

Міністерство освіти і науки України
Тернопільський національний педагогічний університет
імені Володимира Гнатюка
Інженерно-педагогічний факультет
Кафедра комп'ютерних технологій

Кваліфікаційна робота
МЕТОДИКА НАВЧАННЯ СТУДЕНТІВ СТВОРЕННЮ САЙТІВ ПРИ
ВИВЧЕННІ ІНФОРМАТИКИ В КОЛЕДЖІ

Спеціальність 015 Професійна освіта
Спеціалізація 015.39 Цифрові технології
Освітньо-професійна програма
«Професійна освіта (Комп'ютерні технології)»

ВИКОНАЛА: здобувачка вищої освіти
освітнього рівня «магістр»
ОЛІЙНИК Діана Володимирівна

НАУКОВИЙ КЕРІВНИК:
кандидат педагогічних наук, доцент
ЯЩИК Олександр Богданович

РЕЦЕНЗЕНТ:
кандидат технічних наук, доцент
ТНПУ ім. В. Гнатюка
БУРЕГА Назар Васильович

Робота захищена з оцінкою:
Національна шкала _____
Кількість балів: ____ Оцінка: ECTS ____

Міністерство освіти і науки України
Тернопільський національний педагогічний університет
імені Володимира Гнатюка
Інженерно-педагогічний факультет
Кафедра комп'ютерних технологій

ЗАВДАННЯ
ДЛЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТЦІ

Олійник Діані Володимирівні

на тему

«Методика навчання студентів
створенню сайтів при вивченні інформатики в коледжі»

Спеціальність 015 Професійна освіта

Спеціалізація 015.39 Цифрові технології

Освітньо-професійна програма: Професійна освіта (Комп'ютерні технології)

НАУКОВИЙ КЕРІВНИК: канд. пед. наук, доцент Ящик Олександр Богданович

Термін подання студентом на кафедру роботи і супроводжувальних документів:
до 5 грудня 2025 року

Зміст (перелік основних питань, які потрібно розкрити):

1. Проаналізувати теоретичні основи навчання студентів створенню сайтів у коледжах.
2. Розробити методику навчання студентів створенню сайтів в коледжах під час вивчення інформатики.
3. Експериментально перевірити ефективність запропонованої методики в освітньому процесі.

Перелік додаткових матеріалізованих результатів роботи: анкета для оцінювання ефективності навчання студентів, макети сторінок вебсайту.

ГРАФІК ПІДГОТОВКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

№ з/п	ПЕРЕЛІК РОБІТ	Термін виконання, I рік навчання	Термін виконання, II рік навчання	Відмітка наукового керівника
1	Вибір теми, затвердження її на засіданні кафедри, закріплення наукового керівника	Жовтень-листопад 2024		
2	Складання плану роботи і графіку її підготовки, узгодження з науковим керівником	Листопад 2024		
3	Вивчення літературних і електронних джерел, збір та узагальнення фактів, даних	Жовтень-січень 2024-2025		
4	Розробка методики дослідження. Проведення пошукового дослідження	Грудень-березень 2024-2025		
6	Написання розділу 1, подання його для перевірки керівнику	Травень 2025		
5	Проміжний контроль виконання роботи	Травень 2025		
6	Написання розділів 2–3, подання для перевірки керівнику		Жовтень 2025	
7	Завершення написання роботи, оформлення її згідно з вимогами, подання науковому керівнику		Листопад 2025	
8	Подання роботи на зовнішнє рецензування		Листопад 2025	
9	Підготовка роботи до захисту на засіданні кафедри: написання доповіді та подання виготовлених дидактичних матеріалів		Жовтень-листопад 2025	
10	Попередній захист роботи на засіданні кафедри		Листопад 2025	
11	Подання кваліфікаційної роботи та супроводжувальних документів		Початок грудня 2025	
12	Захист роботи на засіданні Екзаменаційної комісії		Грудень 2025	

Графік узгоджено: 24 жовтня 2025 р.

Науковий керівник _____ Ящик О. Б.

(підпис)

Виконавець кваліфікаційної роботи _____ Олійник Д. В.

