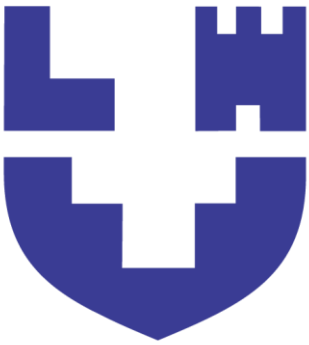
Міністерство освіти та науки України
Луцький національний технічний університет



Web-програмування

Конспект лекцій
для здобувачів першого (бакалаврського) рівня вищої освіти
освітньої програми «Цифровий маркетинг»
галузі знань D Бізнес, адміністрування та право
спеціальності D5 Маркетинг денної та заочної форм навчання

Луцьк 2025

УДК 004.42 (07)
В-26

Електронна копія друкованого видання передана для внесення в репозитарій ЛНТУ
Директор бібліотеки _____ Н. П. Поліщук

Рекомендовано до видання вченою радою факультету бізнесу та права ЛНТУ, протокол №___ від _____ 2025 року.

Голова вченої ради факультету бізнесу та права _____ Л. Л. Ковальська

Розглянуто і схвалено на засіданні кафедри маркетингу ЛНТУ, протокол №___ від ____ _____ 2025 року.
Завідувачка кафедри маркетингу _____ І. Ф. Лорві

Укладач : _____ О. М. Клімович, кандидат економічних наук, доцент кафедри маркетингу ЛНТУ;

Рецензенти : _____ І.Ф. Лорві, кандидат економічних наук, доцент кафедри маркетингу ЛНТУ;

Відповідальний
за випуск: _____ І. Ф. Лорві, кандидат економічних наук, доцент кафедри маркетингу ЛНТУ.

В-26	Web-програмування. Конспект лекцій для здобувачів першого (бакалаврського) рівня вищої освіти галузі знань D Бізнес, адміністрування та право спеціальності D5 Маркетинг ОП Цифровий маркетинг денної та заочної форм навчання / уклад. О.М. Клімович Луцьк: Луцький НТУ, 2025. 56 с.

Методичне видання складене відповідно до діючої програми курсу «Web-програмування» та містить перелік тем лекційних занять та перелік питань самопідготовки до іспиту та список рекомендованої літератури.
Призначене для здобувачів першого (бакалаврського) рівня вищої освіти спеціальності D5 Маркетинг освітньо-професійної програми «Цифровий маркетинг» денної та заочної форм навчання.

ЗМІСТ

ВСТУП

У сучасному цифровому світі веб-програмування є не лише технічним процесом створення веб-ресурсів, а й важливим інструментом реалізації маркетингових стратегій. Веб-технології відіграють ключову роль у залученні клієнтів, формуванні бренду та просуванні товарів і послуг. Від здатності ефективно використовувати цифрові інструменти залежить успішність бізнесу в умовах стрімкої цифрової трансформації.

Дисципліна **«Web-програмування»** є важливою складовою підготовки маркетологів, оскільки забезпечує їх базовими знаннями у сфері створення та управління веб-ресурсами. Вона поєднує в собі теоретичні основи веб-розробки та практичні навички роботи з сучасними технологіями, такими як HTML, CSS, JavaScript, а також знайомить студентів із можливостями конструкторів сайтів та систем управління контентом (CMS).

Метою вивчення дисципліни є надання студентам знань і навичок, необхідних для ефективного використання веб-технологій у маркетинговій діяльності. Особлива увага приділяється розумінню основ веб-дизайну, пошукової оптимізації (SEO), інструментів аналітики та взаємодії з користувачами.

У процесі вивчення дисципліни здобувачі освіти:

- Ознайомляться з основами веб-програмування та ключовими технологіями.
- Навчаться працювати з HTML та CSS для створення базової структури та стилізації веб-сторінок.
- Оволодіють JavaScript для додавання інтерактивності та роботи з динамічними елементами.
- Дослідять можливості CMS та конструкторів сайтів для швидкого створення веб-ресурсів без поглиблених знань у програмуванні.
- Вивчать SEO-принципи для оптимізації веб-сайтів та підвищення їх видимості у пошукових системах.
- Ознайомляться з інструментами веб-аналітики, що дозволяють відстежувати поведінку користувачів та підвищувати ефективність маркетингових кампаній.

Дисципліна «Web-програмування» сприяє формуванню:

Інтегральної компетентності – здатності вирішувати складні спеціалізовані завдання у сфері маркетингу за допомогою веб-технологій.

Загальних компетентностей, зокрема навичок використання інформаційних і комунікаційних технологій.

Спеціальних компетентностей, що включають роботу з маркетинговими інформаційними системами, цифровими маркетинговими інструментами та впровадження результатів маркетингових досліджень у практику.

У результаті вивчення дисципліни студенти зможуть:

- Використовувати цифрові технології для створення, управління та оптимізації веб-ресурсів.

- Аналізувати ефективність веб-сайтів та маркетингових інструментів у цифровому середовищі.
- Використовувати веб-технології для покращення взаємодії з аудиторією та підвищення ефективності маркетингових кампаній.

Вивчення дисципліни **«Web-програмування»** надає студентам необхідні навички для ефективної роботи в сучасному цифровому середовищі, сприяє їх професійному розвитку та адаптації до вимог ринку.

Змістовий модуль 1: Основи веб-програмування

Тема 1. Вступ до веб-програмування

1.1. Огляд веб-технологій. Клієнтська і серверна частини.
1.2. Сфери застосування веб-програмування в маркетингу
1.3. Огляд популярних інструментів для веб-розробки.

1.1. Огляд веб-технологій

Веб-технології охоплюють широкий спектр засобів для створення, управління та обслуговування веб-ресурсів. Основними поняттями, які необхідно знати, є:

Термін	Опис
Інтернет	Глобальна мережа, що об'єднує мільйони пристроїв по всьому світу.
Веб-сайт	Сукупність веб-сторінок, об'єднаних єдиною адресою (доменом).
Веб-сторінка	Документ, створений за допомогою HTML, CSS та JavaScript, який відображається у браузері.
Доменне ім'я	Унікальне ім'я веб-сайту (наприклад, google.com).
IP-адреса	Унікальний числовий ідентифікатор пристрою в мережі Інтернет.
Хостинг	Послуга зберігання веб-сайтів на сервері.
Протокол HTTP/ HTTPS	Протокол передачі гіпертексту, що забезпечує зв'язок між сервером та браузером.

Як працює Інтернет

Інтернет – це глобальна система взаємозв'язаних комп'ютерних мереж. Основні етапи роботи:

1. Користувач вводить URL-адресу в браузері.
2. Браузер надсилає запит через DNS-сервер для отримання IP-адреси веб-ресурсу.
3. Запит передається на веб-сервер, де знаходиться сайт.
4. Сервер обробляє запит і відправляє HTML-код назад у браузер.
5. Браузер відображає веб-сторінку, обробляючи HTML, CSS та JavaScript.

Схема роботи веб-сайту:

📌 Користувач → Браузер → DNS-сервер → Веб-сервер → Браузер (відображення сторінки)

Що таке веб-програмування?

Веб-програмування – це процес створення веб-сайтів та веб-додатків, які працюють в інтернеті. Веб-програмісти використовують різні мови програмування та технології для створення функціональних та інтерактивних веб-сторінок.

Історія веб-програмування

1991 – створення першої веб-сторінки Тімом Бернерс-Лі.

1993 – поява першого графічного веб-браузера Mosaic.

1995 – створення мови JavaScript.

1996 – випуск CSS (Cascading Style Sheets).

2005 – поява технологій AJAX для асинхронного обміну даними.

Значення веб-програмування в маркетингу:

Онлайн-присутність: Без веб-сайту бізнес втрачає можливості для залучення онлайн-аудиторії.

Маркетингові кампанії: Лендінг-пейджі для реклами, форми для збору контактів, інтеграція з CRM-системами — усе це потребує базових знань у веб-програмуванні.

Аналітика: Розміщення коду для відстеження поведінки користувачів (наприклад, Google Analytics) потребує мінімальних навичок роботи з HTML.

Завдяки веб-програмуванню компанії можуть:

- Створювати корпоративні сайти, інтернет-магазини, лендінги.
- Оптимізувати сайти під пошукові системи (SEO).
- Використовувати веб-аналітику для оцінки ефективності маркетингових заходів.
- Впроваджувати інтерактивні рішення, такі як форми зворотного зв'язку, чат-боти, особисті кабінети користувачів.

Основні компоненти веб-програмування:

Компонент	Опис	Приклад технологій
Frontend (Клієнтська частина)	Відповідає за відображення контенту та взаємодію користувача з сайтом.	**HTML, CSS, JavaScript**
Backend (Серверна частина)	Обробка запитів, збереження та обробка даних.	**PHP, Python, Node.js, MySQL, MongoDB**

1. ***Фронтенд*** (Frontend): Клієнтська частина (Frontend): це те, що користувач бачить і з чим взаємодіє. Включає HTML, CSS та JavaScript.

- HTML (HyperText Markup Language): Мова розмітки, що використовується для створення структури веб-сторінок.

- CSS (Cascading Style Sheets): Мова стилізації, що використовується для оформлення вигляду веб-сторінок (шрифти, кольори, розміщення елементів).

- JavaScript: Мова програмування, яка додає динамічні функції та інтерактивність до веб-сторінок.

Фреймворки та бібліотеки:

React: Бібліотека JavaScript для створення користувацьких інтерфейсів.

Angular: Фреймворк для створення динамічних веб-додатків.

Vue.js: Прогресивний фреймворк для створення користувацьких інтерфейсів.

2. ***Бекенд*** (Backend): Серверна частина (Backend): відповідає за логіку, бази даних і роботу з сервером. Використовує мови програмування на зразок PHP, Python або Node.js.

3. **Бази даних**:

Реляційні бази даних (SQL):

MySQL: Популярна реляційна база даних.

PostgreSQL: Потужна реляційна база даних з підтримкою розширених функцій.

SQLite: Легка вбудована база даних.

Нереляційні бази даних (NoSQL):

MongoDB: Документо-орієнтована база даних.

Cassandra: Високопродуктивна база даних для обробки великих обсягів даних.

Redis: Швидка база даних, що зберігає дані в пам'яті.

4. **Інструменти та технології**:

Git: Система контролю версій, яка дозволяє відстежувати зміни в коді.

Docker: Платформа для контейнеризації додатків.

Webpack: Інструмент для зборки модулів JavaScript.

REST та GraphQL: Технології для створення API (інтерфейсів програмування додатків).

Взаємодія Frontend і Backend:

Користувач вводить дані на сайті (наприклад, реєстраційна форма) →

Дані відправляються на сервер (Backend) →

Сервер обробляє дані та відправляє відповідь назад →

Користувач бачить результат на екрані (Frontend).

1.2. Сфери застосування веб-програмування в маркетингу

Веб-програмування відіграє ключову роль у сучасному маркетингу, оскільки більшість бізнесів орієнтовані на цифровий простір. Веб-технології дозволяють створювати ефективні маркетингові інструменти для просування товарів та послуг, взаємодії з клієнтами та аналітики.

Роль веб-програмування для маркетологів

Базове розуміння веб-технологій дає маркетлогу такі переваги:

Оптимізація сайтів для SEO: Навички роботи з HTML дозволяють коригувати мета-теги, заголовки, структуру сторінок для покращення позицій у пошукових системах.

Налаштування аналітики: Знання, як вставити код відстеження, налаштувати цілі в Google Analytics.

Створення лендінгів: Використання конструкторів сайтів або базових CMS (WordPress) для швидкого запуску маркетингових кампаній.

А/Б тестування: Уміння змінювати контент або елементи дизайну для перевірки ефективності різних варіантів реклами.

Корпоративні та комерційні веб-сайти

Один із найважливіших елементів маркетингу — це офіційний сайт компанії.

Як веб-програмування допомагає?

- Створення адаптивного дизайну, який працює на різних пристроях. Розробка динамічного контенту, який легко оновлювати. Оптимізація швидкості завантаження сторінок для покращення користувацького досвіду та SEO.

Приклад: Сайт компанії Nike (https://www.nike.com) пропонує персоналізований контент, рекомендації на основі історії покупок, що значно підвищує ефективність продажів.

- Лендінг-сторінки (Landing Pages). Лендінги використовуються для реклами конкретного продукту або послуги та є важливим маркетинговим інструментом.

Розробка динамічних форм реєстрації та збору лідів (контактів потенційних клієнтів). Оптимізація швидкості завантаження сторінки для покращення конверсії. A/B тестування різних варіантів сторінки для визначення найефективнішого дизайну.

Приклад: Лендінг від Apple для нового iPhone містить візуально привабливий дизайн, анімації та чіткий заклик до дії. За допомогою веб-програмування можна налаштувати автоматичну зміну контенту залежно від геолокації користувача.

