` використовується для поділу контенту. Вона вважається тематичним роздільником, а не просто візуальною лінією.

Для списків застосовують `<ul>` (маркований список), `<ol>` (нумерований список) і `<li>` (пункт). Є також тег `<dl>` (описовий список), який складається з `<dt>` (термін) та `<dd>` (визначення).

Таблиці створюються через `<table>`. У ній використовуються рядки `<tr>`, клітинки `<td>` та заголовки клітинок `<th>`. Для групування є `<thead>`, `<tbody>`, `<tfoot>`.

Щоб показати дані більш сучасно, у HTML5 додали тег `<figure>` для зображень, діаграм і медіа з підписом. Усередині нього можна використовувати `<figcaption>` для текстового опису. Зображення вставляються тегом `<img>`. Атрибут `alt` описує його зміст, що важливо для людей із вадами зору та у випадку, якщо картинка не завантажилася.

Посилання створюються тегом `<a href="...">`. Вони можуть відкриватися у тій самій вкладці або новій (`target="_blank"`). Для інтеграції відео є тег `<video>`, а для аудіо – `<audio>`. Обидва можуть мати атрибут `controls`, що додає панель керування. Є також тег `<iframe>`, який дозволяє вбудовувати інші вебсторінки (наприклад, відео з YouTube чи Google Maps).

Семантичні теги роблять HTML більш зрозумілим. Наприклад, `<header>` – шапка сайту, яка зазвичай містить логотип, меню чи пошук. Тег `<nav>` використовується для основної навігації сайту (меню, посилання на розділи). Це допомагає пошуковим системам і скрінрідерам. Тег `<main>` містить основний контент сторінки, який унікальний для даної вебсторінки. Він не повинен дублювати `header` чи `footer`. Тег `<section>` позначає окремий логічний розділ сторінки (наприклад, «Про нас», «Послуги»). Тег `<article>` використовується для самостійного контенту – новини, поста в блозі, статті. Його можна копіювати та поширювати незалежно. Тег `<aside>` призначений для додаткової інформації (бокова панель, посилання, реклама, примітки). Тег `<footer>` містить підвал сайту – контактні дані, авторські права, посилання на політику конфіденційності тощо.

Крім основних, існують менш поширені, але цікаві семантичні теги:

`<time>` – для позначення дат і часу;

`<address>` – для контактних даних;

`<abbr>` – для скорочень (з підказкою при наведенні);

`<progress>` – для індикатора виконання завдання;

`<meter>` – для вимірювання у межах шкали (наприклад, рівень батареї).

CSS (Cascading Style Sheets) – це мова, яка визначає зовнішній вигляд вебсторінки. Якщо HTML відповідає за структуру та вміст (що показати), то CSS вирішує, як саме це буде виглядати (кольори, шрифти, розташування елементів). Без CSS сторінки виглядають сухо і одноманітно, а завдяки стилям можна зробити їх зручними для читання, привабливими й сучасними.

У CSS кожне правило складається з двох частин: селектора і блоку властивостей. Селектор визначає, до якого елемента застосовуються стилі, а в блоці вказується, що саме змінювати. Наприклад, правило `p { color: red; }` означає: усі абзаци (`<p>`) на сторінці мають бути червоного кольору. Це дуже зручно, адже достатньо змінити один рядок стилів, і весь сайт набуває іншого вигляду.

Є три основні способи як можна задати стилі:

- `inline-стилі` – прямо в атрибуті `style` елемента;

- внутрішні стилі – у секції <style> всередині HTML;
- зовнішні стилі – у окремому файлі .css, який підключається через <link>.

У реальних проєктах найбільше використовують зовнішні файли, бо це зручно: стилі відокремлені від структури, їх можна підключати одразу до кількох сторінок.

Каскадність означає, що стилі можуть нашаровуватися один на одного, і браузер вирішує, які саме застосувати. Наприклад, якщо для абзацу в одному місці задано color: blue, а в іншому color: red, спрацює той стиль, який має більший пріоритет. Пріоритет залежить від того, де саме прописаний стиль і який селектор використано.

Селектори – це інструмент, що дозволяє "знайти" потрібний елемент на сторінці. Можна вибрати всі абзаци (p), всі елементи з певним класом (.menu), конкретний елемент з ідентифікатором (#header). Також існують складніші селектори, які дозволяють впливати на елементи залежно від їх положення у структурі (наприклад, ul li a – це всі посилання всередині списку).

Іноді важливо стилізувати елемент не завжди, а лише за певних умов. Для цього існують псевдокласи. Наприклад, a:hover задає стиль посиланням, коли на них наводять курсор. Є й інші: :first-child, :last-child, :nth-child(n), які допомагають працювати з таблицями чи списками. Це можливість "намалювати" додатковий вміст без зміни HTML-коду. Наприклад, p::before { content: "→ "; } додасть стрілочку перед кожним абзацом. Подібно працює ::after, що зручно для декоративних елементів.

CSS дозволяє робити текст більш читабельним. Властивості color, font-size, font-family, font-weight, font-style відповідають за колір, розмір, шрифт, товщину й курсив. Наприклад, можна задати стиль: h1 { font-size: 32px; font-family: Arial; font-weight: bold; }. Крім основних стилів, можна змінювати інтервали між літерами (letter-spacing), між словами (word-spacing) і рядками (line-height). Властивість text-shadow додає тінь, завдяки чому текст виглядає об'ємніше.

Текст можна вирівняти зліва, справа, по центру або навіть "по ширині" (justify). Це робиться властивістю text-align. Наприклад, для статей часто використовують justify, щоб абзаци виглядали рівномірно.

У CSS можна задавати кольори різними способами. Найпростіше – назвою (red, blue). Часто використовують HEX-коди (#ff0000), або формат rgb(255,0,0). Якщо потрібна прозорість, застосовують rgba, де останній параметр – рівень прозорості (0 до 1). Це дозволяє робити стильні ефекти накладання.

За допомогою background-image можна додати картинку як фон елемента. Властивість background-size дозволяє підлаштувати її під блок: cover розтягує на весь блок, а contain зберігає пропорції. Можна також зробити фон нерухомим (background-attachment: fixed), що створює ефект "паралакса".

Усі блоки мають внутрішні (padding) та зовнішні (margin) відступи. Наприклад, щоб текст не прилипав до рамки, задають padding: 10px;. А щоб відокремити один блок від іншого, використовують margin.

Будь-якому елементу можна задати рамку. У неї є товщина (2px), стиль (solid, dashed, dotted) і колір.

Усі HTML-елементи можна уявити як коробку, яка має чотири шари: вміст (content), внутрішні відступи (padding), рамку (border) і зовнішні відступи (margin). Це важлива концепція, яка допомагає правильно розташовувати елементи

У списків можна змінювати маркери за допомогою list-style-type. Наприклад, circle, square або навіть прибрати маркери зовсім (none). Також є властивість list-style-image, що дозволяє використовувати картинку замість звичайної крапки.

Щоб таблиці виглядали охайно, часто використовують border-collapse: collapse; (злиття рамок), а також задають фон для заголовків (th) і чергування кольорів рядків за допомогою псевдокласів nth-child.

За замовчуванням усі елементи розташовуються статично один під одним. Але іноді треба їх пересунути. Для цього є властивість position, яка може бути relative, absolute, fixed або sticky. Це дозволяє робити шапки, які завжди залишаються зверху, або кнопки, що "приклеюються" при прокрутці.

CSS дозволяє змінювати спосіб відображення елементів. Наприклад, блоковий елемент (display: block) займає всю ширину рядка, а рядковий (inline) розташовується поряд із текстом. Є також inline-block і сучасне значення flex, яке дозволяє легко вирівнювати елементи. Flexbox – це спеціальна модель розташування елементів, яка значно полегшує верстку. Вона дозволяє легко вирівнювати блоки по вертикалі й горизонталі, робити адаптивні сітки та динамічні розмітки. Grid Layout – ще потужніший інструмент, ніж Flexbox. Grid дозволяє створювати сітки з рядків і колонок, розставляти елементи в будь-якому місці. Це основа сучасної верстки сайтів, бо замінює старі "табличні" підходи.

У CSS можна стилізувати поля вводу (input), кнопки (button) і текстові області (textarea). Наприклад, змінювати їхні кольори при фокусі (:focus) або натисканні (:active). Це важливо для зручності користувача.

CSS дозволяє обертати (rotate), масштабувати (scale), зсувати (translate) та нахилити (skew) елементи. Це використовується для створення анімацій або цікавих ефектів.

За допомогою ключових кадрів (@keyframes) можна створювати плавні рухи елементів. Наприклад, кнопка може поступово змінювати колір або обертатися. Анімації роблять сторінку більш живою та інтерактивною.

Щоб зміна кольору чи розміру була плавною, використовують transition. Наприклад, якщо задати button { transition: background 0.3s; }, то при наведенні кнопка змінить фон не різко, а плавно.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ РОБОТИ

Частина 1

Створіть один HTML-документ, у якому виконайте всі завдання. Кожне завдання оформіть у окремому блоці з коментарем `<!-- Завдання № -->`. Використовувати CSS **заборонено**.