(підпис)

АНОТАЦІЯ

Олійник Д. В. Методика навчання студентів створенню сайтів при вивченні інформатики в коледжі: кваліфікаційна робота на здобуття освітнього ступеня «магістр» зі спеціальності 015 Професійна освіта, спеціалізація 015.39 Цифрові технології. Тернопільський національний педагогічний університет імені Володимира Гнатюка. Тернопіль, 2025. 77 с.

У кваліфікаційній роботі досліджено використання HTML, CSS і редактора Visual Studio Code у підготовці студентів фахових коледжів. Створено навчальний вебсайт, на основі якого розроблено лекційні та лабораторні заняття. Проведено апробацію матеріалів практико-орієнтованого підходу до навчання вебтехнологій під час вивчення інформатики в коледжі

Ключові слова: HTML, CSS, Visual Studio Code, вебсайт, вебпрограмування, лабораторна робота, педагогічний експеримент.

ABSTRACT

Oliynyk D.V. Methodology of teaching students to create websites when studying computer science in college: qualification thesis for obtaining the educational degree of “Master” in the speciality 015 Vocational Education, specialisation 015.39 Digital Technologies. Volodymyr Hnatiuk Ternopil National Pedagogical University. Ternopil, 2025. p. 77 p.

The qualification work investigated the use of HTML, CSS and the Visual Studio Code editor in the training of students of professional colleges. An educational website was created, on the basis of which lecture and laboratory classes were developed. Approbation of the materials confirmed the effectiveness of a practice-oriented approach to teaching web technologies.

Keywords: HTML, CSS, Visual Studio Code, website, web programming, laboratory work, pedagogical experiment.

ЗМІСТ

ВСТУП	7
РОЗДІЛ I. ТЕОРЕТИЧНІ ОСНОВИ НАВЧАННЯ СТУДЕНТІВ	
СТВОРЕННЮ САЙТІВ.....	9
1.1 Особливості навчання студентів створенню вебсайтів в процесі вивчення інформатики в коледжах.....	9
1.2 Аналіз інструментів програмного забезпечення для розробки сайтів....	15
1.3 Характеристика основних засобів створення та стилізації вебсайтів.....	18
Висновок до першого розділу	29
РОЗДІЛ II. МЕТОДИКА НАВЧАННЯ СТУДЕНТІВ КОЛЕДЖІВ	
СТВОРЕННЮ САЙТІВ.....	30
2.1 Аналіз змісту підготовки студентів коледжів до розробки сайтів	30
2.2 Методика навчання студентів коледжів створенню вебсайтів у середовищі VS Code.....	36
2.3 Експериментальне дослідження ефективності запропонованої методики.....	57
Висновок до другого розділу	62
ЗАГАЛЬНІ ВИСНОВКИ	63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	64
ДОДАТКИ.....	66

ВСТУП

Сучасний світ неможливо уявити без використання інформаційних технологій та глобальних мереж передачі даних. Інтернет став надзвичайно важливим інструментом комунікації, який дозволяє людям обмінюватися інформацією з будь-якої точки світу [5].

Актуальність теми кваліфікаційної роботи зумовлена тим, що вебресурси формують основу сучасного цифрового простору [10]. Володіння навичками створення вебсторінок за допомогою HTML і CSS, а також уміння працювати з сучасними редакторами коду є необхідною складовою професійної підготовки фахівців у галузі інформаційних технологій. Особливо важливо забезпечити ефективне засвоєння цих навичок студентами фахових коледжів, які готуються до практичної діяльності в IT-сфері.

У межах кваліфікаційної роботи було створено навчальний вебсайт, що включає семантичну HTML-структуру, стилізацію елементів, оптимізацію графіки та реалізацію форми зворотного зв'язку. На його основі розроблено лекційне заняття та лабораторну роботу, які спрямовані на формування практичних умінь студентів. Апробація матеріалів у межах педагогічного експерименту підтвердила ефективність практико-орієнтованого підходу до навчання вебтехнологій.

Об'єкт дослідження: процес навчання студентів коледжів створенню сайтів при вивченні інформатики.

Предмет дослідження: методика навчання студентів коледжів створенню сайтів на прикладі середовища Visual Studio Code.