- Інтернет-магазини (E-commerce). Онлайн-магазини використовують технології веб-програмування для обробки замовлень, інтеграції з платіжними системами та ведення аналітики.

- Створення кошика покупок та оформлення замовлень. Інтеграція з платіжними системами (LiqPay, PayPal, Stripe). Автоматизація маркетингових процесів, таких як рекомендації товарів, email-розсилки та push-повідомлення. Підключення CRM-систем для керування клієнтською базою.

Приклад:

Rozetka (https://www.rozetka.com.ua/) використовує динамічні фільтри товарів, систему відгуків, персоналізовані пропозиції.

- Контент-маркетинг та блогові платформи. Компанії ведуть блоги для залучення трафіку на свої сайти та підвищення впізнаваності бренду.

Автоматичне оновлення контенту через CMS (WordPress, Joomla). SEO-оптимізація блогу (налаштування мета-тегів, структура URL). Впровадження аналітики для відстеження популярності статей та поведінки користувачів.

Приклад:

HubSpot Blog (https://blog.hubspot.com) використовує веб-програмування для автоматичного оновлення статей, інтеграції з CRM і аналізу трафіку.

- Веб-аналітика та автоматизація маркетингу. Веб-програмування допомагає маркетологам автоматизувати процеси збору та обробки даних.

Інтеграція Google Analytics та Google Tag Manager для збору інформації про відвідувачів. Автоматичне відстеження кліків, конверсій, поведінки користувачів. Розробка скриптів для аналізу ефективності рекламних кампаній.

Приклад:

Facebook Ads використовує веб-аналітику для точного таргетингу реклами.

- SEO (Search Engine Optimization) та технічна оптимізація. SEO дозволяє веб-сайтам займати вищі позиції у пошукових системах (Google, Bing).

Розмітка сторінок за допомогою HTML5 та Schema.org. Оптимізація швидкості завантаження через стиснення зображень, кешування, асинхронне завантаження скриптів. Адаптивний дизайн (Responsive Web Design) для коректного відображення на мобільних пристроях.

Приклад:

Amazon активно використовує SEO-оптимізацію для залучення органічного трафіку з Google.

- Соціальні мережі та інтеграція веб-сайтів. Веб-програмування дозволяє інтегрувати сайти з соцмережами для підвищення залученості користувачів.

🔹 Як веб-програмування допомагає?

Додавання кнопок "Поділитися" у соцмережах (Facebook, Instagram, Twitter). Інтеграція Facebook Pixel для реклами та ретаргетингу. Автоматична публікація контенту в соцмережах за допомогою API.

Приклад:

Веб-сайт магазину може автоматично показувати Instagram-стрічку товарів.

- Реклама та ремаркетинг. Веб-програмування дозволяє налаштовувати рекламні кампанії та ремаркетинг.

Розміщення рекламних банерів на сайтах через Google AdSense. Використання JavaScript для динамічного показу реклами. Таргетинг за допомогою cookie-файлів та пікселів.

Приклад:

Google Ads використовує веб-скрипти для відстеження поведінки користувачів та персоналізації реклами.

Веб-програмування є невід'ємною частиною сучасного маркетингу. Воно дозволяє:

- Створювати ефективні корпоративні сайти та лендінги.
- Оптимізувати сайти для пошукових систем (SEO).
- Інтегрувати веб-аналітику для покращення маркетингових кампаній.
- Автоматизувати рекламу та ремаркетинг.

Маркетологи, які розуміють основи веб-програмування, мають значну перевагу, оскільки можуть швидше адаптувати стратегії до цифрового ринку.

1.3. Огляд популярних інструментів для веб-розробки.

Інструменти для написання коду

1. Текстові редактори

Текстові редактори – це прості інструменти, які дозволяють писати і редагувати код. Вони підходять для початківців і часто використовуються професійними розробниками.

Популярні текстові редактори:

Notepad++ (Windows): Простий та легкий текстовий редактор з підсвіткою синтаксису.

Sublime Text (Windows, Mac, Linux): Потужний текстовий редактор з багатьма можливостями.

Visual Studio Code (Windows, Mac, Linux): Безкоштовний редактор від Microsoft з підтримкою багатьох мов програмування та розширень.

Atom (Windows, Mac, Linux): Текстовий редактор від GitHub з багатими можливостями налаштування.

Як працювати з текстовим редактором:

1. Встановіть текстовий редактор: Завантажте та встановіть один з вищезгаданих текстових редакторів з офіційного сайту.
2. Створіть новий файл: Відкрийте редактор та створіть новий файл з розширенням .html, .css, або .js.
3. Напишіть код: Введіть код в редакторі та збережіть файл.
4. Відкрийте файл у браузері: Для HTML файлів просто відкрийте файл у браузері, щоб переглянути результат.

2. Інтегровані середовища розробки (IDE)

IDE (Інтегровані середовища розробки) – це потужні інструменти, які надають функціональні можливості для написання, тестування та налагодження коду.
Популярні IDE:

- Visual Studio Code: Підтримує розширення для багатьох мов програмування та має вбудовані інструменти для налагодження.
- WebStorm: Потужна IDE від JetBrains спеціально для розробки веб-додатків.
- Brackets: Відкритий редактор коду з вбудованими функціями для веб-розробки.

Як працювати з IDE:

1. Встановіть IDE: Завантажте та встановіть IDE з офіційного сайту.
2. Створіть новий проект: Відкрийте IDE та створіть новий проект для веб-розробки.
3. Напишіть код: Використовуйте редактор коду для написання HTML, CSS та JavaScript.
4. Запустіть проект: Багато IDE мають вбудовані сервери для запуску проекту, що дозволяє легко переглядати зміни у браузері.

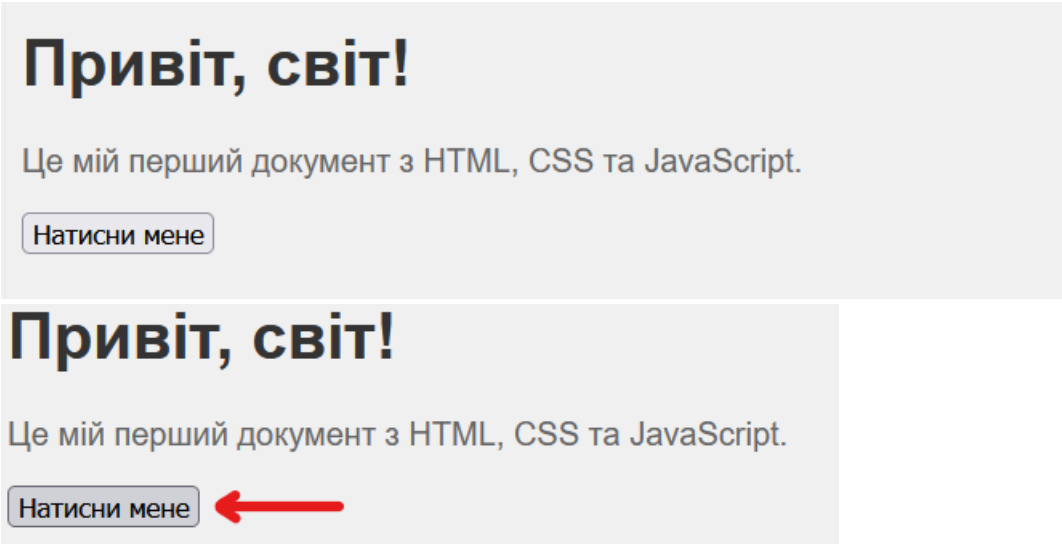
Життєвий цикл розробки веб-сайту

1. **Планування:** Визначення цілей, цільової аудиторії, аналіз конкурентів.
2. **Дизайн:** Створення макетів, прототипів (наприклад, у Figma).
3. **Розробка:** Написання коду для Frontend і Backend.
4. **Тестування:** Перевірка сайту на наявність помилок, забезпечення його безпеки та швидкодії.
5. **Запуск:** Розгортання сайту на сервері.
6. **Підтримка та оновлення:** Регулярне оновлення контенту, оптимізація роботи сайту.

Приклад використання HTML, CSS та JavaScript разом

```html
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Приклад сторінки</title>
    <style>
        body {
            background-color: #f0f0f0;
            font-family: Arial, sans-serif;
        }
        h1 {
            color: #333;
        }
        p {
            color: #666;
        }
    </style>
    <script>
        function showMessage() {
            alert('Привіт, світ!');
        }
    </script>
</head>
<body>
    <h1>Привіт, світ!</h1>
    <p>Це мій перший документ з HTML, CSS та JavaScript.</p>
    <button onclick="showMessage()">Натисни мене</button>
</body>
</html>
```

Результат



Присутня анімація при наведенні курсору.

Тема 2. Основи HTML та CSS

2.1.Що таке HTML. Структура веб-сторінки, теги, атрибути
2.2. Основи CSS. Стилізація веб-сторінок

2.1.Що таке HTML?

HTML (HyperText Markup Language) – це стандартна мова розмітки для створення веб-сторінок. Вона визначає структуру вмісту за допомогою елементів, таких як заголовки, параграфи, зображення та посилання.

Структура HTML-документу

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Веб-програмування</title>
</head>
<body>
    <h1>Вітаємо у світі веб-програмування!</h1>
    <p>Це ваша перша веб-сторінка.</p>
</body>
</html>
```

Розбір структури:

- **<!DOCTYPE html>** – визначає тип документа і версію HTML.
- **<html>** – кореневий елемент HTML-документа.
- **<head>** – містить метадані документа (назва, кодування, стилі).
- **<body>** – містить видимий вміст веб-сторінки.

Основні теги HTML

Теги в HTML — це команди, що визначають структуру сторінки. Вони завжди беруться у кутові дужки < >. Наприклад: <h1>Це заголовок</h1> Це абзац тексту.

Види HTML-тегів (Табл. 1.):

Парні: мають відкриваючий і закриваючий тег. Це абзац тексту.

Одинарні: не потребують закриваючого тегу. <img src="image.jpg" alt="Зображення">

Атрибути HTML. ***Атрибути*** – це додаткові параметри тегів, які змінюють їхню поведінку або вигляд (Табл. 2).

Табл. 1. Приклади тегів

Категорія	Тег	Опис	Приклад
Заголовки	`<h1>` - `<h6>`	Створення заголовків від найбільшого до найменшого.	`<h1>Основний заголовок</h1>`
Абзаци	`<p>`	Визначення текстового блоку (абзацу).	`<p>Це текст абзацу.</p>`
Посилання	`<a>`	Створення гіперпосилань.	`<a href="https://example.com">Перейти</a>`
Зображення	`<img>`	Вставка зображення.	`<img src="photo.jpg" alt="Фото">`
Списки	`<ul>`, `<ol>`, `<li>`	Створення списків (невпорядкованих та впорядкованих).	`<ul><li>Елемент списку</li></ul>`
Таблиці	`<table>`, `<tr>`, `<td>`	Побудова таблиць.	`<table><tr><td>Дані</td></tr></table>`
Форми	`<form>`, `<input>`	Створення форм для введення даних.	`<form><input type="text"></form>`
Контейнери	`<div>`, `<span>`	Блокове та рядкове групування елементів.	`<div>Це блок</div><span>Це рядок</span>`

Табл. 2. Основні атрибути

Атрибут	Опис	Приклад
`href`	Задає посилання	`<a href="https://example.com">Лінк</a>`
`src`	Визначає джерело зображення	`<img src="photo.jpg" alt="Фото">`
`alt`	Опис зображення	`<img src="photo.jpg" alt="Опис">`
`title`	Підказка при наведенні	`<p title="Підказка">Текст</p>`
`id`	Унікальний ідентифікатор	`<div id="header">Заголовок</div>`
`class`	Клас для стилізації елементів	`<p class="highlight">Текст</p>`

Приклад простої веб-сторінки

```html
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Моя перша веб-сторінка</title>
</head>
<body>
    <h1>Привіт, світ!</h1>
    <p>Це мій перший веб-сайт.</p>
    <a href="https://example.com">Посилання на інший сайт</a>
    <img src="https://example.com/image.jpg" alt="Приклад зображення">
</body>
</html>
```

2.2. Основи CSS. Стилізація веб-сторінок

Що таке CSS?

CSS (Cascading Style Sheets) – це мова стилів, яка використовується для опису вигляду HTML-документу. Вона дозволяє змінювати кольори, шрифти, розміри та розташування елементів на сторінці.

Основні можливості CSS:

✅ Зміна кольору тексту та фону.

✅ Налаштування розміру шрифтів.

✅ Вирівнювання контенту.

✅ Адаптивний дизайн.

Синтаксис CSS

```css
body {
    background-color: #f0f0f0;
    font-family: Arial, sans-serif;
}

h1 {
    color: #333;
}

p {
    color: #666;
}
```

Способи підключення CSS: Підключення CSS до HTML

1. **Inline (вбудовані стилі)** – стилі задаються безпосередньо в HTML-елементі.