1. Додайте заголовки від `<h1>` до `<h6>` (наприклад, як розділи статті чи документації). Додайте 2 абзаци тексту, використовуючи теги `<strong>`, `<em>`, `<mark>` для виділення важливих фрагментів. Додайте цитату з використанням тегу `<blockquote>`. Додайте біжучий рядок із власним текстом за допомогою спеціального HTML-тега. Використовуйте горизонтальну лінію `<hr>` для відділення блоків.

2. Зробіть маркований список продуктів. Зробіть нумерований список рецептів (кроки приготування страви). Вставте всередину одного пункту вкладений список.

3. Створіть таблицю **на тему «Розклад уроків»** з 4 колонками і 5 рядками. Використовуйте теги `<thead>`, `<tbody>`, і `<tfoot>`. Приклад таблиці наведено на рисунку 1.

Розклад уроків

День	Урок 1	Урок 2	Урок 3
Понеділок	Математика	Історія	Фізика
Вівторок	Українська мова	Англійська	Інформатика
Середа	Біологія	Хімія	Фізкультура
Четвер	Географія	Математика	Література
П'ятниця	Інформатика	Музика	Трудове навчання

Рисунок 1 – Приклад початкової таблиці

Об'єднайте колонки, як показано на рисунку 2.

Розклад уроків

День	Уроки			Примітки
	Урок 1	Урок 2	Урок 3	
Понеділок	Математика	Історія	Фізика	Контрольна з математики
	Інформатика (практика, 2 години)		Хімія	
Вівторок	Українська мова	Англійська (поглиблений курс)		-
Середа	Біологія	Фізкультура	Географія	Скорочений день
Четвер	Хімія	Лабораторна робота		
П'ятниця	Екскурсія (весь день)			-

Рисунок 2 – Приклад об'єднання колонок

4. Вставте будь-яке зображення (можна з інтернету, посиланням). Зробіть це зображення клікабельним.

5. Створіть форму на тему «Зворотній зв'язок» з полями: ім'я, e-mail, повідомлення. Додайте кнопку «Надіслати». Зробіть поле e-mail з типом email, а для повідомлення використайте <textarea>.

6. Створіть структуру з використанням семантичних тегів <header>, <main>, <nav>, <aside>, <section>, <article>, <progress>, <footer>. У header додайте назву сайту, в main – текст статті, у footer – контактні дані і т. д.

Частина 2

1. Задайте різні кольори для кожного рівня заголовків <h1> ... <h6>. Використайте різні шрифти для заголовків і тексту. Додайте відступи між абзацами та зробіть вирівнювання тексту по ширині (text-align: justify). Для <blockquote> зробіть ліву вертикальну лінію й курсив. Стилiзуйте <mark> – наприклад, змініть колір фону і тексту на власний.

2. Задайте власний стиль маркерів у маркованому списку (наприклад, кружечки, квадрати). Додатково задайте стиль маркеру, як власну картинку. Для нумерованого списку використайте римські цифри. Зробіть відступи між пунктами списків. Вкладений список зробіть із іншим стилем маркерів (наприклад, «—» або circle).

3. Зробіть таблицю з рамками та злитими межами (border-collapse: collapse). Додайте різні кольори для рядків (nth-child(even) і nth-child(odd)). Заголовок таблиці (thead) виділіть іншим фоном і жирним шрифтом. Для об'єднаних клітинок задайте вирівнювання тексту по центру і вертикалі. Зробіть hover-ефект: при наведенні на рядок змінюється його фон.

4. Для зображення встановіть обмеження максимальної ширини, наприклад 400px. Додайте рамку та тінь (box-shadow). При наведенні на зображення зробіть легке збільшення (transform: scale(1.05)). Для клікабельності додайте ефект наведення (курсор змінюється на pointer).

5. Для форми задайте ширину полів (input, textarea) і зробіть їх з плавними заокругленими кутами. Додайте відступи (padding) усередині полів і між ними. Кнопку «Надіслати» зробіть кольоровою з hover-ефектом (наприклад, змінює колір при наведенні). Для поля e-mail зробіть зелений контур при правильному введенні (:valid) і червоний – при неправильному (:invalid).

6. Для семантичної структури додайте:

- Для <header> додайте фон і змініть колір тексту.
- У <nav> створіть горизонтальне меню та вирівняйте його елементи.

- Для <main> зробіть ширину контенту обмеженою (наприклад, 800px) і центруйте його.

- Для <section> та <article> додайте тіні або рамки для відокремлення.

— Для `<aside>` створіть блок з додатковою інформацією (наприклад, коротке резюме або посилання).

— Для `<footer>` зробіть затемнений фон і світлий текст.

Додайте внутрішні відступи (`padding`) для всіх блоків, щоб контент не прилипав до країв.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке HTML і яку роль він відіграє у створенні вебсторінок?
2. Для чого використовуються теги `<h1>` ... `<h6>` і чим вони відрізняються?
3. Яке призначення тегів `<strong>`, `<em>`, `<mark>`? У чому їхня різниця?
4. Які є основні типи списків у HTML і коли варто використовувати кожен?
5. Як створити вкладений список? Наведіть приклад.
6. Для чого потрібен тег `<blockquote>`? Чим він відрізняється від `<q>`?
7. Яка структура таблиці у HTML? Які теги в ній обов'язкові?
8. Як у таблиці об'єднати кілька клітинок? Які атрибути для цього використовуються?
9. Що таке семантичні теги? Наведіть приклади та поясніть їх значення.
10. Чим відрізняються теги `<header>`, `<main>`, `<section>`, `<article>`, `<footer>`?
11. Для чого потрібен тег `<form>` і які є основні типи полів у формах?
12. Які способи підключення CSS ви знаєте? Який із них вважається найкращим і чому?
13. Що таке каскадність у CSS? Як браузер вирішує, який стиль застосувати?
14. Чим відрізняються селектори за тегом, класом та ідентифікатором?
15. Що таке псевдокласи і псевдоелементи? Наведіть приклади їх використання.
16. Що таке модель коробки (Box Model) у CSS? З яких частин вона складається?
17. У чому різниця між `inline`, `block` та `inline-block` елементами?
18. Які можливості надають Flexbox і CSS Grid? У яких випадках краще використовувати кожен із них?
19. Як працюють властивості `transition` та `animation` у CSS? Наведіть приклад.
20. Чому важливо відокремлювати структуру (HTML) від стилів (CSS)? Які переваги це дає при створенні сайтів?

ЛАБОРАТОРНА РОБОТА № 4

ПОЗИЦІОНУВАННЯ, FLEXBOX ТА GRID ДЛЯ ОФОРМЛЕННЯ ВЕБСТОРИНОК

Мета роботи: Закріпити навички верстки вебсторінок за допомогою сучасних можливостей CSS, зокрема властивостей позиціонування елементів, засобів вирівнювання та розташування контенту (Flexbox і Grid Layout), а також принципів адаптивного дизайну. Ознайомитися з використанням псевдокласів та псевдоелементів для створення інтерактивних ефектів і покращення візуального сприйняття сторінки.

Короткі теоретичні відомості

Веброзробка базується на чіткій організації структури сторінки, де кожен елемент має своє місце і спосіб відображення. Однією з ключових властивостей, що визначає, як саме елемент поводить себе у потоці документа, є властивість `display`. Вона визначає, чи буде елемент займати цілий рядок, розташовуватись поруч з іншими або взагалі не відображатись. Саме розуміння цієї властивості дозволяє створювати гнучкі й структуровані макети вебсторінок.

Тип відображення `block` використовується для елементів, які займають всю доступну ширину батьківського контейнера. Прикладом таких елементів є `<div>`, `<p>`, `<header>` чи `<section>`. Кожен блоковий елемент починається з нового рядка, що дозволяє будувати сторінку у вигляді чітких вертикальних структур. Це зручно для створення великих секцій або контейнерів, у яких можна розміщувати інші елементи.

На відміну від цього, властивість `inline` робить елемент стрічковим, тобто таким, що розташовується в одному рядку з іншими. При цьому елемент не перериває потік тексту і не починається з нового рядка. Прикладами таких елементів є `<span>`, `<a>` або `<strong>`. Вони підходять для оформлення невеликих частин тексту або вставки інтерактивних елементів без зміни загальної структури сторінки.

Компромісом між двома попередніми типами є значення `inline-block`. Елементи з таким відображенням поєднують властивості обох типів: вони можуть стояти поруч в одному рядку, як стрічкові, але при цьому поведуться як блокові – можна задавати ширину, висоту, відступи та інші параметри. Це особливо зручно при створенні карток або кнопок, які мають бути на одному рівні, але при цьому мати власне форматування. Існує також значення `none`, яке повністю прибирає елемент із візуального потоку сторінки. Такий елемент не займає місця і не впливає на інші об'єкти. Це часто використовується для приховування елементів, які можуть з'являтися пізніше за допомогою JavaScript або CSS-анімацій, наприклад, у випадку випадаючих меню чи модальних вікон.

У вебдизайні часто виникає потреба розташувати елементи не лише один під одним, а й так, щоб вони частково перекривалися. Для цього використовуються властивості `position` та `z-index`, які дозволяють керувати просторовим розташуванням елементів на сторінці. Зазвичай, коли елементи перекриваються, важливо визначити, який з них має бути зверху, а який – під ним. Саме це й контролюється за допомогою `z-index` – властивості, що задає “глибину” шару елемента у візуальному просторі сторінки.