Мета: розробити методику навчання студентів створенню сайтів у процесі вивчення інформатики в коледжі та перевірити її ефективність.

Завдання кваліфікаційної роботи:

1. Проаналізувати теоретичні основи навчання студентів створенню сайтів у коледжах.

2. Розробити методику навчання студентів створенню сайтів в коледжах під час вивчення інформатики.

3. Експериментально перевірити ефективність запропонованої методики в освітньому процесі.

Методи дослідження: теоретичні (аналіз, узагальнення літературних джерел і проєктування методичного забезпечення); емпіричні (педагогічне спостереження, анкетування, опитування, педагогічний експеримент).

Наукова новизна полягає в удосконаленні методики підготовки студентів коледжів до розробки вебсайтів під час вивчення інформатики із застосуванням інтерактивних методів (метод проєктів) навчання в освітньому процесі.

Практичне значення полягає в розробці методики навчання студентів коледжів створенню сайтів із застосуванням середовища Visual Studio Code, що передбачає поетапне вивчення процесу розробки вебсайту з врахуванням стилізації ресурсу.

Апробація результатів дослідження. Участь у V Міжнародній конференції «Моделі міждисциплінарних та міжгалузевих освітніх та освітньо-наукових програм в умовах воєнного стану: виклики та варіанти впровадження» (м. Одеса, 10–12 жовтня 2025 року.). За результатами конференції опубліковано тези на тему: «Особливості навчання студентів створенню вебсайтів в процесі вивчення інформатики в коледжах» [Додаток В].

РОЗДІЛ І. ТЕОРЕТИЧНІ ОСНОВИ НАВЧАННЯ СТУДЕНТІВ СТВОРЕННЮ САЙТІВ

1.1 Особливості навчання студентів створенню вебсайтів в процесі вивчення інформатики в коледжах

Навчання інформатики в коледжах часто не відповідає тим вимогам, які ставить сьогоdnішній ринок праці. Однією з найбільш помітних проблем є те, що студентів майже не вчать створювати вебсайти або роблять це дуже поверхнево. Хоча веброзробка є однією з найпопулярніших сфер ІТ, а навички створення сайтів потрібні в багатьох професіях, у програмі більшості коледжів ця тема займає зовсім небагато місця. Зазвичай вебтехнології вивчаються у межах кількох занять, де студенти знайомляться лише з основними HTML-тегами та базовими правилами оформлення тексту на сторінці. Такого мінімального обсягу знань недостатньо, щоб навчитися робити повноцінний сучасний сайт.

Додатковою проблемою є те, що курс інформатики часто орієнтований переважно на теорію. Студенти багато вчать про історію розвитку комп'ютерної техніки, принципи роботи операційних систем, правила безпеки в мережі та інші загальні відомості. Хоча ці знання важливі, їх занадто багато у порівнянні з тим, скільки часу приділяється практичним навичкам. У результаті студенти можуть знати, що таке HTML або вебсайт, але не розуміють, як з нуля створити власну сторінку, як працює верстка або як сайт виглядатиме на екранах різних пристроїв.

Практична частина уроків теж має свої недоліки. Часто студенти виконують однакові завдання за інструкцією, де потрібно просто повторити вже готовий приклад. Це не дає можливості проявити творчість, спробувати різні варіанти оформлення або навчитися шукати рішення самостійно. Таке навчання не формує справжніх навичок веброзробки, адже для цього потрібні проекти, де студент сам обирає тему, стиль, структуру сайту і вчиться виправляти власні помилки.

Не менш важливим є технічне забезпечення. У деяких коледжах комп'ютери застарілі, працюють повільно або не підтримують сучасні інструменти, якими користуються веброзробники. Студенти часто не знайомі з такими програмами і сервісами, як Visual Studio Code, GitHub, Figma або системи макетування. Через це вони не знають, як працюють справжні інструменти, з якими стикаються розробники в реальних компаніях.

Ще одна проблема – відсутність зв'язку з реальними проєктами та представниками ІТ-сфери. У коледжах рідко пропонують студентам створити сайт для справжнього замовника або попрацювати над командним проєктом. Через це студенти не бачать, як їхні знання можуть застосовуватися на практиці, і не отримують досвіду, який дуже цінується роботодавцями.