```html
<p style="color: red;">Це червоний текст</p>
```

2. **Internal (внутрішні стилі)** – стилі додаються в `<head>` у блоці `<style>`.

```html
<head>
    <style>
        h1 {
            color: blue;
            text-align: center;
        }
    </style>
</head>
```

або

```html
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <style>
        body {
            background-color: #f0f0f0;
        }
        h1 {
            color: #333;
        }
    </style>
    <title>Вбудовані стилі</title>
</head>
<body>
    <h1>Привіт, світ!</h1>
</body>
</html>
```

3. **External (зовнішні стилі)** – окремий CSS-файл (`styles.css`).

```html
<link rel="stylesheet" href="styles.css">
```

Файл `styles.css`:

```css
body {
    background-color: #f0f0f0;
    font-family: Arial, sans-serif;
}
h1 {
    color: darkgreen;
}
```

або

```css
body {
    background-color: #f0f0f0;
}
h1 {
    color: #333;
}
```

```html
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="styles.css">
    <title>Зовнішні стилі</title>
</head>
<body>
    <h1>Привіт, світ!</h1>
</body>
</html>
```

Основні властивості CSS:

Властивість	Опис	Приклад
`color`	Колір тексту	`color: red;`
`background-color`	Колір фону	`background-color: lightblue;`
`font-size`	Розмір шрифту	`font-size: 20px;`
`text-align`	Вирівнювання тексту	`text-align: center;`
`margin`	Зовнішній відступ	`margin: 10px;`
`padding`	Внутрішній відступ	`padding: 10px;`
`border`	Рамка навколо елементу	`border: 1px solid black;`
`width`, `height`	Ширина та висота	`width: 100px; height: 200px;`

Отже**, HTML** відповідає за структуру веб-сторінки, а **CSS** — за її вигляд. Використання різних типів стилізації дозволяє зробити сайт привабливим і зручним для користувачів.

Підключіть CSS до HTML:

Додайте в <head> тег для підключення стилів.

```html
<head>
    <meta charset="UTF-8">
    <title>Моя перша сторінка</title>
    <link rel="stylesheet" href="styles.css">
</head>
```

Збережіть обидва файли.

Відкрийте `index.html` подвійним кліком або через браузер

Тема 3. Основні компоненти веб-програмування

3.1. Введення в JavaScript

3.2. Взаємодія з DOM (Document Object Model)

3.1. Введення в JavaScript

Що таке JavaScript?

JavaScript – це мова програмування, яка використовується для створення інтерактивних та динамічних елементів на веб-сторінках. З її допомогою можна змінювати HTML та CSS на льоту, обробляти події користувача та працювати з даними.

JavaScript є однією з основних технологій веб-розробки, яка дозволяє додавати інтерактивність та динамізм на веб-сторінки. За допомогою цього мови програмування можна змінювати контент сторінки без необхідності її перезавантаження, а також взаємодіяти з користувачем в режимі реального часу. Веб-сторінки стають динамічними завдяки JavaScript, дозволяючи змінювати елементи сторінки, працювати з формами, а також створювати інтерактивні анімації, меню та багато іншого.

Основи JavaScript

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Основи JavaScript</title>
    <script>
        function showMessage() {
            alert('Привіт, світ!');
        }
    </script>
</head>
<body>
    <button onclick="showMessage()">Натисни мене</button>
</body>
</html>
```

Використання JavaScript для зміни контенту

```html
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Динамічний контент</title>
    <script>
        function changeContent() {
            document.getElementById('myParagraph').innerHTML = 'Зміст змінено!';
        }
    </script>
</head>
<body>
    <p id="myParagraph">Це оригінальний текст.</p>
    <button onclick="changeContent()">Змінити текст</button>
</body>
</html>
```

Основні можливості JavaScript:

✅ Динамічна зміна контенту (текст, зображення, стилі).
✅ Обробка подій (натискання кнопок, введення даних).
✅ Валідація форм перед відправкою.
✅ Робота з даними без перезавантаження сторінки (AJAX).
✅ Анімації та інтерактивні елементи.

Підключення JavaScript до HTML:

JavaScript можна додавати у документ кількома способами:

Inline (вбудований):

```html
<button onclick="alert('Привіт!')">Натисни мене</button>
```

Internal (у блоці <script>):

```html
<script>
    alert("Привіт зі скрипту!");
</script>
```

External (зовнішній файл):

Створіть файл script.js і підключіть його:

```html
<script src="script.js"></script>
```

Змінні у JavaScript:

Змінні використовуються для зберігання даних, які можуть змінюватися під час виконання програми. Змінні – це контейнери для зберігання даних.

В JavaScript є кілька способів оголошення змінних:

- var — старий спосіб оголошення змінних, не рекомендується використовувати в сучасних проектах.
- let — сучасний спосіб оголошення змінних з блочним рівнем видимості.
- const — використовується для оголошення змінних, значення яких не повинно змінюватися.

```javascript
let age = 25;  // змінна з можливістю зміни значення
const name = "John";  // змінна, значення якої не можна змінити
```

```javascript
// Оголошення змінних
let ім'я = "Марія";  // змінна, яку можна змінювати
const рік = 2024;    // константа, значення якої не можна змінити
var місто = "Київ"; // застарілий спосіб оголошення змінних

// Виведення в консоль
console.log("Мене звати " + ім'я);
```

🔷 Основні типи даних:

- String (рядок): "Привіт"
- Number (число): 2024
- Boolean (логічний): true, false
- Array (масив): ['яблуко', 'банан', 'груша']
- Object (об’єкт**):** {ім'я: 'Марія', вік: 25}

Оператори – дозволяють виконувати дії над змінними.

```javascript
let a = 10;
let b = 5;

// Арифметичні оператори
console.log(a + b);  // Додавання: 15
console.log(a - b);  // Віднімання: 5
console.log(a * b);  // Множення: 50
console.log(a / b);  // Ділення: 2

// Оператори порівняння
console.log(a > b);   // true
console.log(a === b); // false (строге порівняння)
```

Функції у JavaScript:

Функції – це блоки коду, які виконують певне завдання. Функції дозволяють організувати код у блоки, які можна викликати в різних місцях програми, забезпечуючи повторне використання коду.

```javascript
// Оголошення функції
function привітання(ім'я) {
    console.log("Привіт, " + ім'я + "!");
}

// Виклик функції
привітання("Марія");  // Виведе: Привіт, Марія!
```

Стрілочна функція (сучасний синтаксис):

```javascript
const квадрат = (число) => число * число;
console.log(квадрат(5));  // Виведе: 25
```

Цикли у JavaScript

Цикли дозволяють повторно виконувати певний блок коду. Найбільш поширені цикли — це for, while, і do...while.

Цикли дозволяють повторювати одну і ту ж дію кілька разів.

1. **for** – цикл із лічильником:
   for (let i = 1; i <= 5; i++) { console.log("Крок " + i); }
2. **while** – цикл, що виконується, поки умова є істинною:
   let число = 0; while (число < 3) { console.log(число); число++; }

3.2. Взаємодія з DOM (Document Object Model)

Один із основних аспектів, за допомогою якого JavaScript робить веб-сторінки динамічними, — це взаємодія з елементами DOM (Document Object Model). DOM дозволяє програмісту змінювати HTML та CSS на сторінці, а також реагувати на події користувача. Кожен елемент HTML (наприклад, `<div>`, `<p>`, `<button>`) є об'єктом у цій моделі, і JavaScript може взаємодіяти з цими об'єктами, щоб змінювати їх або виконувати інші дії. **DOM** – це модель веб-сторінки, яка дозволяє JavaScript взаємодіяти з HTML.

Основні методи роботи з DOM:

1. Отримання елементів:
   const заголовок = document.getElementById("заголовок"); // По ID
   const параграф = document.querySelector("p"); // Перший елемент по селектору
2. Зміна контенту:
   заголовок.innerHTML = "Новий заголовок!";

параграф.textContent = "Оновлений текст.";

3. Зміна стилів:
   параграф.style.color = "blue";
   параграф.style.fontSize = "20px";
4. Обробка подій:
   document.getElementById("кнопка").addEventListener("click", function() {
   alert("Кнопку натиснуто!"); });

Створення інтерактивних елементів. Інтерактивність на веб-сторінці можна досягти, використовуючи JavaScript для зміни контенту або стилів в реальному часі.

1. **Інтерактивне меню:** JavaScript може використовуватись для створення меню, яке змінюється в залежності від дій користувача. Наприклад, меню може з'являтися після натискання кнопки:

```javascript
let menu = document.getElementById("menu");
let button = document.getElementById("menuBtn");

button.addEventListener("click", function() {
    if (menu.style.display === "none") {
        menu.style.display = "block";
    } else {
        menu.style.display = "none";
    }
});
```

2. **Анімації:** За допомогою JavaScript можна створювати анімації, які додатково покращують досвід користувачів.

```javascript
let box = document.getElementById("box");

function animate() {
    let position = 0;
    let interval = setInterval(function() {
        if (position === 200) {
            clearInterval(interval);
        } else {
            position++;
            box.style.left = position + "px";
        }
    }, 5);
}
animate();
```

Ми розглянули основи веб-програмування, включаючи HTML, CSS та JavaScript. Ці технології є фундаментальними для створення веб-сайтів та веб-додатків.

Тема 4. Конструктори сайтів та CMS

4.1. Мови програмування у веб-розробці
4.2. Огляд популярних конструкторів сайтів
4.3. CMS: Системи керування контентом

4.1. Мови програмування у веб-розробці

Сучасна веб-розробка пропонує кілька підходів до створення сайтів. Розглянемо основні: використання конструкторів сайтів і систем керування контентом (CMS – Content Management System). Конструктори сайтів забезпечують швидке створення веб-ресурсів без програмування, тоді як CMS надають більше гнучкості та можливостей для налаштування.

Розуміння принципів роботи цих інструментів є важливим для маркетологів, підприємців і розробників, які прагнуть ефективно створювати та підтримувати веб-сайти.

Веб-сайти створюються за допомогою різних мов програмування, які працюють на стороні клієнта та сервера. Використання цих мов залежить від обраної технології розробки сайту – чи це буде ручне програмування**,** конструктори сайтів або CMS**.**

Мови програмування:

- JavaScript (Node.js): Використовується для написання серверної частини додатків.
- Python: Використовується разом з фреймворками, такими як Django або Flask.
- PHP: Популярна мова для веб-розробки, використовується з фреймворками, такими як Laravel.
- Ruby: Використовується з фреймворком Ruby on Rails.
- Java: Використовується з фреймворками, такими як Spring.

Фреймворки:

- Express.js: Фреймворк для Node.js.
- Django: Потужний фреймворк для Python.
- Ruby on Rails: Фреймворк для Ruby.
- React.js – створений Facebook, широко використовується для створення інтерфейсів користувача.

Фреймворки (React, Vue.js, Angular, Laravel, Django)

Фреймворки – це структуровані набори інструментів і бібліотек, які спрощують процес створення веб-додатків, забезпечуючи зручність та прискорюючи розробку. Також вони забезпечують швидке завантаження сторінок і кращу взаємодію користувачів із контентом. Особливо важливі для роботи з CMS, кастомними веб-додатками та більш складними веб-проєктами.

- Використовуються для складних, інтерактивних і високонавантажених сайтів.
- Потребують знання програмування, але дають максимальну кастомізацію.
- Використовуються у великих компаніях та стартапах.

Сервери та хостинг:

Веб-сервер – це програмне забезпечення або фізичний сервер, що зберігає файли сайту та відповідає на запити користувачів. Кожен веб-сайт або веб-додаток потребує середовища для зберігання своїх файлів і доступу до них користувачами через Інтернет. Це середовище забезпечується серверами **та** хостингом**.**

Сервер – це потужний комп'ютер або спеціальне програмне забезпечення, яке обробляє запити користувачів і надає їм доступ до веб-сайту.

Види серверів у веб-програмуванні:

Веб-сервер – обробляє HTTP-запити і повертає веб-сторінки (наприклад, Apache, Nginx).

База даних (Database Server) – зберігає інформацію про користувачів, товари, контент сайту (MySQL, PostgreSQL).

Поштовий сервер – обробляє електронну пошту (SMTP, IMAP, POP3).

Файловий сервер – зберігає файли, доступні через Інтернет (FTP-сервер).

Фізичний сервер – реальний комп'ютер у дата-центрі. Дорогий, але дає повний контроль.

Віртуальний сервер (VPS, VDS) – частина фізичного сервера, поділеного між декількома користувачами. Оптимальне рішення за ціною та гнучкістю.

Популярні веб-сервери: Apache, Nginx, LiteSpeed.