Значення `z-index` приймає цілі числа: чим вище значення, тим ближче елемент розташований до користувача. Якщо три блоки мають різні `z-index` – наприклад, 1, 2 і 3 – то блок із найбільшим значенням буде зверху, навіть якщо в HTML-коді він іде останнім. Проте, щоб ця властивість працювала, елемент повинен мати встановлене позиціонування (`position: relative, absolute, fixed` або `sticky`). Без цього `z-index` не впливатиме на порядок накладання.

Для кращої візуалізації ефекту перекриття використовують тінь (`box-shadow`) або прозорість (`opacity`). Завдяки тіням можна створити ілюзію глибини, а за допомогою прозорості – побачити нижні елементи крізь верхні. Поєднання цих ефектів допомагає розробникам зрозуміти принципи накладання елементів і створювати більш реалістичні, об’ємні інтерфейси. Такі прийоми широко використовуються в дизайні карток, спливаючих вікон, меню чи анімацій.

Властивість `position` у CSS використовується для керування тим, як саме елементи розташовуються на вебсторінці. Вона дозволяє виводити блоки не лише в потоці документа, а й змінювати їх положення відносно інших елементів або меж екрана. Це одна з ключових властивостей для побудови складних макетів і створення ефектів перекриття, фіксації чи точного позиціонування.

Значення `static` є стандартним для всіх елементів – вони розміщуються у потоці документа одне під одним, відповідно до порядку в коді. Коли використовується `relative`, елемент залишається у потоці, але його можна зсунути на певну відстань за допомогою властивостей `top, bottom, left, right`. Це зручно, коли потрібно трохи змінити положення елемента без порушення загальної структури сторінки.

Значення `absolute` дозволяє повністю вивести елемент з потоку документа. Такий елемент позиціонується відносно найближчого батьківського блоку з властивістю `position: relative`. Це дає змогу розміщувати елементи поверх інших або у точному місці в межах певного контейнера.

Значення `fixed` фіксує елемент відносно вікна браузера. Це означає, що він залишається на одному місці навіть при прокручуванні сторінки. Такий тип позиціонування часто використовують для навігаційних панелей, кнопок або декоративних елементів.

Також існує значення `sticky`, яке поєднує поведінку `relative` і `fixed`. Елемент спочатку рухається разом із потоком документа, але коли користувач прокручує сторінку до певного місця, він "прилипає" до заданої позиції. Комбінація різних типів позиціонування дозволяє створювати складні й динамічні макети, забезпечуючи точне керування простором і взаємним розташуванням елементів на сторінці.

Властивість `float` у CSS використовується для вирівнювання елементів ліворуч або праворуч у межах контейнера, дозволяючи іншим елементам – зазвичай тексту – обтікати їх. Спочатку ця властивість застосовувалася для верстки сторінок із зображеннями, які мали бути обтікані текстом, але згодом стала популярним інструментом для створення макетів ще до появи Flexbox та Grid. Значення `float: left` або `float: right` дозволяють розмістити блок з однієї зі сторін, а текст або інші елементи заповнюють простір поруч із ним.

Коли елемент "спливає" за допомогою `float`, він виводиться з нормального потоку документа, тому інші блоки можуть ігнорувати його висоту. Через це іноді виникає ситуація, коли батьківський контейнер не враховує розміри своїх дочірніх елементів, які мають `float`. Щоб уникнути цього, використовують властивість `clear`, яка скидає обтікання й відновлює нормальний потік. Наприклад, `clear: both` гарантує, що елемент відображатиметься під усіма попередніми блоками з `float`, незалежно від того, з якого боку вони були вирівняні.

Використання `float` потребує уважності, оскільки його легко сплутати з інструментами сучасного вирівнювання. Сьогодні `float` частіше застосовується для невеликих декоративних елементів або текстового обтікання зображень, але не для побудови цілої структури сторінки – для цього краще використовувати сучасніші системи верстки, такі як Flexbox або Grid.

Flexbox (Flexible Box Layout) – це сучасна модель верстки, яка створена для гнучкого розташування елементів у рядок або стовпчик. Основна перевага Flexbox полягає в тому, що він автоматично розподіляє вільний простір між елементами та дозволяє легко змінювати вирівнювання, напрямок і порядок відображення елементів без потреби використовувати `float` чи `position`.

Основні властивості Flexbox – це `justify-content` та `align-items`. Вони відповідають за вирівнювання елементів по головній та поперечній осях. Наприклад, `justify-content: center` вирівнює всі елементи по центру головної осі (зазвичай горизонтально), `space-between` рівномірно розподіляє елементи з відступами між ними, а `space-around` додає простір як між елементами, так і навколо них.

Властивість `align-items` контролює вертикальне вирівнювання у flex-контейнері. Її значення `center` розташовує всі елементи по вертикалі по центру, `flex-start` – вирівнює по верхньому краю, а `flex-end` – по нижньому. Це особливо зручно, коли потрібно створити гармонійно вирівняні кнопки

або елементи інтерфейсу, навіть якщо вони мають різну висоту чи довжину.

Таким чином, `float` і `flex` вирішують схожі завдання – керують розташуванням елементів на сторінці, але підходять для різних ситуацій. `Float` більше підходить для обтікання контенту, тоді як `Flexbox` – для сучасної, адаптивної та динамічної верстки. Використання `Flexbox` дає змогу створювати чисту, передбачувану структуру, де вирівнювання й відстані контролюються кількома властивостями, а не складними обхідними методами.

`CSS Grid Layout` – це потужна система двовимірного компоновання, яка дозволяє розташовувати елементи як по горизонталі, так і по вертикалі. На відміну від `Flexbox`, який працює лише в одному напрямку (рядок або стовпчик), `Grid` надає можливість створювати складні сітки з точно контрольованим розміщенням елементів у рядках і колонках. Це робить його ідеальним інструментом для побудови повних макетів сторінок або секцій із фіксованою або гнучкою структурою.

У системі `Grid` використовуються властивості `display: grid` або `display: inline-grid`, які перетворюють контейнер на сітку. Далі за допомогою властивостей `grid-template-columns` та `grid-template-rows` визначається кількість і розміри колонок і рядків. Наприклад, можна задати три колонки однакової ширини за допомогою запису `grid-template-columns: repeat(3, 1fr);`, де `1fr` означає частку доступного простору.

`Grid` також дозволяє задавати відстані між елементами за допомогою властивості `gap`, що замінює потребу у використанні відступів (`margin`) між блоками. Це робить код чистішим і забезпечує більш точне керування відстанями. Крім того, можна керувати розміщенням елементів за допомогою властивостей `grid-column` і `grid-row`, які визначають, через скільки стовпців або рядків має тягнутися певний елемент.

Однією з найсильніших сторін `Grid` є його здатність до адаптивності. Використовуючи відносні одиниці вимірювання (наприклад, `fr`, `%`, `minmax()`), можна створювати макети, що автоматично підлаштовуються під ширину екрана. Наприклад, властивість `grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));` дозволяє створити сітку, у якій елементи будуть перебудовуватись залежно від розміру вікна браузера – і це робить верстку по-справжньому гнучкою.

Важливою концепцією у `Grid` є вирівнювання елементів, яке здійснюється за допомогою властивостей `justify-items`, `align-items`, `justify-content` та `align-content`. Вони дозволяють точно позиціонувати елементи всередині сітки – наприклад, розташувати всі блоки по центру або вирівняти їх по верхньому краю. Це особливо корисно для створення гармонійних інтерфейсів, де кожен елемент має своє логічне місце.

Адаптивна верстка (`responsive design`) – це підхід до веброзробки, за якого сторінка автоматично змінює своє відображення залежно від розміру екрана або пристрою. Основна мета адаптивної верстки – забезпечити

зручність перегляду сайту як на великому моніторі, так і на смартфоні. Для цього використовуються гнучкі розміри елементів, відсоткові значення ширини, сітки, а також спеціальні інструменти, такі як медіа-запити.

Медіа-запити (media queries) – це правило CSS, яке дозволяє задавати різні стилі для різних пристроїв або розмірів вікна. Наприклад, можна написати @media (max-width: 768px) і змінити структуру сторінки, щоб усі елементи відображалися у стовпчик замість ряду. Це дозволяє уникати горизонтальної прокрутки на телефонах і робить інтерфейс зручним для користувача.

Адаптивність також досягається завдяки використанню гнучких одиниць, таких як vw, vh, % та em, які змінюються пропорційно до розміру екрана або батьківського елемента. Наприклад, зображення з властивістю max-width: 100% автоматично зменшиться разом із контейнером, не виходячи за його межі.

Сучасна практика вебдизайну часто поєднує Grid і Flexbox для досягнення максимальної адаптивності. Наприклад, Grid може визначати загальну структуру сторінки (шапка, контент, підвал), а Flexbox – розташування елементів у межах цих секцій. Такий підхід дозволяє створювати макети, які виглядають однаково добре на будь-якому пристрої.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ РОБОТИ

1. Створіть HTML-сторінку, на якій продемонструйте роботу різних значень властивості display. Створіть кілька елементів (наприклад, кольорові блоки або картки з текстом) і застосуйте до них значення: block, inline, inline-block, none.

У звіт додайте коротке пояснення (під кодом кожного елемента), що описує його поведінку.