- AWS (Amazon Web Services): Потужна платформа для хостингу та хмарних обчислень.
- Heroku: Платформа для розгортання додатків.
- DigitalOcean: Простий у використанні хостинг для розробників.

Хостинг – це послуга оренди простору на сервері для розміщення сайту.

Табл. 4.1. Основні види хостингу

Тип хостингу	Опис	Приклад використання
Віртуальний (Shared hosting)	Один сервер ділиться між кількома сайтами.	Малий бізнес, блоги.
VPS (Virtual Private Server)	Віртуальний сервер з виділеними ресурсами.	Інтернет-магазини, середні проекти.
Виділений сервер	Окремий фізичний сервер для одного проекту.	Великі онлайн-проекти, корпоративні сайти.
Хмарний хостинг	Використання ресурсів кількох серверів одночасно.	Динамічні, масштабовані проекти.

Популярні хостинг-провайдери

- SiteGround, Bluehost, Hostinger – для малого бізнесу та блогів
- DigitalOcean, Linode, Vultr – для розробників і стартапів
- AWS (Amazon Web Services), Google Cloud, Microsoft Azure – для великих проектів і корпоративних рішень.

Як вибрати хостинг для веб-сайту?

Швидкість – Час завантаження сайту має бути мінімальним.

Надійність (Uptime) – Час безперервної роботи сервера має бути 99,9% або більше**.**

Простота керування – Панель управління (cPanel, Plesk) має бути зручною.

Безпека – Підтримка SSL-сертифікатів, резервне копіювання, захист від DDoS-атак.

Підтримка – Краще обирати хостинг з 24/7 підтримкою.

Доменне ім'я та зв'язок із хостингом

Доменне ім'я – це адреса сайту (наприклад, google.com).

1. Користувач вводить URL (наприклад, mysite.com).

2. DNS (система доменних імен) знаходить сервер, на якому розміщений сайт.

3. Сервер відправляє веб-сторінку у браузер користувача.

Отже, підсумуємо:

- Сервер – це основа для зберігання та обробки даних сайту.
- Хостинг – це послуга, яка дозволяє розміщувати сайт на сервері.
- Вибір хостингу залежить від складності проєкту, бюджету та необхідної швидкодії.
- Доменне ім'я – це "адреса" сайту, яка дозволяє користувачам його знайти.

Вибір правильного сервера та хостингу – це ключовий етап у створенні сайту, що впливає на його швидкість, безпеку та доступність для користувачів.

4.2. Огляд популярних конструкторів сайтів

Конструктори сайтів – це платформи, які дозволяють створювати веб-сайти без знань програмування, використовуючи інтуїтивний інтерфейс перетягування елементів (drag & drop). На ринку представлено багато конструкторів сайтів, які мають свої унікальні особливості. У цій таблиці наведені основні платформи, їхні можливості, переваги та недоліки.

Основні характеристики:

- Простота у використанні.
- Вбудовані шаблони та елементи дизайну.
- Обмежена гнучкість налаштувань.
- Не потребують окремого хостингу.

Якщо потрібно швидко створити простий сайт – обирайте Wix. Для інтернет-магазину – найкращий варіант Shopify або Weebly**.** Якщо потрібна повна кастомізація дизайну – використовуйте Webflow**.**

Табл. 4.2. Популярні конструктори сайтів

Назва	Можливості	Переваги	Недоліки	Для кого підходить?
Shopify	Створення онлайн-магазинів, інтеграція з маркетплейсами, підтримка безлічі платіжних систем	✅ Спеціалізований для e-commerce ✅ Великий вибір тем і додатків ✅ Надійна техпідтримка	❌ Висока комісія за транзакції ❌ Обмежені можливості кастомізації дизайну	Підприємці, які продають товари онлайн
Webflow	Професійний веб-дизайн, адаптивність, можливість експорту коду	✅ Висока свобода у дизайні ✅ Можливість експорту HTML, CSS, JS ✅ Підтримка CMS	❌ Вимагає технічних знань ❌ Вища крива навчання	Дизайнери, розробники, корпоративні сайти
Zyro	Швидке створення лендінгів, AI-генерація контенту	✅ Простий у використанні ✅ Доступна ціна ✅ Мінімальні технічні вимоги	❌ Обмежена кількість функцій ❌ Менший вибір шаблонів	Малі бізнеси, персональні сайти
Google Sites	Легке створення сайтів, інтеграція з Google Drive	✅ Безкоштовний ✅ Мінімалістичний дизайн ✅ Простота налаштувань	❌ Дуже обмежена кастомізація ❌ Не підходить для комерційних сайтів	Освітні проекти, внутрішні корпоративні сайти
Wix	Drag-and-drop редактор, велика бібліотека шаблонів, SEO-інструменти	✅ Легкість у використанні ✅ Велика кількість шаблонів ✅ Вбудовані маркетингові функції	❌ Обмежена можливість кастомізації коду ❌ Не можна змінити шаблон після вибору	Фрілансери, малі бізнеси, блогери
Squarespace	Преміальні дизайни, вбудовані маркетингові інструменти	✅ Красиві шаблони ✅ Вбудовані SEO та аналітика ✅ Хороший рівень безпеки	❌ Менше інтеграцій, ніж у конкурентів ❌ Відсутність безкоштовного плану	Художники, фотографи, малі бізнеси

Назва	Можливості	Переваги	Недоліки	Для кого підходить?
WordPress + Elementor	Повноцінна CMS, безліч плагінів та тем, розширена кастомізація	✅ Найгнучкіша система ✅ Велика кількість розширень ✅ Хороший для SEO	❌ Потребує технічних знань ❌ Безпека залежить від хостингу та плагінів	Бізнес, блоги, корпоративні сайти
Framer	Інтерактивні сайти, анімація, інтеграція з Figma	✅ Чудові можливості дизайну ✅ Висока швидкість роботи ✅ Гарний UX	❌ Відносно нова платформа ❌ Менше шаблонів, ніж у Wix або Squarespace	Дизайнери, маркетологи

Ці конструктори дозволяють швидко створювати сайти без необхідності глибокого знання коду. Для маркетологів особливо корисними будуть Wix, Squarespace, Shopify та WordPress + Elementor.

4.3. CMS: Системи керування контентом

CMS (Content Management System – Система управління контентом)

CMS – це програмне забезпечення, яке дозволяє створювати, редагувати та управляти контентом веб-сайту через зручний інтерфейс. На відміну від конструкторів, CMS надають більше можливостей для кастомізації, але вимагають базових знань адміністрування сайтів.

Основні характеристики:

- Гнучкість та розширюваність.
- Велика кількість плагінів і тем.
- Потребує встановлення на хостинг.
- Вимагає базових знань роботи з веб-серверами.

Основні функції CMS:

1. Створення та редагування контенту: CMS дозволяє користувачам створювати, редагувати та публікувати текстові, графічні та мультимедійні елементи на сайті. Веб-редактор (WYSIWYG - What You See Is What You Get) дає змогу зручно працювати з контентом, навіть якщо користувач не має навичок програмування.
2. Управління структурою сайту: CMS дозволяє створювати різні сторінки та розділи сайту, організовувати їх за категоріями, додавати меню та навігацію.
3. Інтеграція з іншими системами: Системи можуть бути інтегровані з іншими інструментами, такими як CRM, ERP, платіжні системи, аналітичні платформи тощо.

4. Користувацькі ролі та дозволи: CMS дозволяє визначати ролі користувачів і налаштовувати рівні доступу, що дає змогу кільком користувачам працювати з контентом на сайті.

5. Мультимовність: Деякі CMS дозволяють створювати багатомовні сайти, що важливо для глобальних компаній або організацій.

Табл. 4.3. Популярні CMS

CMS	Призначення	Переваги	Недоліки
WordPress	Блоги, корпоративні сайти, магазини	🔹 Гнучкість, велика кількість плагінів 🔹 Велика спільнота та підтримка 🔹 SEO-дружня платформа	❌ Потребує окремого хостингу ❌ Необхідне обслуговування плагінів
Joomla	Сайти компаній, соціальні мережі, портали	🔹 Потужні вбудовані можливості 🔹 Багато безкоштовних шаблонів 🔹 Можливість створення багатомовних сайтів	❌ Важчий у налаштуванні, ніж WordPress ❌ Менше плагінів
Drupal	Великі портали, урядові сайти, корпоративні рішення	🔹 Високий рівень безпеки 🔹 Гнучкість та масштабованість 🔹 Добре підходить для великих проектів	❌ Висока крива навчання ❌ Менше шаблонів та плагінів
Magento	Інтернет-магазини	🔹 Орієнтований на e-commerce 🔹 Розширені функції продажу 🔹 Високий рівень кастомізації	❌ Потребує багато ресурсів сервера ❌ Складний для початківців
OpenCart	Малий та середній e-commerce	🔹 Легкий у налаштуванні 🔹 Менше навантажує сервер 🔹 Велика кількість платіжних модулів	❌ Обмежена можливість кастомізації ❌ Менше плагінів, ніж у WordPress
PrestaShop	Середні онлайн-магазини	🔹 Готові рішення для e-commerce 🔹 Дружній до SEO 🔹 Гнучка система плагінів	❌ Висока ціна деяких доповнень ❌ Складніше налаштувати, ніж OpenCart
TYPO3	Великі корпоративні сайти	🔹 Хороша продуктивність 🔹 Розширені можливості адміністрування 🔹 Висока безпека	❌ Вимагає досвіду ❌ Менше популярних ресурсів для навчання
MODX	Гнучкі креативні проекти	🔹 Велика свобода у створенні дизайну 🔹 Хороша продуктивність 🔹 Можливість роботи без плагінів	❌ Складність у вивченні ❌ Вузькоспеціалізоване використання

WordPress – найкращий вибір для блогів, бізнес-сайтів і новачків.
Joomla та Drupal підходять для складних багатосторінкових проектів.
Magento, OpenCart та PrestaShop – оптимальні рішення для e-commerce.

Додаткові критерії вибору CMS або конструктора сайтів.

Якщо ви обираєте між конструктором або CMS, зверніть увагу на такі фактори:

Табл. Порівняння конструкторів сайтів та CMS.

Критерій	Конструктори сайтів	CMS
Легкість у використанні	✅ Дуже прості у налаштуванні	❌ Вимагають базових знань
Вартість	🟠 Може бути вища через підписку	✅ Дешевше, якщо є свій хостинг
Гнучкість дизайну	❌ Обмежена кастомізація	✅ Можна змінювати будь-який елемент
Розширюваність	❌ Мінімальні можливості для додавання нових функцій	✅ Велика кількість модулів і плагінів
Підтримка SEO	🟠 Обмежені можливості	✅ Розширена оптимізація
Безпека	✅ Автоматичні оновлення	🟠 Потрібне адміністрування
Швидкість роботи	✅ Оптимізовані сервери	🟠 Залежить від хостингу

Розвиток сучасного цифрового простору вимагає ефективних інструментів для створення та управління веб-сайтами. У цьому контексті конструктори сайтів **і** системи керування контентом (CMS) відіграють ключову роль, забезпечуючи зручність, гнучкість та доступність веб-розробки як для початківців, так і для досвідчених користувачів.

Вибір між конструкторами сайтів та CMS залежить від специфіки проекту, бюджету та рівня технічної підготовки користувача. Для малого бізнесу**,** персональних сайтів або швидкого запуску ідеальним вибором буде конструктор сайтів**.** Якщо ж необхідно створити багатофункціональний сайт із можливістю розширення, варто звернути увагу на CMS.

Сучасний ринок вимагає від компаній мати зручні, адаптивні та SEO-оптимізовані веб-сайти. Використання веб-платформ дозволяє ефективно просувати продукти, залучати клієнтів та аналізувати поведінку користувачів. У зв’язку з цим, знання про конструктори сайтів і CMS є важливими для маркетологів, підприємців та веб-розробників.

Таким чином, освоєння цих технологій є важливим кроком для будь-якого спеціаліста, який працює у сфері цифрового маркетингу та онлайн-бізнесу.

Тема 5. Основи веб-дизайну та UX/UI

5.1. Принципи дизайну для веб-сторінок
5.2. Основи UX/UI
5.3. Інструменти для створення прототипів (Figma, Adobe XD)

У XXI столітті веб-сайт є не просто цифровою візитівкою компанії, а багатофункціональним комунікаційним та маркетинговим інструментом, що забезпечує першу точку контакту споживача з брендом. Веб-сайт впливає на перше враження, формує довіру, підтримує продажі, автоматизує взаємодію з клієнтом і виконує аналітичні функції.

У цьому контексті веб-дизайн перестає бути винятково сферою інтересів графічних дизайнерів чи програмістів. Він стає обов'язковим предметом розуміння для фахівця з маркетингу, який відповідає за ефективність онлайн-присутності бренду.

Ключ до успішного сайту — поєднання естетичної привабливості (UI) з функціональною ефективністю (UX). Вдале поєднання цих двох аспектів сприяє підвищенню конверсії, покращенню користувацького досвіду і, відповідно, зміцненню конкурентної позиції компанії на ринку.

Ця лекція має на меті сформувати в студентів-маркетологів системне уявлення про:

- фундаментальні принципи сучасного веб-дизайну,
- основи UX/UI та їх значення для користувацького шляху,
- використання інструментів для створення інтерактивних прототипів.

5.1. Принципи дизайну для веб-сторінок

З погляду маркетолога, веб-дизайн є не стільки художньою дисципліною, скільки інструментом впливу на поведінку користувача. Він дозволяє досягати стратегічних цілей: збільшення конверсій, формування лояльності, скорочення шляху до цільової дії. Саме через ефективний дизайн відбувається візуалізація брендових цінностей та адаптація комунікації під потреби цільової аудиторії.

Наприклад, погано організований інтерфейс з плутаною навігацією і непримітною кнопкою «Купити» може звести нанівець усі зусилля з SEO та реклами. Натомість чітка ієрархія, гармонійна кольорова палітра та зручне розташування елементів підвищують залучення й сприяють ухваленню рішень користувачами.

Основні принципи веб-дизайну

Існує кілька універсальних принципів, яких варто дотримуватись при створенні дизайну будь-якого веб-ресурсу. Вони базуються на психології сприйняття, ергономіці та сучасних стандартах юзабіліті.

- Ієрархія

Візуальна ієрархія визначає порядок, у якому користувач сприймає інформацію. Основні повідомлення повинні бути **найпомітнішими**, розміщені у верхній частині сторінки або виділені за допомогою розміру, кольору, шрифту.

Приклад: заголовок H1 має бути більшим і жирнішим за підзаголовки; кнопка «Почати безкоштовно» має контрастувати з фоном.

- Простота

Надмірне ускладнення дизайну лише заплутує користувача. Простий, інтуїтивно зрозумілий інтерфейс дозволяє швидко зорієнтуватись і прийняти рішення. Дизайнери часто дотримуються правила *"одна сторінка — одне основне повідомлення"*.

Приклад: на лендінгу має бути лише один основний CTA («Зареєструватися», «Замовити консультацію»), а не кілька конкурентних.

- Консистентність

Єдність у стилі — ключ до впізнаваності та комфорту користувача. Елементи інтерфейсу (кнопки, заголовки, іконки) повинні виглядати та поводитися однаково на всіх сторінках. Це формує *довіру* та зменшує когнітивне навантаження.

Приклад: кнопка «Купити» на головній і в каталозі має бути однаковою за кольором, формою та поведінкою.

- Контраст

Контраст між фоном і текстом, між основними та допоміжними елементами, дозволяє користувачам легко знаходити потрібне. Контраст слід застосовувати не лише до кольору, а й до *розміру, форми, товщини ліній*.

Приклад: кнопка CTA має бути яскравою (синя, червона), якщо фон — світлий.

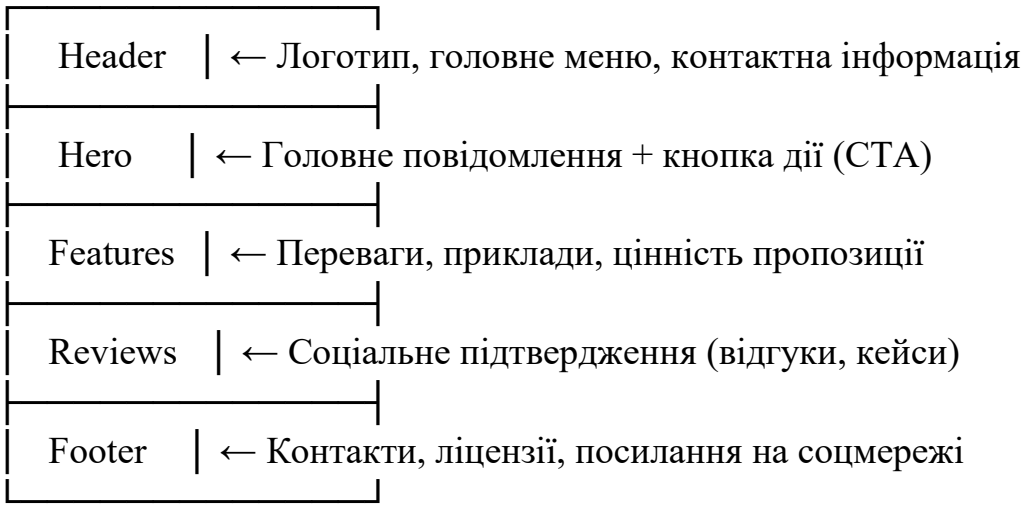
- Адаптивність

Сайт повинен коректно відображатися на всіх типах пристроїв — від великих моніторів до смартфонів. Адаптивний дизайн (responsive design) — це не тренд, а необхідність, зважаючи на переважання мобільного трафіку у більшості ніш.

Таблиця 5.1. Основні принципи веб-дизайну

Принцип	Пояснення
Ієрархія	Користувач має отримувати інформацію у порядку її важливості. Найголовніше — вище, яскравіше, більшим шрифтом. Це дозволяє швидко зорієнтуватись на сторінці й не витрачати час на пошук.
Простота	Мінімізація інформаційного навантаження. Один екран — одне повідомлення. Надмірна кількість тексту, кнопок чи банерів лише відволікає увагу. Прості дизайни підвищують конверсію.
Консистентність	Всі елементи дизайну повинні бути стилістично узгодженими. Однакові кольори, шрифти, поведінка кнопок. Це формує очікування користувача та створює відчуття професійності.
Контраст	Забезпечує читабельність і виділення ключових елементів. Наприклад, темний текст на світлому фоні або яскрава кнопка на нейтральному тлі.
Адаптивність	Сайт повинен виглядати добре на різних пристроях — від комп'ютера до смартфона. Це не тільки питання зручності, а й фактор рейтингу у пошукових системах.

Схема: Структура типової лендінг-сторінки



Таке компонування — не випадкове. Воно базується на зоні уваги користувача, яка природно рухається згори вниз. Найважливіше — на початку.

5.2. Основи UX/UI

Успіх будь-якого цифрового продукту — сайту, мобільного застосунку чи електронного сервісу — безпосередньо залежить від якості досвіду користувача (UX) та зручності візуального інтерфейсу (UI). Для маркетолога це означає не лише

вивчення поведінки відвідувачів, а й участь у формуванні середовища, яке стимулює конверсію, задоволення та лояльність клієнта.

UX (User Experience або досвід користувача) — це сукупність вражень, які отримує користувач під час взаємодії з продуктом. Сайт із якісним UX — це сайт, де все інтуїтивно зрозуміло, швидко, зручно, передбачувано.

Ключові компоненти UX:

- Юзабіліті (usability): зручність використання — наскільки легко виконати цільову дію.
- Інформаційна архітектура: логічна структура контенту, ієрархія сторінок, навігація.
- Шлях користувача (user journey): сценарії, якими користувачі рухаються до мети (наприклад, від головної сторінки до покупки).
- Персонажі (user personas): уявні типові представники аудиторії з визначеними болями, потребами та поведінкою.
- Фідбек: наявність зворотного зв'язку після дії (повідомлення про помилку, підтвердження успіху тощо).
- Юзабіліті-тестування: спостереження за користувачами в реальному часі, виявлення бар'єрів і непорозумінь у взаємодії з інтерфейсом.

Приклад:

Уявімо, що користувач заходить на сайт, щоби записатися на безкоштовну консультацію. Якщо йому потрібно зробити більше трьох кліків, форма занадто велика, а кнопка «Надіслати» малопомітна — UX поганий. Якщо ж форма коротка, навігація логічна, а підтвердження приходить миттєво — UX хороший.

📌 *Хороший UX — це той, який користувач навіть не помічає.*

UI (User Interface або інтерфейс користувача) — це те, що бачить користувач: графічні елементи, кольори, типографіка, візуальна структура, кнопки, іконки. UI не менш важливий, ніж UX, адже саме зовнішній вигляд створює перше враження і сприяє впізнаваності бренду.

Основні елементи UI:

- Кольорова палітра: підтримує фірмовий стиль, задає настрій (наприклад, синій — довіра, зелений — екологічність, червоний — дія).
- Типографіка: вибір шрифтів і їх комбінація, читабельність на різних пристроях.
- Іконографіка: піктограми для швидкої навігації та сприйняття.
- Сітка (grid): структура розміщення блоків, яка забезпечує гармонію, логіку і гнучкість при адаптації.
- Анімація: ефекти при наведенні, плавні переходи, які покращують взаємодію.

Таблиця 5.2. Порівняння UX і UI:

UX	UI
Планує шлях користувача	Формує зовнішній вигляд цього шляху
Фокус на зручності	Фокус на візуальній привабливості
Визначає, як усе працює	Визначає, як усе виглядає
Створює логіку, сценарії	Створює кнопки, кольори, шрифти
Адаптується до потреб користувача	Адаптується до фірмового стилю

Для маркетолога UX/UI — це не технічна теорія, а реальний важіль впливу на результати:

- Конверсія: зрозумілий інтерфейс і простий шлях до кнопки «Купити» можуть збільшити конверсію вдвічі.
- Лояльність: зручний інтерфейс формує позитивне враження → збільшує повторні візити.
- SEO: Google враховує поведінкові фактори, тож UX впливає на позиції в пошуку.
- Реклама: реклама веде на посадкову сторінку, а UI вирішує, чи затримається користувач там.

Залежність конверсії від часу взаємодії

Час до виконання цільової дії (CTA):	Конверсія:
до 10 сек	~ 3,5%
20–30 сек	~ 2,1%
понад 40 сек	~ 0,7%

Чим швидше і простіше користувач знаходить, що шукає — тим вищий відсоток виконання бажаної дії.

5.3. Інструменти для створення прототипів (Figma, Adobe XD)

На етапі розробки веб-сайту важливо ще до програмування візуалізувати структуру, логіку та вигляд інтерфейсу. Саме для цього створюються прототипи — інтерактивні макети, що дозволяють учасникам команди погодити основні елементи майбутнього сайту, скоротити час і знизити ризики помилок.

Прототип — це не просто “картинка сайту”. Це засіб комунікації між маркетологом, дизайнером, програмістом і клієнтом.

Чому прототипування важливе

1. Знижує вартість розробки. Виправити помилку на рівні ескізу в Figma — набагато дешевше, ніж переписувати код.

2. Погодження логіки. Прототип дозволяє маркетологу перевірити, чи відповідає структура сайту очікуваним сценаріям користувача.

3. Видимість. Навіть непрофесіонал (замовник, SMM-фахівець) може оцінити майбутній інтерфейс без технічних знань.

4. Економія часу. Програміст працює за готовим прототипом, без "вгадувань" щодо того, як мають виглядати кнопки, блоки, меню.

Табл. 5.3. Види прототипів

Тип прототипу	Характеристика	Коли застосовується
Низької точності	Простий скетч або wireframe	На стадії планування, аналізу структури
Середньої	Вже схожий на інтерфейс, але без стилю	Для тестування функціональності
Високої точності	Реалістичний макет із анімацією, переходами	Перед передачею у розробку

📌 Маркетологу найчастіше доводиться працювати з прототипами середньої або високої точності, бо саме вони імітують справжній інтерфейс.

Інструменти для прототипування

Найпопулярнішими інструментами в сучасній практиці є Figma і Adobe XD. Вони дозволяють створювати інтерактивні макети, тестувати навігацію, а також працювати в команді над єдиним проектом.

Інструмент	Платформа	Особливості	Переваги для маркетолога
Figma	Онлайн	Спільна робота в браузері	Безкоштовна, не потребує встановлення, зрозуміла
Adobe XD	Десктоп	Професійне середовище дизайну	Інтеграція з Adobe-пакетом, офлайн-доступ

Практичний приклад: Робота з Figma

Сценарій:

1. Маркетолог створює персонаж цільової аудиторії (наприклад: "Ольга, 29 років, шукає курси онлайн").

2. Створює прототип головної сторінки в Figma:

- Hero-блок: заголовок "Освіта майбутнього — сьогодні", кнопка "Записатися".
- Блок переваг: "Диплом", "Онлайн-формат", "Гарантія результату".
- CTA: форма запису.

3. Спільно з дизайнером переглядає варіанти стилів кнопок, кольорових схем.

4. Після затвердження передає макет розробнику.

📌 *Усе — без написання жодного рядка коду.*

Схема: Як виглядає типовий цикл створення інтерфейсу

Маркетингова задача

↓

UX-дослідження (персонажі, сценарії, карта кліків)

↓

Прототип (Figma / XD)

↓

UI-дизайн (стилізація, колір, форма)

↓

Передача у фронтенд-розробку

Додаткові можливості для маркетолога:

- Створення банерів, лендингів, email-шаблонів у Figma.
- Розмітка сеток рекламних кампаній (наприклад, Facebook Ads).
- Тестування нових версій елементів (наприклад, дві кнопки різного кольору для A/B тесту).
- Співпраця з командою в реальному часі — коментування, версійність, обговорення змін.