2. Створіть три блоки, які частково перекривають один одного. Задайте кожному різний **z-index**, щоб визначити порядок накладання. Додайте тінь або прозорість (opacity), щоб чітко бачити, як працює накладання.

3. Створіть секцію Hero з фоновим зображенням. Додайте текст і кнопку, які будуть **поверх зображення**. Використайте **position: relative** для батьківського блоку та **absolute** – для розміщення елементів. Додайте декоративний елемент (іконку чи коло), який має **fixed position** і завжди залишається в одному місці при прокрутці сторінки.

4. Створіть контейнер з фіксованою висотою (наприклад, 200px) та вставте в нього довгий текст або зображення. Використайте різні варіанти **overflow** (hidden, scroll, auto) і продемонструйте різницю в поведінці елемента.

5. Створіть макет, де зображення зліва або справа обтікається текстом. Використайте властивість **float** для вирівнювання зображення. Додайте **clear**, щоб відновити потік документа.

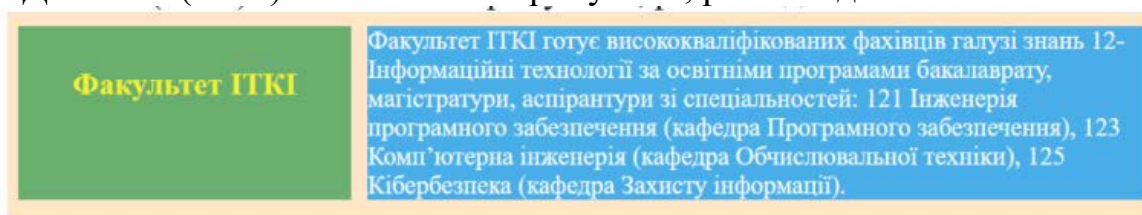
6. Створіть секцію з кількома кнопками різної довжини. Вирівняйте їх горизонтально або вертикально за допомогою **justify-content** та **align-items** (у flex-контейнері). Перевірте різні значення вирівнювання (center, space-between, space-around).

7. Створіть навігаційне меню з пунктами «Головна», «Про мене», «Контакти». Додайте ефекти наведення (:hover, :active, :focus) для кожного пункту. Використайте псевдоелементи ::before або ::after для декоративної лінії або іконки біля пункту меню.

8. Створіть сторінку з двома секціями – бічною панеллю і основним контентом, у яких однакова висота, незалежно від розміру контенту. Завдання виконати трьома способами, використовуючи варіанти значень властивості display:

1) table; 2) flex; 3) grid.

Для aside (зліва) встановити ширину 35%, решта – для main.



9. Напишіть розмітку grid для варіанту макета, що показано на рисунку.



Додаткове завдання.

10*. Адаптивна верстка

Створіть невеликий сайт або лендінг (3–4 секції) і додайте **медіа-запити** (@media) для різних ширин екрана.

На мобільному екрані елементи мають розташовуватись у стовпчик, меню – перетворюватися на бургер-іконку, зображення – масштабуватись. Перевірте відображення при ширині 1200px, 768px, 480px.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке властивість `display` і які основні її значення використовуються в CSS?
2. У чому різниця між `display: block`, `display: inline` та `display: inline-block`?
3. Для чого використовується властивість `position` і які є типи позиціонування елементів у CSS?
4. Як працює `position: absolute` і відносно чого позиціонується елемент?
5. Поясніть відмінність між `position: fixed` і `position: sticky`.
6. Що визначає властивість `z-index` і коли вона має ефект?
7. Для чого використовується властивість `overflow`? Назвіть її основні значення.
8. Як працює властивість `float` і яке призначення має `clear`?
9. Що таке Flexbox і для чого він використовується у верстці?
10. Які основні властивості керують вирівнюванням елементів у flex-контейнері?
11. У чому перевага `justify-content: space-between` порівняно з `center` у Flexbox?
12. Що таке CSS Grid і чим він відрізняється від Flexbox?
13. Для чого використовуються властивості `grid-template-columns` і `grid-template-rows`?
14. Що таке адаптивна верстка і для чого застосовуються медіа-запити (`@media`)?
15. Як правильно тестувати адаптивність сторінки під різні розміри екранів?
16. Створіть три прямокутники, які перекривають один одного, і задайте їм різний `z-index`, щоб визначити порядок відображення.
17. Зробіть блок із текстом і зображенням, де текст обтікає картинку за допомогою властивості `float`.
18. Створіть контейнер із трьома кнопками різної довжини та вирівняйте їх по центру за допомогою Flexbox.
19. Використайте Grid для побудови сітки з трьох колонок і двох рядків. Задайте однакові відступи між елементами (gap).
20. Реалізуйте простий приклад адаптивного макета: при ширині екрана до 600px усі блоки розташовуються у стовпчик, а при більшій ширині – у ряд.

ЛАБОРАТОРНА РОБОТА № 5

РЕАЛІЗАЦІЯ АДАПТИВНОЇ ВЕБСТОРІНКИ ЗА ІНДИВІДУАЛЬНИМ ДИЗАЙН-МАКЕТОМ

Мета роботи: Закріпити навички створення вебсторінок шляхом відтворення інтерфейсу за попередньо розробленим дизайн-макетом. Опрацювати сучасні можливості HTML і CSS для реалізації структури та стилізації сторінки, зокрема використання Flexbox і Grid Layout, адаптивної верстки, типографіки та кольорових рішень відповідно до макета. Закріпити навички роботи з псевдокласами, псевдоелементами та медіа-запитами з метою покращення інтерактивності та коректного відображення сторінки на різних пристроях.

Короткі теоретичні відомості

Верстка вебсторінки є процесом, у якому графічний дизайн перетворюється на робочий вебінтерфейс. Під час цього розробник має враховувати логіку структури, взаємодію елементів, адаптацію під різні пристрої та відповідність початковому макету.

HTML виступає основою будь-якої вебсторінки й визначає її структурний скелет. Завдяки HTML формується логічна ієрархія елементів, що згодом впливає на їхню стилізацію та поведінку. Саме через розмітку задається зміст, заголовки, списки, зображення та кнопки.

Семантична складова HTML відіграє важливу роль, оскільки правильне використання тегів допомагає формувати інтерфейс, зрозумілий як користувачам, так і браузерам. Семантичні елементи дозволяють упорядкувати структуру та зробити її передбачуваною.

У сучасній розробці все більше уваги приділяється доступності, і семантичні елементи сприяють тому, щоб різні допоміжні технології могли правильно читати вебсторінку. Це підвищує якість взаємодії з інтерфейсом.

CSS є інструментом, що відповідає за зовнішній вигляд сторінки, надаючи розробнику повний контроль над візуальною частиною інтерфейсу. Стилі дозволяють визначити кольори, шрифти, проміжки та інші декоративні параметри. Однією з базових властивостей CSS є каскадність, завдяки якій стильові правила можуть накладатися одне на одне. Це створює гнучку систему пріоритетів, яку важливо враховувати при написанні стилів.

Важливе місце займає успадкування, яке дозволяє окремим властивостям поширюватися на дочірні елементи без додаткового дублювання коду. Це робить стилізацію більш структурованою та легко підтримуваною. Специфічність у CSS визначає силу конкретного правила, що впливає на те, які стилі застосовуватимуться в разі конфліктів.

Розуміння специфічності дозволяє уникати непередбачуваних візуальних результатів. Окрему групу формують властивості, відповідальні за тип відображення елементів. Знання поведінки блокових, рядкових та комбінованих елементів допомагає точніше керувати компонованням сторінки.

Flexbox є потужним інструментом для створення гнучких інтерфейсів, адже він дозволяє контролювати вирівнювання, напрямок і розподіл елементів у контейнері. Цей механізм ідеально підходить для горизонтальних та вертикальних структур. Flexbox спрощує адаптивну верстку, тому що елементи можуть змінювати свої розміри залежно від простору, що доступний. Це забезпечує природну поведінку інтерфейсу без складних розрахунків.

CSS Grid пропонує двовимірний підхід до побудови макетів, дозволяючи створювати сітки будь-якої складності. Розробник може визначати точну кількість колонок, рядків та їх співвідношення. Grid дає можливість працювати як із простими макетами, так і з багаторівневими інтерфейсами, де потрібне точне позиціонування елементів. Це робить його незамінним для реалізації складних дизайн-макетів.

Положення елементів на сторінці може контролюватися через систему позиціонування, яка дозволяє визначати фіксовані, відносні або абсолютні координати. Завдяки цьому можна створювати ефекти накладання та плаваючих блоків.

Позиціонування корисне для створення унікальних декоративних елементів, таких як іконки, підказки або кнопки, що мають відхилятися від стандартних потоків розташування.

Псевдокласи CSS дозволяють реагувати на дії користувача, зокрема наведення, фокусування або взаємодію з формами. Вони роблять інтерфейс живим і динамічним без необхідності використовувати JavaScript.

Псевдоелементи розширюють можливості дизайну, адже дозволяють створювати додаткові графічні вкраплення, не змінюючи HTML-код. Це підвищує гнучкість у роботі з оформленням.