Сайт Weblium.com (український конструктор сайтів) демонструє приклад якісного застосування принципів веб-дизайну. На головній сторінці реалізована чітка ієрархія: яскравий CTA у Hero-блоці, послідовна структура секцій, читабельний текст, єдність стилю. Завдяки цьому сайт забезпечує зручність користування і високу ефективність взаємодії з користувачем.

Прототипування — це обов’язковий етап цифрового проєкту. Воно дозволяє маркетологу брати участь у створенні інтерфейсу, а не лише в його аналізі після запуску. Інструменти типу Figma і Adobe XD стали невід’ємною частиною щоденної практики. Вони дозволяють втілювати ідеї швидко, ефективно й узгоджено — що є критично важливим для реалізації якісного маркетингового продукту.

Тема 6. Основи SEO (Search Engine Optimization) *просування*

6.1. Вступ до пошукової оптимізації.
6.2. Ключові фактори SEO.
6.3. Оптимізація контенту та структури сайту для пошукових систем.

У сучасному цифровому маркетингу *SEO (Search Engine Optimization)* є однією з ключових стратегій залучення органічного трафіку — тобто відвідувачів, які знаходять сайт через пошукові системи (Google, Bing, Ecosia тощо).

🌐 SEO — це мистецтво бути першим у Google тоді, коли це має значення.

Пошукова оптимізація дозволяє підвищити видимість сайту, збільшити кількість переходів та знизити вартість залучення ліда у порівнянні з платною рекламою.

6.1. Вступ до пошукової оптимізації.

SEO (Search Engine Optimization) — це процес оптимізації вебсайту з метою покращення його видимості в органічних (неоплачених) результатах пошуку. Найпопулярніша пошукова система — це, безперечно, **Google**, який обробляє понад 90% глобальних пошукових запитів. Але також існують Bing, Yahoo, DuckDuckGo, Ecosia тощо.

📌 Простими словами: SEO допомагає зробити так, щоб сайт знаходили ті, хто шукає інформацію, пов’язану з вашим продуктом чи послугою.

Можна створити найкращий сайт, але якщо його ніхто не знайде, він не працює на бізнес. SEO — це фундамент цифрової присутності бренду. Для маркетолога це означає:

- Залучення безкоштовного (органічного) трафіку;
- Підвищення впізнаваності бренду;
- Формування довіри (користувачі довіряють топовим результатам);
- Зменшення вартості залучення клієнта в довгостроковій перспективі.

📈 Аналітика показує: понад 70% кліків припадає на перші 5 результатів пошуку, тоді як 90% користувачів не переходять на другу сторінку результатів Google.

Пошукова система — це складна екосистема. Вона сканує (crawl), індексує (index) та ранжує (rank) вебсторінки.

Якщо сторінку не можна “пройти” або “зрозуміти” ботом — вона не потрапить у видачу. Тому важливо оптимізувати як контент, так і код та структуру сайту.

- Пошуковик не "бачить" сайт так, як людина. Для нього важливі структура, метадані, код, швидкість.

- Якщо сторінка не проходить етап *Crawling* або *Indexing* — її не існує для Google, незалежно від її краси чи цінності.

Власне тому SEO — це комунікація не тільки з користувачем, а й з алгоритмами.

🔍 *Етапи роботи пошукової системи:*

Етап	Опис	Приклад або коментар
Crawling	Сканування сайту пошуковими ботами (Googlebot, Bingbot тощо). Бот переходить по посиланнях і «читає» сторінки.	Якщо сторінка ізольована (не має посилань) або закрита в robots.txt — її не сканують.
Indexing	Збереження змісту сторінки в індексі Google. Індексація означає, що сторінку вже можна показувати в пошуку.	Якщо сторінка без title, description, або має дублі — індексація буде неповною.
Ranking	Алгоритм Google визначає, яку сторінку і на якому місці показати за певним запитом.	Понад 200 факторів ранжування, серед яких — релевантність, авторитетність, UX тощо.
Serving Results	Пошукова система показує користувачеві релевантні результати, враховуючи його намір, місцезнаходження, пристрій тощо.	Два користувачі можуть отримати різні результати за однаковим запитом.

Сучасні тренди SEO

Що важливо у сучасних тенденціяї:

- Core Web Vitals (зручність користування, швидкість, стабільність інтерфейсу)
- Mobile-first indexing (Google ранжує за мобільною версією!)
- Пошук за наміром (Search Intent): не лише ключові слова, а ціль користувача
- Змістовні тексти замість “наповнення ключами”
- E-E-A-T: Experience, Expertise, Authoritativeness, Trust — довіра до джерела

Типи SEO

Важливо розрізняти три ключові типи SEO, які потрібно поєднувати у стратегії:

Табл. 6.1. Типи SEO

Тип SEO	Характеристика	Хто відповідальний
On-page SEO	Оптимізація контенту, ключових слів, мета-тегів	Копірайтер, SEO-фахівець
Off-page SEO	Отримання зовнішніх посилань, PR-активність	PR, маркетолог
Technical SEO	Технічна якість сайту: швидкість, адаптивність, код	Розробник, SEO-фахівець

Ключова відмінність: SEO ≠ реклама

SEO працює повільніше, але довготриваліше. Це інвестиція у зростання, а не витрати на кліки.

Табл. 6.2. Порівняння SEO vs Реклама (PPC):

Параметр	SEO	PPC (Google Ads, Meta Ads)
Час ефекту	Середньо/довгостроковий	Миттєвий
Ціна	Дешевше в перспективі	Дорожче, але швидко
Контроль	Обмежений (Google змінює алгоритми)	Повний (керування ставками)
Надійність	Висока після досягнення позицій	Тимчасова (поки є бюджет)

SEO — це не “чарівне налаштування”, а системна робота з контентом, структурою і кодом**.** Основне завдання маркетолога — розуміти логіку SEO і вміти формувати якісні ТЗ, аналізувати сайти, працювати з аналітикою. Без SEO цифрова стратегія — як бізнес без вивіски: існує, але ніхто не бачить.

6.2. Ключові фактори SEO.

Щоб сайт успішно з’являвся в результатах пошуку, його потрібно оптимізувати за низкою ключових параметрів. Ці фактори впливають на те, наскільки високо Google або інші пошукові системи поставлять сторінку у своїй видачі.

Всі фактори можна умовно поділити на такі групи:

On-page SEO (внутрішня оптимізація)

Це всі дії, які відбуваються безпосередньо на сторінці сайту. Їх завдання — зробити сторінку максимально зрозумілою для пошукових роботів та привабливою для користувачів.

Сюди входить:

- оптимізація заголовків (<title>, <h1>, <h2> тощо);
- мета-описи (<meta description>);
- правильна структура URL;
- ключові слова (розміщення в тексті, заголовках, ALT-атрибутах);
- внутрішні посилання;

- оптимізація зображень (розмір, формат, назва, alt).

📌 Приклад:

Добре: <title>10 порад для покращення SEO у 2025 році</title>

Погано: <title>Головна сторінка</title>

Content — зміст сторінки

Google навчився розуміти суть і якість контенту, а не лише кількість ключових слів.

Правило: Пишемо для людей, оптимізуємо для Google.

SEO-контент має бути:

- унікальним (ніякого копіювання);
- релевантним (відповідати запитам користувачів);
- структурованим (абзаци, заголовки, списки, таблиці);
- достатньо об'ємним (понад 800–1000 слів — для статей).

Google також оцінює:

- чи легко читається текст,
- чи є в ньому структура,
- як довго користувач затримується на сторінці.

Technical SEO (технічна оптимізація). Технічне SEO забезпечує доступність і зручність сканування сайту.

Основні пункти:

- швидкість завантаження (PageSpeed Insights);
- мобільна адаптивність (Mobile-Friendly Test);
- відсутність дублікатів;
- наявність SSL-сертифіката (https://);
- правильна структура HTML;
- sitemap.xml і robots.txt;
- використання мікророзмітки (schema.org);
- чисті URL (людинозрозумілі посилання).

📌 Приклад поганого – URL: https://site.com/?id=789&cat=5

Приклад хорошого – URL: https://site.com/marketing/seo-osnovy

Off-page SEO (зовнішні фактори)

Це все, що відбувається за межами сайту, але впливає на його рейтинг.

Основний фактор — зовнішні посилання (backlinks). Чим більше авторитетних сайтів посилається на вашу сторінку, тим більше довіри вона отримує від Google.

✓ Добрі посилання:

- з тематичних блогів, ЗМІ, партнерів;
- розміщені в контексті;
- без зловживання анкорами.

❌ Погані посилання:

- куплені;
- зі спам-ресурсів;
- з неякісних бірж.

📈 Інші фактори:

- згадування бренду;
- активність у соцмережах (не напряму впливає, але підсилює присутність);
- Google Business Profile для локального SEO.

User Experience (UX) — поведінкові фактори. Google оцінює поведінку реальних користувачів: чи залишаються вони на сайті, чи натискають на щось, чи повертаються назад у пошук (pogo-sticking).

Основні показники:

- CTR (click-through rate) — скільки людей клікнули по вашому результату в пошуку;
- Показник відмов (bounce rate) — скільки одразу вийшли;
- Середній час на сторінці — чим більше, тим краще.

SEO — це багатогранна система, у якій кожен елемент відіграє роль у досягненні високих позицій у пошуковій видачі.

Важливо розуміти, що:

- Контент є основою, але без технічної оптимізації навіть найкраща стаття може залишитись «невидимою» для Google;
- Зовнішні посилання — це «голоси довіри» з боку інших сайтів;
- UX-фактори — сигнал про якість із точки зору користувача;
- Маркетолог має виступати не просто замовником SEO, а співучасником цього процесу, здатним критично оцінити якість оптимізації.

Табл. 6.3. ТОП-5 ключових факторів SEO

📌 Фактор	🧭 Приклади	🎯 Вплив на SEO
Контент	Унікальні, структуровані статті	🔥🔥🔥
Ключові слова	Оптимізовані заголовки, природні ключі	🔥🔥
Швидкість завантаження	Показники Core Web Vitals	🔥🔥
Backlinks	Авторитетні посилання з тематичних ресурсів	🔥🔥🔥
Мобільна адаптивність	Адаптація під смартфони	🔥🔥

📌 *Комплексний підхід до SEO — запорука стабільного і довготривалого зростання видимості в Інтернеті.*

6.3. Оптимізація контенту та структури сайту для пошукових систем.

Контент і структура — це ті дві площини, у яких маркетолог реально може і повинен брати участь: формувати вимоги, створювати якісний зміст і забезпечувати логічну організацію сторінок**.**

Контент — це не лише «текст для читання». Це інформаційна відповідь на запит користувача. Google оцінює не лише наявність ключових слів, а й змістовність, логічність і корисність матеріалу.

📌 Рекомендація: Кожна сторінка має бути оптимізована під одне основне ключове слово (або фразу) + синоніми, пов’язані терміни (LSI).

Основні вимоги до контенту:

🎯 Критерій	📌 Коментар
Унікальність	Текст має бути оригінальним. Плагіат або копіпаст — причина деіндексації.
Структурованість	Використання заголовків (H1, H2, H3), списків, абзаців, блоків.
Цінність	Контент має відповідати на запит користувача, а не просто повторювати ключові слова.
Оптимальна довжина	Середня довжина ТОП-статті — 1000–2500 слів. Але важливий не обсяг, а якість.
Ключові слова	Вживати природно в заголовках, перших абзацах, ALT-зображеннях.
Візуалізація	Зображення, графіки, іконки, таблиці покращують сприйняття і поведінкові фактори.

Приклад: Правильна структура SEO-статті

```
H1: Як просувати сайт у Google у 2025 році

- Вступ (опис проблеми, цілі статті)
- H2: Що таке SEO?
- H2: Основні чинники ранжування
- H2: Покрокова стратегія просування
  - H3: Технічна оптимізація
  - H3: Контентна стратегія
  - H3: Лінкбілдинг
- H2: Часті помилки новачків
- H2: Висновки
- CTA: Отримати безкоштовний аудит
```

Оптимізація структури сайту

Правильна структура сайту не лише допомагає користувачеві орієнтуватись, а й значно полегшує індексацію для пошукових роботів.

Приклад правильної URL-структури:

❌ Погано	✅ Добре
`site.com/index.php?page=12`	`site.com/seo/osnovy`
`site.com/post?id=product123`	`site.com/shop/seo-tools/serp-analyzer`

🔧 Елементи правильної структури:

🧱 Елемент	📌 Значення
Логічна ієрархія сторінок	Категорії → Підкатегорії → Товари / Статті. Не більше 3 рівнів вкладеності.
Breadcrumbs (хлібні крихти)	Навігаційна підказка для користувача і пошуковика.
Внутрішнє перелінкування	Посилання між пов'язаними сторінками. Покращує індексацію і SEO-вплив.
URL-структура	Коротка, зрозуміла, містить ключове слово.
Sitemap.xml	XML-файл зі списком усіх сторінок сайту для пошукових систем.
robots.txt	Файл для заборони сканування службових частин сайту.
Мобільна адаптація	Сайт має однаково зручно працювати на смартфоні і ПК.

Часті помилки у структурі сайту:

- надмірно складна або плутана навігація;
- дублі сторінок, неканонічні URL;
- відсутність головних елементів (title, description);
- відсутність мобільної версії (критично у 2025 р.);
- ланцюжки переадресацій, биті посилання.

Контент і структура сайту — це перші два стовпи якісної SEO-стратегії. Саме на них маркетолог має реальний вплив:

- Контент — через формування ТЗ, написання, редагування;
- Структура — через співпрацю з дизайнером і розробником.

📌 *SEO — це не лише справа програмістів. Це зона відповідальності всього маркетингового відділу.*

Тема 7. Проектування та тестування веб-додатків

7.1. Основи проектування та тестування веб-додатків.

7.2. Використання автоматизованих тестових інструментів.

7.3. Тестування взаємодії: користувач і додаток. Оцінка зручності використання.

7.4. Запуск додатка та його підтримка

7.1. Основи проектування та тестування веб-додатків.

Перш ніж щось тестувати, потрібно його спроєктувати. У веброзробці це означає не лише створення красивого макету, а розробку логіки, структури та сценаріїв взаємодії користувача з продуктом.

Проєктування веб-додатку — це процес планування:

- архітектури майбутнього інтерфейсу;
- логіки переходів між сторінками;
- даних, які будуть оброблятися;
- ключових точок взаємодії з користувачем (CTA, реєстрація, запит тощо).

Важливо: маркетолог бере участь у проєктуванні через формування user flow**,** персонажів**,** користувацьких сценаріїв**,** а також участь у UX-дослідженнях.

Етапи проєктування веб-додатку:

🔢 Етап	📌 Що виконується
1. Визначення цілей	Яку бізнес-проблему вирішує додаток? Хто цільова аудиторія?
2. Складання user flow	Яким шляхом рухається користувач? Від старту до цільової дії?
3. Визначення функцій	Які функції має реалізовувати система? (реєстрація, завантаження, чат)
4. Побудова структури	Архітектура сторінок, блоків, форм, зв'язків.
5. Прототипування (Wireframes)	Створення ескізів майбутніх інтерфейсів.
6. Валідація (тестування макетів)	Перевірка прототипу на реальних користувачах.

📌 Приклад: User Flow для вебінарної платформи

Головна → Кнопка "Зареєструватись" → Форма → Підтвердження поштою → Особистий кабінет

На кожному етапі маркетолог оцінює: чи все зрозуміло? чи є мотивація? чи є довіра? чи можна щось скоротити?

Проєктування — це основа якісного веб-додатку**.** Якщо цей етап пройдено невдало, то й найкраще тестування не врятує продукт.

Маркетолог не є дизайнером або програмістом, але він має стратегічну роль — формувати логіку взаємодії з точки зору цільової аудиторії.

Гарний додаток починається не з коду — а з розуміння користувача.

У сучасному цифровому середовищі веб-додатки — це інтерактивні онлайн-інструменти, які обробляють дані користувачів, надають персоналізовані функції, комунікують з базами даних. До них належать:

- системи бронювання;
- онлайн-магазини;
- банківські додатки;
- сервіси запису;
- особисті кабінети;
- освітні платформи.

Роль маркетолога в проєктуванні

🧩 Завдання	📍 Приклад дій маркетолога
Аналіз аудиторії	Створення персонажів, аналіз болей і потреб
User flow	Побудова сценаріїв — «як користувач рухається до покупки»
Визначення CTA	Розміщення кнопок, форм, мотиваційних закликів
Перевірка структури	Участь у складанні логічної навігації
Тестування прототипу	Проведення глибинних інтерв'ю, спостереження

У таких додатках навіть незначна помилка може спричинити втрату довіри, даних чи грошей. Тому тестування є невіддільною частиною життєвого циклу веб-додатку — на кожному етапі, від прототипу до запуску.

Тестування веб-додатків — це процес перевірки того, як працює додаток відповідно до вимог, очікувань користувача та технічних стандартів.

📌 Мета тестування: виявити помилки до того, як їх виявлять користувачі.

Тестування охоплює не лише перевірку "чи працює кнопка", а й:

- перевірку відповідності дизайну (UI) та зручності (UX);
- логіку запитів і виводу даних;
- обробку граничних ситуацій (наприклад, введення неправильного e-mail);
- стабільність під час навантаження;
- безпеку при передачі персональних даних.

У великих IT-компаніях для цього є QA-інженери (Quality Assurance). Однак маркетолог або менеджер проекту також може й повинен брати участь у процесі перевірки, зокрема:

- формувати чек-листи для тестування з погляду користувача;
- моделювати типові сценарії взаємодії;
- перевіряти реалізацію цільових дій (CTA);
- узгоджувати очікувану поведінку системи з дизайнером і розробником.

Навіть якщо маркетолог не програмує, він має чітко розуміти, що:

- невірно реалізований UX → менше конверсій;
- неперевірені форми → втрата лідів;
- поганий мобільний досвід → зниження в SEO та рекламі;
- неправильне передавання даних → неможливість аналізу в CRM чи аналітиці.

Класифікація тестування веб-додатків

🧪 Вид тестування	🔍 Що перевіряє	📌 Приклад
Функціональне	Чи працюють основні функції згідно з вимогами	Кнопка "Відправити" відправляє форму
Тестування юзабіліті	Наскільки зручний та зрозумілий інтерфейс для користувача	Чи легко знайти кнопку реєстрації
Тестування безпеки	Чи захищені дані від стороннього втручання	Чи не можна отримати чужу інформацію
Мобільне тестування	Чи коректно відображається додаток на мобільних пристроях	Чи адаптується сторінка до екрану
Кросбраузерне	Чи працює додаток однаково в Chrome, Safari, Firefox тощо	Чи правильно відображаються стилі
Регресійне тестування	Чи збереглася працездатність після змін або оновлень	Чи не «зламалась» форма після редизайну

Тестування — це не технічна формальність, а критичний етап якості цифрового продукту**.** Маркетолог, який вміє ставити правильні запитання і тестувати інтерфейс очима клієнта, — незамінний для команди**.**

Навіть найкреативніший вебдизайн втрачає сенс, якщо кнопка не працює або форма не надсилає дані.

7.2. Використання автоматизованих тестових інструментів.

Автоматизоване тестування — це процес перевірки працездатності веб-додатка за допомогою спеціальних програм, які імітують дії користувача, запускають функції, перевіряють правильність результатів — усе це без втручання людини.

Його головна мета — швидко та надійно протестувати великий обсяг функціоналу, особливо після змін або оновлень.

Це дозволяє:

- економити час на рутинних перевірках;
- зменшити людський фактор (помилки ручного тестування);
- отримати об’єктивні та повторювані результати;
- швидше запускати нові версії продукту.

Приклад: Lighthouse-тест сайту
Інструмент: Lighthouse (Chrome DevTools)

📌 Що він аналізує:

- продуктивність (час завантаження);
- доступність (чи можна працювати з клавіатурою);
- SEO-фактори (теги, мобільність);
- PWA (якщо є офлайн-режим).

Приклад оцінки:
Performance: 92
Accessibility: 87
Best Practices: 100
SEO: 96
Такий звіт допомагає не тільки програмісту, а й маркетологу, який хоче покращити досвід користувача або SEO.

Види автоматизованого тестування

🔍 Тип тесту	📌 Що перевіряє	📈 Коли застосовують
Unit tests	Окремі модулі або функції (логіка, калькулятор, перевірка пошти)	На рівні програміста
Integration tests	Взаємодію між частинами (форма → БД → email)	При об’єднанні компонентів
End-to-end tests	Повний сценарій користувача (від входу до підтвердження дії)	Перед релізом, для перевірки UX
Performance tests	Стабільність під навантаженням	Перед масштабуванням або піковими навантаженнями
Regression tests	Чи все працює після оновлення коду	Після кожного спринту/редизайну

Популярні інструменти автоматизації

🛠️ Інструмент	💻 Категорія	📌 Призначення	🌟 Переваги
Selenium	End-to-end	Автоматичне клікування, введення даних у браузері	Працює з Chrome, Firefox, Safari
Cypress	End-to-end	Тестування фронтенду на JavaScript	Дуже швидкий, інтегрується з React
Jest	Unit / Integration	Тестування JS-функцій	Популярний серед розробників Frontend
Postman/ Newman	API testing	Перевірка запитів до серверу	Без коду, зручний для маркетолога
Lighthouse	Performance / SEO	Оцінка продуктивності, доступності, UX, SEO	Вбудований у Chrome, видає оцінки

Автоматизоване тестування — це не "іграшка розробника", а життєво необхідний елемент якісного цифрового продукту**.**

7.3. Тестування взаємодії: користувач і додаток. Оцінка зручності використання.

Що таке тестування взаємодії? Це процес перевірки того, як користувач реально взаємодіє з веб-додатком**.** Йдеться не лише про технічну справність кнопок, а про логіку, зручність, емоції та передбачуваність взаємодії.

🎯 Завдання — виявити бар’єри, що заважають користувачу досягти цілі.

Основні сценарії для перевірки

🔍 Що тестується	🧪 Приклад запитання під час тестування
Форма реєстрації/підписки	Чи всі поля працюють? Чи є підказки? Що буде, якщо залишити поле порожнім?
Кошик і оформлення замовлення	Чи зрозуміло, що робити далі? Чи не "зникає" товар?
CTA-кнопки	Чи зрозумілий текст? Чи помітна кнопка? Чи викликає вона довіру?
Пошук/фільтри/навігація	Чи можна легко знайти потрібне? Як швидко? Чи не "губиться" користувач?
Повідомлення про помилки/успіх	Чи достатньо інформативні? Чи не лякають?
Інтерактивні елементи (чат, вкладки)	Чи не зависають? Чи зручно з мобільного?

Методи тестування UX

Маркетолог може використовувати не технічні, а поведінкові підходи**,** які дозволяють виявити реальні проблеми:

Метод	Пояснення
User testing	Дати користувачу завдання й спостерігати, як він його виконує
Сценарії (use cases)	Симуляція типових дій (знайти товар, зареєструватись, залишити відгук)
Аналітика поведінки	Використання Google Analytics, Hotjar, Clarity для відстеження кліків
Теплові карти (heatmaps)	Показують, де користувачі клікають, скролять, затримуються
Опитування/зворотний зв'язок	Короткі форми на сайті, чат-бот, інтерв'ю

Як виглядає простий юзабіліті-тест (покроково)

Сценарій: протестувати процес реєстрації на сайті онлайн-курсів

Кроки:

1. Дати людині завдання: «Зареєструйся на курс "Маркетинг 2025"».
2. Спостерігати:
   - чи розуміє вона, де це зробити;
   - як реагує на форму;
   - що її зупиняє.
3. Записати:
   - час виконання;
   - емоційні реакції (здивування, роздратування);
   - фрази типу «а де ж кнопка?».
4. Після дій — коротке інтерв'ю: що було зручно / що незрозуміло.

Після тестування важливо сформулювати конкретні рекомендації: що саме потрібно покращити.

Маркетолог має узагальнити результати в простій формі:

📌 Проблема	🛠️ Причина	💡 Рекомендація
Користувачі не бачать кнопку реєстрації	Занадто сірий колір, нижче згину сторінки	Підняти кнопку вище, змінити колір
Часто помилки в e-mail	Немає прикладу або маски у полі	Додати автоперевірку, підказку
Високий bounce rate	Надто довге завантаження	Оптимізувати зображення

UX-тестування — це міст між продуктом і реальним користувачем. І саме маркетолог відіграє ключову роль у тому, щоб користувач не просто «знайшов» сайт, а досяг там мети зручно і безперешкодно.

UX — це не "красива кнопка", а зручний і логічний шлях до дії.

Оцінка зручності використання (Usability Evaluation)

Зручність (usability) — це якість користувацького досвіду, яка визначає, наскільки легко, швидко та приємно користувач може досягти своїх цілей при роботі з додатком.

📌 Якщо функція є, але нею складно користуватись — вона неефективна. Маркетолог повинен уміти оцінювати не лише присутність функціоналу, а й зручність його використання.