Адаптивна верстка – це підхід до створення вебінтерфейсів, при якому сторінка автоматично підлаштовується під розміри екрана та характеристики пристрою. Це означає, що один і той самий сайт повинен однаково зручно сприйматися на смартфоні, планшеті, ноутбуці й великому моніторі, хоч кожен із них має різну ширину екрана, щільність пікселів і спосіб взаємодії.

Основна ідея адаптивності полягає в тому, щоб інтерфейс залишався зрозумілим та зручним незалежно від того, на якому пристрої його переглядають. Важливим аспектом є не лише перестановка елементів, а й підбір правильних розмірів шрифтів, проміжків, картинок і кнопок. Особливо важливо, щоб мобільна версія була легкою, адже саме з телефонів сьогодні заходить більшість користувачів.

Адаптивний дизайн традиційно поділяється на кілька підходів. Перший – це **Responsive Design** (відгукливий дизайн), який передбачає плавне підлаштування макета залежно від ширини екрана. Такий сайт не має жорстко визначених "мобільної" чи "десктопної" версій – він розтягується, перебудовується і змінює структуру в режимі реального часу.

Другий підхід – **Adaptive Design**, у якому сайт має набори окремих фіксованих макетів під різні розміри екранів. Наприклад, одна версія для 320px, інша для 768px, третя для 1024px тощо. Браузер просто обирає найкращий варіант. Такий підхід зустрічається рідше, але все ще використовується.

У сучасній розробці найбільш популярним є саме відгукливий дизайн, оскільки він універсальний, плавний і добре комбінується з гнучкими одиницями вимірювання. Саме його ви реалізуєте в своїх лабораторних роботах.

Щоб сайт був адаптивним, розробник має використовувати правильну структуру HTML і гнучкі можливості CSS. Важливу роль відіграють гнучкі сітки, які можна будувати засобами Flexbox або Grid. Flexbox забезпечує адаптивне вирівнювання елементів у одному напрямку – наприклад, картки можуть автоматично переноситися на наступний рядок. Grid дозволяє змінювати кількість колонок залежно від ширини екрана, що робить макет справді універсальним.

Не менш важливим елементом адаптивності є **гнучкі одиниці вимірювання**, такі як відсотки, `vh`, `vw` і `rem`. Вони замінюють жорсткі піксельні величини й дозволяють елементам природно змінювати свої пропорції. Наприклад, блок шириною 80% завжди займатиме цю частину екрана, незалежно від того, який це пристрій.

Ключова техніка адаптивності – це **медіа-запити** (media queries). Вони дозволяють застосовувати різні стилі залежно від ширини вікна браузера. Наприклад, на широких екранах меню може бути горизонтальним, а на мобільних – перетворюватися на «бургер-іконку». Медіа-запити можуть регулювати не лише розміри, а й приховувати або спрощувати деякі елементи, щоб інтерфейс не перевантажувався.

При побудові адаптивного макета важливо дотримуватися принципу «Mobile First». Це означає, що спочатку розробляється проста, легка мобільна версія, а вже потім, у міру розширення екрана, додаються нові елементи або ускладнюється розмітка. Такий підхід забезпечує найкращу продуктивність, оскільки мобільний пристрій отримує лише те, що дійсно потрібно.

Крім цього, адаптивність стосується і графічного контенту. Зображення не повинні виходити за межі контейнера, а відео мають підлаштовуватися під пропорції екрана. Для цього використовують властивості `max-width: 100%` і обгортки з фіксованими співвідношеннями сторін, які дозволяють медіа коректно масштабується.

Шрифти також повинні реагувати на зміну ширини екрана. У сучасній верстці використовується підхід *fluid typography*, коли розмір тексту плавно змінюється. Це досягається через функцію `clamp` або одиниці `vw`. Такий текст не буде ні занадто дрібним на великих екранах, ні занадто великим на телефонах.

Важливим аспектом адаптивності є інтерфейсні елементи – кнопки, форми, меню та інтерактивні блоки. На мобільних пристроях кнопки повинні бути більшими, відступи – ширшими, а текст – читабельнішим. Це забезпечує комфортне користування пальцями, а не мишею. Неправильно підібраний розмір торкання може суттєво погіршити досвід користувача.

Важливо також думати про адаптивність змісту. Якщо на широкому екрані можна розмістити три колонки тексту, то на мобільному це буде незручно. Тому контент повинен перебудовуватися в один стовпець, щоб читання було природним і без прокрутки вбік.

Меню навігації є одним із найскладніших елементів адаптивності. На мобільних екранах немає місця для традиційних горизонтальних списків, тому вони перетворюються в приховану панель, яка відкривається через іконку. Це дозволяє зберегти повну функціональність без зайвого перевантаження інтерфейсу.

Під час адаптивної верстки важливо стежити за тим, щоб елементи не перекривалися, не накладалися одне на одного і не виходили за межі екрана. Регулярна перевірка на різних розмірах вікна допомагає помітити потенційні проблеми ще до того, як сайт буде переглядатися користувачами. Сучасні інструменти, такі як DevTools у браузері, дозволяють швидко перевіряти вигляд сторінки на різних пристроях. Це дає змогу виявити помилки адаптивності та коректно налаштувати медіа-запити. Крім ширини екрана, адаптивність може враховувати й інші параметри, такі як орієнтація (портретна або ландшафтна), щільність пікселів і рівень масштабування. Це дозволяє створити інтерфейс, що точно відповідає можливостям конкретного пристрою.

Окрему увагу потрібно приділити продуктивності. Адаптивні сайти повинні завантажуватися швидко, навіть на слабких смартфонах. Для цього зменшують вагу зображень, оптимізують шрифти та мінімізують CSS-файли.

Адаптивність стосується не лише статичних елементів, а й інтерактивних. Анімації, що добре виглядають на великому екрані, можуть бути надто важкими для мобільних пристроїв. Тому важливо враховувати продуктивність і плавність.

Сучасні фреймворки, такі як Bootstrap і Tailwind, пропонують вбудовані рішення для адаптивності. Вони дозволяють швидше створювати адаптивні компоненти, проте все одно важливо розуміти базові принципи, щоб використовувати їх правильно.

Правильна адаптивність робить сайт не лише красивим, а й доступним. Це особливо важливо для людей із порушеннями зору, які користуються збільшеним масштабом або спеціальними пристроями.

У підсумку адаптивний дизайн – це комплексний підхід, який включає продуману структуру, гнучкі одиниці вимірювання, медіа-запити, оптимізацію графіки та інтерфейсу. Це не просто набір технічних прийомів, а повноцінна філософія розробки.

Добре реалізована адаптивність забезпечує комфортний перегляд сайту будь-де: вдома за комп'ютером, у транспорті з телефона чи на планшеті під час навчання. Такий сайт залишається зручним, приємним і зрозумілим для кожного користувача.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ РОБОТИ

1. На основі створеного дизайну в сервісі Figma (згідно завдання до лабораторної роботи №2) розробити повноцінну вебсторінку, дотримуючись затвердженого макета та кольорової схеми. Створити HTML-структуру, що семантично відповідає змісту сторінки.

2. Виконати стилізацію сторінки за допомогою CSS, застосувавши сучасні технології верстки – Flexbox та CSS Grid – для побудови адаптивних блоків, модульних сіток та коректного вирівнювання елементів.

3. Забезпечити адаптивність сторінки для різних розмірів екранів (мобільні пристрої, планшети, десктопи). Реалізувати адаптивну типографіку, масштабування зображень та перебудову макета через медіазапити.

4. Додати декоративні елементи за допомогою псевдокласів та псевдоелементів (:hover, :focus, ::before, ::after).

5. Провести тестування створеної сторінки: перевірити коректність відображення у різних браузерях, адаптивність у різних режимах перегляду, відповідність вихідному дизайну та валідність HTML/CSS-коду. Це можна зробити за допомогою офіційних валідаторів, таких як W3C HTML Validator та W3C CSS Validator. Надати висновки за результатами виконаної роботи.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке семантична структура HTML і чому вона важлива при створенні вебсторінок?

2. Для чого використовують Flexbox і у яких випадках він є найзручнішим інструментом?

3. Які основні відмінності між Flexbox та CSS Grid?

4. Що таке медіазапити і як вони допомагають у створенні адаптивного дизайну?
5. Які типи адаптивності існують та чим відрізняється респонсивний дизайн від адаптивного?
6. Як правильно організувати сітку сторінки за допомогою Grid Layout?
7. Які принципи потрібно враховувати при розробці мобільної версії сайту?
8. Що таке «mobile-first» підхід і які його переваги?
9. Як забезпечити коректне масштабування зображень у адаптивній верстці?
10. Для чого використовують псевдокласи :hover, :focus, :active?
11. Які можливості надають псевдоелементи ::before та ::after?
12. Чому важливо тестувати сторінку у різних браузерах?
13. Які інструменти браузера допомагають перевірити адаптивність сторінки?
14. Що таке валідація HTML та CSS та для чого вона проводиться?
15. Назвіть основні помилки, які може показати HTML-валідатор.
16. Які властивості CSS найчастіше використовуються для вирівнювання елементів у Flexbox?
17. Як працює властивість grid-template-columns і для чого вона потрібна?
18. Що таке брейкпоінти та як правильно їх вибирати для макета?
19. Чому важливо дотримуватися дизайну, створеного на попередньому етапі роботи?
20. Які показники якості верстки дозволяють оцінити успішність виконання лабораторної роботи?