Ключові параметри зручності

🧪 Критерій	🔍 Питання для оцінки
Доступність	Чи можна швидко знайти функцію?
Передбачуваність	Чи зрозуміло, що станеться після кліку?
Швидкість	Скільки часу потрібно, щоб досягти результату?
Помилкостійкість	Чи можна легко виправити помилку без втрати даних?
Підказки/зворотний зв'язок	Чи система пояснює, що відбувається?

Методи оцінки зручності

🛠️ Метод	📌 Як застосовувати
Heuristic evaluation	Перевірка за 10 принципами Юзабіліті (за Нільсеном)
Опитування користувачів (CSAT, SUS)	Опитування після дії: наскільки зручно було виконати задачу
Запис екрана (session recording)	Аналіз поведінки користувача в реальному часі
Аналіз помилок (Error analysis)	Де найчастіше користувачі роблять помилки?
Task completion rate (TCR)	% користувачів, які змогли успішно завершити дію

Приклад: SUS (System Usability Scale)

Маркетолог може провести опитування на 10 запитань із п'ятибальною шкалою (від "повністю не погоджуюсь" до "повністю погоджуюсь"). Це дозволяє отримати узагальнену оцінку юзабіліті в балах від 0 до 100.

Приклад запитань:

1. Я часто хочу використовувати цей додаток.
2. Інтерфейс здався мені простим.
3. Мені потрібно багато навчання, щоб використовувати додаток.
4. Усі функції працюють узгоджено.
5. Я впевнений у своїх діях під час користування.

📈 Показник понад 70 балів SUS — ознака хорошої зручності.

Оцінка зручності — це критично важлива відповідальність маркетолога, оскільки вона безпосередньо впливає на:

- рівень задоволеності користувача;
- повторне використання (retention);
- кількість успішних цільових дій (конверсії);
- зменшення навантаження на службу підтримки.

Чим зручніше додаток — тим менше потрібно пояснень, тим вища конверсія, тим кращий відгук.

7.4. Запуск додатка та його підтримка

Продакшн-сервер — це основне середовище, де додаток стає доступним для користувачів. Перед цим він існує лише в локальному середовищі (на комп'ютері розробника) або на тестовому сервері (staging**)**, де команда перевіряє всі функції.

Етапи розгортання веб-додатку

📌 Крок	📝 Що відбувається
1. Підготовка коду	Завершення розробки, фінальні правки, версія релізу
2. Перевірка на staging	Останнє тестування всієї системи в середовищі, максимально наближеному до реального
3. Backup	Резервне копіювання сайту, щоб уникнути втрати даних
4. Розгортання (deployment)	Перенесення коду на сервер із доменом і публічним доступом
5. Перевірка продуктивності	Перевірка швидкості, SEO, адаптивності, безпеки
6. Сповіщення користувачів	Банер або повідомлення про запуск або оновлення

Маркетолог бере участь у комунікаціях, контролі релізу, запуску оновленої реклами або контенту.

Що має бути на продакшн-сервері

Елемент	Значення для запуску сайту
SSL-сертифікат	Захищене з'єднання HTTPS (необхідно для SEO і довіри)
Analytics	Встановлений Google Analytics / GA4 для аналізу поведінки
Вебмайстер-інструменти	Підключення до Google Search Console (перевірка індексації)
Резервне копіювання (Backup)	Автоматичне щоденне/тижневе копіювання даних
Моніторинг працездатності	Повідомлення, якщо сайт «впав»

Розгортання — це лише початок життєвого циклу. Успішний додаток потребує постійного аналізу, підтримки та вдосконалення.

Основні аспекти підтримки:

🧰 Завдання	🔍 Приклади і дії
Оновлення функцій	Додати новий блок, форму, фільтр
Виправлення помилок	Зламалась форма, не працює мобільна версія
Безпекові оновлення	Підтримка актуальних бібліотек, SSL, захист від атак
UX-оптимізація	На основі фідбеку змінити розташування кнопки або CTA
Підтримка мобільності	Адаптація під нові пристрої або розширення
Аналітика та оптимізація	Виявлення відмов, аналіз поведінки, A/B тести

Дії які потрібно виконати:

- фіксуємо зворотний зв'язок користувачів**;**
- аналізуємо аналітику поведінки (Google Analytics, Hotjar);
- створюємо або погоджуємо запити на доопрацювання**;**
- ініціюємо A/B тестування для підвищення конверсії;
- беремо участь у реліз-плануванні спільно з IT-командою.

Підтримка додатка — це постійна співпраця всіх учасників команди**:** розробників, дизайнерів, контент-менеджерів, аналітиків і, безумовно, маркетологів.

Саме маркетолог стає "голосом користувача" в команді й виконує роль зв'язкового між бізнесом і технологіями**.**

Проєктування є вихідною точкою, де маркетолог бере участь у формуванні логіки взаємодії користувача з додатком, розробці користувацьких сценаріїв (user

flow) та визначенні ключових точок конверсії. Вдале проєктування — це гарантія ефективного подальшого UX.

Тестування — невіддільна частина якості продукту. Студенти ознайомились із базовими видами тестування, серед яких функціональне, юзабіліті, безпекове, мобільне, кросбраузерне та регресійне. Розглянуто автоматизовані інструменти (Selenium, Cypress, Lighthouse), з якими має бути знайомий навіть не технічний спеціаліст.

Фінальним етапом є розгортання додатку на продакшн-сервери, що супроводжується фінальним тестуванням, активацією аналітики, комунікацією з користувачами. Надалі продукт вимагає постійної підтримки: оновлень, моніторингу, SEO-оптимізації, UX-аудиту та інтеграції зворотного зв'язку від реальних користувачів.

Роль маркетолога або менеджера у фазі підтримки та розвитку продукту після розгортання.

Напрям	Дії маркетолога або менеджера
Моніторинг аналітики	Перегляд метрик: показник відмов, середній час на сторінці, конверсії. Виявлення «вузьких місць».
Робота зі зворотним зв'язком	Обробка відгуків з форми на сайті, email, соціальних мереж. Створення бази запитів користувачів.
Планування оновлень	Визначення функціоналу, якого бракує. Узагальнення запитів від користувачів. Формування ТЗ.
A/B тестування	Запуск двох версій однієї сторінки/блоку. Вимірювання, яка дає кращі результати.
Контент-менеджмент	Оновлення текстів, банерів, описів товарів. Додавання новин, блогів.
SEO-оптимізація	Підтримка актуальності ключових слів, мета-тегів, структури URL. Аудит позицій у пошуку.
UX-аналіз	Аналіз навігації, мікроанімацій, швидкості реакцій. Пошук бар'єрів для користувача.
Координація релізів	Участь у плануванні нових версій, підготовка маркетингових матеріалів до оновлення.
Безпека й довіра	Контроль наявності SSL-сертифіката, політики конфіденційності, безпечної форми збору даних.
Підтримка документації	Актуалізація інструкцій, довідкових матеріалів, FAQ для користувачів.
Маркетингова автоматизація	Інтеграція email-розсилок, чат-ботів, ретаргетингу після певних дій у додатку.

Контрольні питання:

1. Що таке веб-програмування? Які його ключові сфери застосування в маркетингу?
2. Чим відрізняється клієнтська частина від серверної у веб-розробці?
3. Назвіть основні ролі у процесі створення вебдодатків.
4. Які інструменти використовуються для створення сучасних вебсайтів?
5. Чому маркетологу важливо розуміти базові принципи веб-програмування?
6. Що таке HTML і яку роль він виконує у створенні вебсторінки?
7. Назвіть основні структурні елементи HTML-документа.
8. Як за допомогою CSS змінити колір і розмір тексту?
9. Що таке селектор у CSS? Наведіть приклад.
10. Як створити гіперпосилання у HTML?
11. Який тег відповідає за заголовки? Яка їхня ієрархія?
12. Чому важлива семантична розмітка для SEO?
13. Що таке JavaScript і чому він важливий для динаміки сайту?
14. Які типи змінних існують у JavaScript?
15. Що таке DOM і як JavaScript може з ним взаємодіяти?
16. Наведіть приклад події у JavaScript (event).
17. Як JavaScript може реагувати на дії користувача (натискання, введення)?
18. Що таке функція у JavaScript і як її оголосити?
19. Які існують популярні конструктори сайтів? Назвіть українські приклади.
20. Що таке CMS і чим вона відрізняється від конструктора?
21. Які переваги використання CMS для малого бізнесу?
22. Що входить у поняття «хостинг» та «домен»?
23. Які типи контенту можна розміщувати через CMS?
24. Чому маркетологу важливо вміти самостійно оновлювати контент у CMS?
25. Назвіть 5 базових принципів ефективного веб-дизайну.
26. Чим UX відрізняється від UI?
27. Які елементи покращують користувацький досвід на сайті?
28. Що таке адаптивний дизайн і чому він критичний у 2025 році?
29. Які інструменти використовують для створення прототипів (макетів)?
30. Що таке CTA і як він впливає на поведінку користувача?
31. Що таке SEO і яку мету воно переслідує?
32. Назвіть 3 ключові фактори, які впливають на ранжування сайту.
33. Яка різниця між On-page та Off-page SEO?
34. Як оптимізувати контент для кращої індексації в Google?
35. Яку роль відіграє швидкість завантаження сторінки в SEO?
36. Чим органічний трафік відрізняється від платного?
37. З чого починається проєктування веб-додатку?
38. Що таке юзабіліті-тестування? Як воно проводиться?
39. Які інструменти застосовуються для автоматизованого тестування?
40. Назвіть типові помилки, що виникають під час UX-тестування.
41. Як маркетолог може впливати на оновлення вебдодатку?
42. Що таке staging-сервер і для чого він потрібен?

РЕКОМЕНДОВАНІ ДЖЕРЕЛА ІНФОРМАЦІЇ

1. Хома І. Б. Основи веб-дизайну та програмування: навч. посібник. – Львів: ЛНУ ім. І. Франка, 2022.
2. Семенюк І. І. Основи веб-програмування. HTML, CSS, JavaScript. – К.: Ліра-К, 2021.
3. ІТ-посібник «Маркетинг і цифрові технології» / під ред. О. Шестакової – Харків: УІМ, 2023.
4. Браян Гоган. HTML & CSS: Design and Build Websites. – Wiley, 2020.
5. Jon Duckett. JavaScript and jQuery: Interactive Front-End Web Development. – Wiley, 2021.
6. Кирилюк О. П. Цифровий маркетинг у вебсередовищі. – НУ «Львівська політехніка», 2022.
7. Котлер Філіп, Гермаван Катарджая, Їван Сетьяван. Маркетинг 4.0. Від традиційного до цифрового. Київ : КМ-БУКС, 2019. 224 с.

Онлайн-курси та платформи

8. Prometheus (UA): Основи веб-розробки, Цифровий маркетинг, HTML для початківців.
9. Coursera (EN/UA): Web Design for Everybody, Meta Front-End Developer Professional.
10. EdEra (UA): Вступ до програмування та HTML/CSS.
11. Udemy (EN): The Complete JavaScript Course, WordPress for Beginners.
12. Codecademy (EN): Інтерактивне навчання HTML, CSS, JavaScript, SEO.
13. GoITeens / GoIT UA: Курси для початківців із акцентом на UI/UX.

Документація та стандарти

14. MDN Web Docs (Mozilla) — офіційна документація HTML, CSS, JS
15. Google Search Central — ресурс для SEO-фахівців і маркетологів
16. W3C — стандарти веб-технологій
17. Schema.org — мікророзмітка для пошукових систем
18. Google Lighthouse — вимірювання швидкості, доступності, SEO

Українські освітні ресурси

19. Освіта Дія — безкоштовні мінікурси з ІТ та цифрової грамотності
20. ITVDN Україна — відеокурси з основ веб-програмування
21. [YouTube канал "Фрілансер по життю"] — україномовні практичні уроки HTML, CSS, JavaScript

Практичні інструменти та редактори

22. Figma.com — інструмент для UX/UI прототипів
23. Visual Studio Code — безкоштовний редактор коду
24. Netlify — безкоштовне хостування вебсайтів
25. Glitch.com — інтерактивне середовище для експериментів із кодом
26. PageSpeed Insights — перевірка продуктивності сайту

Web-програмування. Конспект лекцій для здобувачів першого (бакалаврського) рівня вищої освіти галузі знань D Бізнес, адміністрування та право спеціальності D5 Маркетинг ОП Цифровий маркетинг денної та заочної форм навчання / уклад. О.М. Клімович Луцьк: Луцький НТУ, 2025. 56 с.

Комп’ютерний набір О. М. Клімович

Редактор О. М. Клімович

Підп. до друку 2025 р.

Формат 60х84/16. Папір офс. Гарнітура Таймс.

Ум. друк. арк. 1,74. Обл.-вид. арк. 2,3.

Тираж 50 прим. Зам__.

Відділ іміджу та промоції

Луцького національного технічного університету

43018 м. Луцьк, вул. Львівська, 75

Друк – Відділ іміджу та промоції ЛНТУ