ЛАБОРАТОРНА РОБОТА № 6

РЕАЛІЗАЦІЯ ІНТЕРАКТИВНИХ ЕЛЕМЕНТІВ НА ВЕБСТОРІНЦІ

Мета роботи: Ознайомитися з основами додавання інтерактивності до вебсторінок та навчитися реалізовувати динамічну поведінку елементів інтерфейсу за допомогою JavaScript. Закріпити навички роботи з DOM, подіями користувача, удосконаливши готовий макет вебсторінки з попередньої лабораторної роботи.

Короткі теоретичні відомості

Сучасні вебтехнології дозволяють створювати інтерактивні вебсторінки, які реагують на дії користувача, змінюють свій зміст та вигляд у реальному часі. Така інтерактивність значно підвищує зручність використання сайту і робить його більш привабливим для відвідувачів. Взаємодія користувача з вебсторінкою стає не просто переглядом інформації, а активною участю, що особливо важливо для персональних сайтів, портфоліо та навчальних ресурсів.

Вебсторінки формуються на основі трьох основних технологій: HTML, CSS та JavaScript. HTML визначає структуру документа і містить усі основні елементи сторінки, такі як заголовки, абзаци, списки, зображення, посилання та форми. Кожен елемент HTML можна розглядати як об'єкт, який можна модифікувати за допомогою стилів та скриптів.

CSS відповідає за оформлення вебсторінки, визначає кольори, шрифти, розміри, відступи та розташування елементів на сторінці. Використання CSS дозволяє створювати естетично привабливі вебресурси і забезпечує послідовність дизайну, що важливо для комфортного сприйняття інформації користувачем. Крім того, CSS підтримує медіа-запити, які дозволяють робити сторінку адаптивною, тобто коректно відображати її на пристроях з різними розмірами екранів.

JavaScript додає вебсторінкам динамічну поведінку. Ця мова програмування виконується безпосередньо у браузері користувача і дозволяє змінювати елементи HTML та CSS під час роботи сторінки. За допомогою JavaScript можна реалізувати логіку роботи форм, реагування на події користувача, анімацію, перемикання елементів та багато інших функцій, що роблять сайт інтерактивним.

Інтерактивна галерея зображень, або слайдер, є популярним елементом сучасних вебсторінок. Вона дозволяє відвідувачу переглядати кілька фотографій чи ілюстрацій, перемикаючись між ними за допомогою кнопок або інших елементів керування. Основною перевагою слайдера є те, що зміна зображень відбувається **без перезавантаження сторінки**, завдяки динамічній роботі з DOM за допомогою JavaScript.

Для реалізації слайдера спочатку необхідно підготувати HTML-контейнер, в якому буде розміщене зображення та елементи керування. Зазвичай використовують блок `<div>` як контейнер, всередині якого розташовують елемент `<img>` для поточного зображення та кнопки «Попереднє» та «Наступне». Це дозволяє користувачу перемикатися між зображеннями і забезпечує логічну структуру сторінки.

Після створення контейнера необхідно в JavaScript визначити масив зображень, які будуть відображатися у слайдері. Кожне зображення має бути представлене шляхом до файлу, а також потрібна змінна для збереження індексу поточного зображення. Це дозволяє відстежувати, яке зображення зараз показується на сторінці, і переміщатися між елементами масиву.

Для перемикання зображень створюють функцію `showImage(index)`, яка змінює атрибут `src` елемента `<img>` на поточний елемент масиву. Кнопки «Попереднє» та «Наступне» отримують обробники подій, які змінюють значення `currentIndex` та викликають функцію для оновлення зображення. При досягненні кінця масиву або початку, слайдер може зациклюватися, щоб користувач міг безперервно переглядати картинки.

Основою взаємодії JavaScript з HTML-документом є об'єктна модель документа (DOM). DOM представляє вебсторінку у вигляді ієрархічного дерева об'єктів, де кожен елемент HTML є окремим об'єктом. Це дозволяє програмно змінювати текст, стилі, атрибути елементів, додавати або видаляти об'єкти та обробляти події користувача без перезавантаження сторінки.

Однією з популярних форм інтерактивності є використання форм та анкет на вебсторінках. Анкета-тест дозволяє користувачеві відповідати на запитання, після чого сторінка може аналізувати відповіді і виводити результати. Така взаємодія забезпечує активну участь відвідувача і може бути використана для навчальних або персональних цілей, наприклад, для тестування знань або збору інформації про вподобання.

Реалізація анкети часто передбачає поетапне відображення запитань. Користувач бачить лише одне запитання за раз, що підвищує зручність сприйняття інформації. Після вибору відповіді користувач може переходити до наступного питання, а всі відповіді зберігаються у змінних або масивах, що дозволяє обробляти їх після завершення опитування.

Обробка подій є ключовим аспектом інтерактивності. Події виникають у відповідь на дії користувача, такі як клік, наведення курсора, введення тексту або прокрутка сторінки. Обробники подій у JavaScript дозволяють виконувати певний код при виникненні події, що забезпечує динамічну взаємодію зі сторінкою та зміни в її вмісті.

Зміна стилів сторінки за допомогою JavaScript є ще одним прикладом інтерактивності. Користувач може змінювати колір фону, тему оформлення або зовнішній вигляд окремих елементів сторінки.

Найзручнішим способом реалізації такої функції є додавання або видалення CSS-класів, що дозволяє зберігати стилі окремо від логіки роботи сторінки. Інтерактивна галерея зображень є популярним елементом персональних вебсайтів. Вона дозволяє демонструвати фотографії, ілюстрації або будь-які графічні матеріали та забезпечує зручне перемикання між ними. Галерея зазвичай реалізується з використанням масиву зображень та індексу поточного елемента, який змінюється при натисканні кнопок керування. Перемикання зображень без перезавантаження сторінки забезпечує плавну взаємодію і покращує користувацький досвід. Додаткові анімації та CSS-переходи роблять перемикання більш естетичним і привабливим, а також надають сторінці сучасного вигляду.

Адаптивний дизайн є важливим аспектом сучасної веброзробки. Він дозволяє вебсторінці коректно відображатися на пристроях з різною шириною екрану, включаючи настільні комп'ютери, планшети та мобільні телефони. Адаптивність підвищує доступність сайту та зручність його використання. Медіа-запити CSS (@media) дозволяють змінювати стилі сторінки залежно від ширини екрану або типу пристрою. За допомогою медіа-запитів можна налаштувати розміри шрифтів, блоків, зображень та розташування елементів, що робить сторінку зручною для користувача на будь-якому пристрої.

Оптимізація зображень є невід'ємною частиною адаптивної веброзробки. Великі файли можуть значно уповільнювати завантаження сторінки, особливо на мобільних пристроях. Використання відповідних форматів, стиснення та адаптивних розмірів дозволяє забезпечити швидку роботу сайту. Перевірка роботи вебсторінки при зміні розміру вікна браузера або на різних пристроях є обов'язковим етапом розробки. Це дозволяє виявити помилки у верстці та інтерактивності та забезпечити коректну роботу всіх елементів.

Комплексне поєднання інтерактивності та адаптивності створює зручний та сучасний вебінтерфейс, що підвищує залученість користувачів і покращує загальне враження від вебсайту.

Персональні вебсайти, такі як сайт «Про себе», є ефективним способом представлення інформації про автора. Інтерактивні елементи, такі як анкети, галереї та кнопки для зміни стилю, роблять сайт більш живим і цікавим для відвідувачів.

Використання інтерактивних функцій також дозволяє демонструвати практичні навички роботи з вебтехнологіями, зокрема з JavaScript та DOM, що є важливим елементом навчального процесу у сфері інформаційних технологій.

Використання медіа-запитів та адаптивної верстки забезпечує універсальність вебсайту і можливість його коректного відображення на будь-яких пристроях. Це є важливою вимогою сучасної веброзробки і дозволяє охопити широку аудиторію користувачів.

Комплексне застосування інтерактивних елементів, адаптивної верстки, оптимізації зображень та обробки подій формує високоякісний персональний вебресурс, який поєднує зручність, естетику та функціональність.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ РОБОТИ

1. На основі вебсторінки, розробленої у лабораторній роботі № 5, реалізувати окремий інтерактивний блок анкети-тесту та додати його до готової вебсторінки. Кількість запитань анкети має бути не менше 10. Програма має поетапно виводити запитання анкети, дозволяючи користувачу обирати варіанти відповідей, та після завершення опитування, відображати результати на сторінці. Тематика запитань анкети повинна відповідати загальній тематиці вебсторінки та розкривати її зміст.

2. Додати кілька кнопок для зміни кольорової теми сайту. Реалізувати можливість перемикання між світлою та темною темами оформлення. При натисканні на відповідну кнопку фон сторінки та основні елементи інтерфейсу повинні змінюватися відповідно до обраної теми без перезавантаження сторінки.

3. Створити інтерактивну галерею зображень, яка повинна забезпечувати можливість перемикання зображень без перезавантаження сторінки за допомогою елементів керування.

4. Забезпечити коректне відображення вебсторінки на пристроях з різною шириною екрана (настільні комп'ютери, планшети, мобільні телефони). Реалізувати адаптивну верстку з використанням медіа-запитів (@media). Оптимізувати розміри зображень та елементів інтерфейсу для зменшення часу завантаження сторінки. Перевірити коректність роботи всіх інтерактивних елементів після зміни розмірів вікна браузера.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке DOM і яку роль він відіграє у створенні інтерактивних вебсторінок?

2. Які способи підключення JavaScript до HTML-документа ви знаєте?

3. Для чого використовуються обробники подій у JavaScript?

4. Які події найчастіше застосовуються для взаємодії користувача зі сторінкою?

5. Як реалізувати поетапне відображення запитань в анкеті без перезавантаження сторінки?

6. У чому різниця між радіо-кнопками (radio) та прапорцями (checkbox)?

7. Як отримати значення, обране користувачем у формі?

8. Яким чином можна підрахувати результати анкети-тесту за допомогою JavaScript?
9. Що таке динамічна зміна стилів сторінки і як вона реалізується?
10. Які способи зміни кольору фону сторінки за допомогою JavaScript ви знаєте?
11. Що таке CSS-клас і чому його зручно використовувати для зміни теми сторінки?
12. Як реалізувати перемикання зображень у галереї без перезавантаження сторінки?
13. Яку роль відіграють масиви при створенні галереї зображень?
14. Що таке адаптивна верстка і для чого вона потрібна?
15. Що таке медіа-запити (@media) та як вони використовуються?
16. Які основні типи пристроїв потрібно враховувати при створенні адаптивного сайту?
17. Чому важливо забезпечити зручну взаємодію з елементами на сенсорних пристроях?
18. Які способи оптимізації зображень для вебсторінки ви знаєте?
19. Чому необхідно перевіряти роботу інтерактивних елементів при зміні розміру вікна браузера?
20. Які переваги має використання JavaScript для додавання інтерактивності до персонального вебсайту?

ЛАБОРАТОРНА РОБОТА № 7

ДОДАВАННЯ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ ТА АНІМАЦІЇ ДО ВЕБСТОРИНКИ

Мета роботи: Ознайомитися з можливостями використання мультимедійних елементів та анімаційних ефектів під час розробки сучасних вебсторінок.

Короткі теоретичні відомості

Мультимедійний контент є важливою складовою сучасних вебсайтів. До мультимедійних елементів належать зображення, аудіо, відео, анімація та інші засоби візуального й звукового представлення інформації. Використання мультимедіа дозволяє зробити вебсторінку більш інформативною, цікавою та зручною для сприйняття користувачами. Завдяки мультимедійному наповненню відвідувачі можуть не лише читати текстову інформацію, а й переглядати відеоматеріали, слухати аудіозаписи та взаємодіяти з анімованими елементами інтерфейсу.

Для розміщення графічних зображень використовується тег `<img>`. Він є одним із найпоширеніших елементів веброзробки та дозволяє відображати фотографії, ілюстрації, логотипи та інші графічні об'єкти. Основним атрибутом цього тегу є `src`, який визначає шлях до файлу зображення. Атрибут `alt` містить альтернативний текст, що відображається у випадку, якщо зображення не вдалося завантажити, а також використовується програмами екранного доступу для людей з порушеннями зору. Атрибути `width` та `height` дозволяють задавати розміри зображення.

Приклад використання:

```

```

Для групування зображень часто використовують тег `<figure>`, який дозволяє об'єднати графічний матеріал з його описом. Підпис до зображення створюється за допомогою тегу `<figcaption>`.

```
<figure>
```

```
  
```

```
  <figcaption>Пейзаж Карпат</figcaption>
```

```
</figure>
```

Для додавання аудіофайлів використовується тег `<audio>`. Цей елемент дозволяє відтворювати музику, подкасти, звукові ефекти та інші аудіозаписи без використання сторонніх плагінів. Атрибут `controls` додає елементи керування відтворенням.

```
<audio controls>
```

```
  <source src="music.mp3" type="audio/mpeg">
```

```
</audio>
```

Тег <audio> підтримує декілька важливих атрибутів:

- controls відображає панель керування;
- autoplay є автоматичним відтворенням після завантаження сторінки;
- loop – це повторне відтворення після завершення;
- muted означає вимкнення звуку;
- preload визначає спосіб попереднього завантаження файлу.

Для відтворення відеоматеріалів використовується тег <video>. Він підтримує сучасні формати відео та дозволяє відображати відео без додаткового програмного забезпечення.

Основними атрибутами тегу <video> є controls – елементи керування; width та height – розміри відео; autoplay – автоматичний запуск; loop – циклічне відтворення; muted – вимкнення звуку; poster – зображення-заставка перед запуском відео.

Приклад:

```
<video controls width="600" poster="preview.jpg">  
  <source src="movie.mp4" type="video/mp4">  
</video>
```

Сучасні вебсайти широко використовують галереї зображень. Галерея являє собою набір графічних елементів, які можуть відображатися у вигляді сітки або слайдера. Для створення галерей часто застосовують CSS Flexbox або CSS Grid.

Flexbox дозволяє легко розташовувати елементи в одному рядку або стовпці.

```
.gallery {  
  display: flex;  
  gap: 20px;  
}
```

CSS Grid надає більше можливостей для побудови складних макетів.

```
.gallery {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  gap: 20px;  
}
```

Оформлення мультимедійного контенту здійснюється за допомогою CSS (Cascading Style Sheets). CSS відповідає за зовнішній вигляд вебсторінки та дозволяє налаштовувати кольори, шрифти, розміри, відступи та розташування елементів.

Для створення плавних змін використовується властивість transition.

```
button {  
  transition: 0.3s;  
}
```

При наведенні курсора можна змінювати властивості елемента:

```
button:hover {  
  background-color: blue;
```

```
color: white;
}
```

Псевдоклас `:hover` є одним із найбільш використовуваних засобів створення інтерактивності. Він активується під час наведення курсора на елемент.

Для реалізації складніших візуальних ефектів використовуються CSS-анімації. Анімація створюється за допомогою правила `@keyframes`, яке визначає послідовність змін станів елемента.

```
@keyframes move {
  from {
    transform: translateX(0);
  }
  to {
    transform: translateX(200px);
  }
}
```

Після створення анімації її необхідно застосувати до елемента.

```
.box {
  animation: move 3s infinite;
}
```

Властивість `animation` дозволяє задавати тривалість анімації, кількість повторень, напрямок руху, затримку запуску, функцію швидкості виконання.

Для роботи з анімаціями часто використовуються властивості `transform`, які дозволяють змінювати положення, масштаб та обертання елементів. Найбільш поширені функції:

- `transform: scale(1.2);`
- `transform: rotate(45deg);`
- `transform: translateX(100px);`
- `transform: translateY(50px);`

Під час створення мультимедійних вебсторінок важливим аспектом є оптимізація файлів. Надмірно великі зображення, аудіо та відео можуть суттєво збільшити час завантаження сторінки. Для зменшення навантаження використовуються сучасні формати файлів, такі як JPG, PNG, WebP для зображень, MP3 для аудіо та MP4 для відео.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ РОБОТИ

1. Додати на вебсторінку (розроблену у лабораторній роботі № 6) зображення за допомогою тегу `<img>` з використанням атрибутів `src`, `alt`, `width` або CSS-стилів.

2. Реалізувати мультимедійну секцію з аудіофайлом за допомогою тегу `<audio>` та увімкнути елементи керування (`controls`).

3. Додати відео на вебсторінку за допомогою тегу `<video>` з можливістю керування відтворенням.

4. Створити галерею зображень (у вигляді сітки або ряду) з використанням CSS Flexbox або CSS Grid.

5. Оформити мультимедійні елементи за допомогою CSS (відступи, рамки, тіні, вирівнювання).

6. Реалізувати інтерактивні ефекти наведення курсора (`:hover`) для кнопок, зображень або блоків.

7. Використати властивість `transition` для створення плавних переходів між станами елементів.

8. Створити мінімум одну CSS-анімацію за допомогою `@keyframes` (наприклад: поява, рух або масштабування елемента).

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке мультимедійний контент у веброзробці?
2. Які основні типи мультимедійних елементів використовуються на вебсторінках?
3. Для чого призначений тег `<img>` і які його основні атрибути?
4. Яке призначення атрибута `alt` у зображеннях?
5. Для чого використовується тег `<audio>`?
6. Які атрибути дозволяють керувати відтворенням аудіо?
7. Для чого призначений тег `<video>`?
8. Які основні параметри можна задати для відео в HTML?
9. У чому різниця між тегам `<img>`, `<audio>` та `<video>`?
10. Що таке CSS-анімація і для чого вона використовується?
11. Яке призначення правила `@keyframes` у CSS?
12. Що робить властивість `transition`?
13. Які способи створення галереї зображень існують у веброзробці?
14. Які проблеми можуть виникнути при використанні великих відео- та аудіофайлів?
15. Які переваги використання мультимедіа на вебсторінках?

ЛАБОРАТОРНА РОБОТА № 8

ПУБЛІКАЦІЯ ВЕБСАЙТУ В МЕРЕЖІ «ІНТЕРНЕТ»

Мета роботи: Ознайомитися з основними принципами публікації вебсайтів у мережі «Інтернет». Навчитися підготовлювати вебпроект до розміщення на сервері, організовувати структуру файлів сайту, використовувати відносні шляхи до ресурсів, а також виконувати завантаження вебсайту на хостинг або платформу GitHub Pages. Закріпити навички перевірки працездатності вебресурсу після публікації та тестування його відображення в різних браузерах і на різних пристроях.

Короткі теоретичні відомості

Після завершення розробки вебсайту важливим етапом є його публікація в мережі «Інтернет». Публікація вебресурсу означає розміщення всіх файлів сайту на сервері, який забезпечує доступ до нього користувачам через браузер. Для цього використовуються спеціальні сервіси хостингу або платформи для розміщення статичних сайтів, наприклад GitHub Pages або інші вебхостинги.

Вебсайт складається з набору файлів: HTML-документів, CSS-стилів, JavaScript-скриптів, зображень, відео та інших ресурсів. Для коректної роботи сайту важливо правильно організувати структуру проєкту, оскільки неправильні шляхи до файлів можуть призвести до помилок відображення сторінки.

Одним із базових принципів публікації є використання відносних шляхів до файлів. Відносні шляхи дозволяють правильно підключати зображення, стилі та скрипти незалежно від місця розташування сайту на сервері. Це забезпечує коректну роботу вебресурсу після завантаження в мережу.

Перед публікацією вебсайту необхідно виконати його підготовку. Вона включає перевірку структури файлів, видалення зайвих або тестових елементів, а також перевірку коректності всіх посилань. Важливим етапом є оптимізація ресурсу, яка полягає у зменшенні розміру зображень, перевірці швидкості завантаження сторінки та усуненні помилок у коді.

Для публікації статичних вебсайтів часто використовується платформа GitHub Pages. Вона дозволяє безкоштовно розміщувати вебсторінки без необхідності налаштування власного сервера. Сайт завантажується у вигляді репозиторію, після чого стає доступним за унікальною URL-адресою.

Також для публікації можуть використовуватися FTP-сервери, які дозволяють передавати файли з локального комп'ютера на віддалений

сервер. У цьому випадку використовується спеціальне програмне забезпечення для завантаження файлів і налаштування доступу до сайту.

Важливим аспектом є коректність роботи вебсайту після публікації. Необхідно перевірити всі сторінки, посилання, мультимедійні елементи та інтерактивні функції. Особливу увагу слід приділити тому, щоб усі файли завантажувалися без помилок, а структура сайту відповідала початковому проєкту.

Після публікації вебсайту також рекомендується перевірити його відображення в різних браузерах та на різних пристроях. Це дозволяє переконатися у правильності адаптивної верстки та сумісності з різними платформами.

Окрему роль відіграє файл README, який часто додається до проєкту при розміщенні на GitHub. Він містить опис сайту, інструкцію щодо його запуску та основну інформацію про проєкт. Це робить проєкт більш зрозумілим для інших користувачів або розробників.

Таким чином, публікація вебсайту є завершальним етапом веброзробки, який дозволяє зробити створений ресурс доступним для широкого кола користувачів у мережі «Інтернет».

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ РОБОТИ

1. Підготувати готовий вебсайт до публікації (очистити від зайвих файлів і тестового коду).
2. Перевірити правильність структури проєкту (HTML, CSS, JS, зображення).
3. За необхідності, замінити абсолютні шляхи на відносні.
4. Перевірити коректність роботи всіх сторінок і посилань.
5. Оптимізувати зображення та мультимедійні файли для швидкого завантаження.
6. Завантажити вебсайт на платформу GitHub Pages або інший хостинг.
7. Отримати публічне посилання на вебсайт.
8. Перевірити роботу сайту після публікації в різних браузерах.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що означає публікація вебсайту в інтернеті?
2. Які основні способи розміщення вебсайтів існують?
3. Що таке хостинг і для чого він використовується?
4. Що таке GitHub Pages і як він працює?
5. У чому різниця між локальним і віддаленим розміщенням сайту?
6. Що таке відносні шляхи у веброзробці?
7. Чому важливо правильно організовувати структуру проєкту?
8. Які файли зазвичай входять до складу вебсайту?

9. Для чого потрібен файл README у вебпроекті?
10. Які основні етапи підготовки сайту до публікації?
11. Чому потрібно тестувати сайт після його розміщення в інтернеті?
12. Які проблеми можуть виникнути після публікації сайту?
13. Як перевірити коректність роботи сайту в різних браузерах?
14. Що таке оптимізація вебсайту перед публікацією?
15. Чому важливо перевіряти адаптивність після розміщення сайту?

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Основи програмування з HTML, CSS та JavaScript. URL: <https://prometheus.org.ua/prometheus-free/programming-fundamentals-html-css-javascript/> (дата звернення: 02.05.2026).
2. Курінний С. Розробка веб-сайтів для початківців: HTML, CSS, JavaScript. Київ, 2019. 156 с.
3. Кучай А. В., Білан С. О. Основи HTML. Київ, 2018. № 12. С. 20-25.
4. What is HTML? URL: <https://www.hostinger.com/tutorials/what-is-html/> (дата звернення: 12.04.2026).
5. Основи CSS URL: <https://www.w3.org/Style/Examples/007/units.en.html> (дата звернення: 18.04.2026).
6. CSS для початківців URL: https://www.tutorialspoint.com/css/css_measurement_units.htm (дата звернення: 25.04.2026).
7. What is JavaScript? URL: <https://www.tutorialsteacher.com/javascript/what-is-javascript> (дата звернення: 02.05.2026).
8. Haverbeke M. Eloquent JavaScript: A Modern Introduction to Programming. 3rd ed. No Starch Press, 2018. 472 p.
9. Flanagan D. JavaScript: The Definitive Guide. 7th ed. O'Reilly Media, 2020. 706 p.
10. JS Theories. URL: https://is.theorizeit.org/wiki/Main_Page (дата звернення: 12.04.2026).
11. Simpson K. You Don't Know JS. O'Reilly Media, 2015. 278 p.
12. Zeldman J. Designing with Web Standards. 3rd ed. New Riders, 2009. 376 p.
13. Розробка веб-застосунків URL: <https://webstudio2u.net/ua/site-develop/641-razrabotka-veb-prilozheniy.html> (дата звернення: 29.04.2026).
14. Web application URL: <https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app> (дата звернення: 24.04.2026).
15. Websites vs web applications URL: <https://www.wix.com/velo/blog/post/websites-vs-web-applications> (дата звернення: 12.04.2026).
16. Responsive Web Design Basics. URL: <https://web.dev/responsive-web-design-basics/> (дата звернення: 12.04.2026).
17. Krug S. Don't Make Me Think. New Riders, 2014. 216 p.
18. Norman D. The Design of Everyday Things. MIT Press, 2013. 368 p.
19. Shneiderman B. Designing the User Interface: Strategies for Effective Human-Computer Interaction. 6th ed. Pearson, 2016. 672 p.
20. Raskin J. The Humane Interface. Addison-Wesley, 2000. 256 p.
21. Keith J. HTML5 for Web Designers. A Book Apart, 2010. 87 p.
22. Verou L. CSS Secrets. O'Reilly Media, 2015. 392 p.

23. Niederst Robbins J. Learning Web Design. 5th ed. O'Reilly Media, 2018. 808 p.
24. Modern Web Development. URL: <https://developer.mozilla.org/en-US/docs/Learn> (дата звернення: 04.05.2026).
25. Bootstrap Documentation. URL: <https://getbootstrap.com/docs/> (дата звернення: 04.05.2026).
26. HTML Multimedia – Working with Audio and Video URL: https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding (дата звернення: 30.04.2026).
27. HTML <video> element. URL: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/video> (дата звернення: 30.04.2026).
28. HTML <audio> element. URL: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio> (дата звернення: 30.04.2026).
29. CSS Animations Guide. URL: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animations (дата звернення: 30.04.2026).
30. CSS Transitions. URL: <https://developer.mozilla.org/en-US/docs/Web/CSS/transition> (дата звернення: 30.04.2026).
31. CSS @keyframes Rule. URL: <https://developer.mozilla.org/en-US/docs/Web/CSS/@keyframes> (дата звернення: 30.04.2026).
32. Web Accessibility Basics (alt text, media). URL: <https://www.w3.org/WAI/fundamentals/accessibility-intro/> (дата звернення: 30.04.2026).
33. GitHub Pages Getting Started. URL: <https://docs.github.com/en/pages/getting-started-with-github-pages> (дата звернення: 30.04.2026).
34. Image Optimization Guide for Web. URL: <https://web.dev/fast/#optimize-your-images> (дата звернення: 30.04.2026).

Електронне навчальне видання

**Олена Валеріївна Войцеховська
Вікторія Олегівна Кривенька**

**Методичні вказівки до виконання лабораторних робіт з
дисципліни «Web-технології та Web-дизайн»
зі спеціальності «Комп'ютерна інженерія»
(освітня програма «Системне програмування»)**

Рукопис оформила О. Войцеховська

Редактор Н. Кравчук

Оригінал-макет виготовлено в РВВ ВНТУ

Підписано до видання 23.09.2026 р.
Гарнітура Times New Roman.
Зам. № P2026-123

Видавець та виготовлювач
Вінницький національний технічний університет,
Редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95,
м. Вінниця, 21021.
Email: rvv.vntu@gmail.com
Свідоцтво суб'єкта видавничої справи
серія ДК No 3516 від 01.07.2009 р.