Такі недоліки негативно впливають на мотивацію студентів. Коли навчання здається абстрактним і не пов'язане з реальним життям, інтерес знижується. Багато студентів не усвідомлюють, що веброзробка може стати їхнім майбутнім фахом, або що ця навичка може допомогти у будь-якій сфері – від маркетингу до підприємництва. Через це вони не прагнуть розвиватися у цьому напрямі самостійно, а отримані на заняттях знання залишаються поверховими.

Навчання студентів створенню вебсайтів у коледжах у межах дисципліни «Інформатика» має прикладний і професійно орієнтований характер, оскільки веброзробка є однією з найбільш затребуваних цифрових компетентностей сучасного фахівця. Особливістю такого навчання є поєднання теоретичних основ із практичною діяльністю, що забезпечує формування в студентів умінь і навичок реального проєктування та реалізації вебресурсів.

Важливою особливістю є поетапність навчання: від засвоєння базових понять вебтехнологій (структура вебсторінки, принципи роботи мережі Інтернет, HTML-розмітка) до опанування каскадних таблиць стилів (CSS), основ клієнтського програмування (JavaScript) та елементів адаптивного дизайну. Такий підхід сприяє логічному та усвідомленому формуванню знань, запобігає фрагментарності у сприйнятті навчального матеріалу.

Навчання створенню вебсайтів у коледжах ґрунтується на практико-орієнтованому підході. Значна частина навчального часу відводиться виконанню лабораторних і практичних робіт, у ході яких студенти створюють власні вебсторінки, сайти-візитівки, інформаційні або навчальні ресурси. Це дозволяє не лише закріпити теоретичні знання, а й розвивати алгоритмічне мислення, креативність, уважність до деталей та навички самостійної роботи.

Суттєвою особливістю є використання проєктних та інтерактивних методів навчання. Робота над індивідуальними або груповими вебпроєктами сприяє формуванню командної взаємодії, відповідальності за результат, уміння планувати етапи розробки та презентувати готовий продукт. Водночас застосування кейсів, проблемних завдань і аналізу реальних вебресурсів допомагає студентам краще зрозуміти вимоги сучасного цифрового середовища.

Окрему увагу в процесі навчання приділяють формуванню інформаційної та медіаграмотності студентів. Під час створення вебсайтів вони навчаються добирати та критично оцінювати інформацію, дотримуватися авторського права, правил безпеки в Інтернеті та етичних норм цифрової комунікації. Це є важливою складовою підготовки майбутніх фахівців у різних галузях.

Ще однією особливістю є адаптація змісту навчання до рівня підготовки студентів і специфіки спеціальностей коледжу. Викладач має враховувати різний стартовий рівень цифрових навичок, поєднуючи індивідуалізацію навчання з диференційованими завданнями та використанням сучасних онлайн-інструментів і платформ.

Отже, навчання студентів створенню вебсайтів у процесі вивчення інформатики в коледжах характеризується практичною спрямованістю, поетапністю, інтеграцією сучасних вебтехнологій і активних методів навчання, що в цілому сприяє формуванню професійних і цифрових компетентностей майбутніх фахівців.

Практико-орієнтований підхід є провідним у навчанні студентів створенню вебсайтів у процесі вивчення інформатики в коледжах, оскільки він спрямований на формування не лише теоретичних знань, а насамперед професійних умінь і

навичок, необхідних для реальної діяльності в цифровому середовищі. Застосування цього підходу забезпечує тісний зв'язок навчального матеріалу з практичними завданнями та майбутньою професійною діяльністю студентів.

Сутність практико-орієнтованого підходу полягає в організації навчального процесу через виконання практичних, лабораторних і проєктних робіт, у ході яких студенти безпосередньо створюють вебсторінки та вебсайти. Починаючи з простих завдань (розмітка тексту, додавання зображень і гіперпосилань), студенти поступово переходять до складніших видів діяльності: оформлення дизайну за допомогою CSS, створення навігаційної структури сайту, забезпечення адаптивності для різних пристроїв. Така послідовність сприяє закріпленню знань через практичний досвід.

Важливою складовою практико-орієнтованого навчання є виконання навчальних вебпроєктів. Проєктна діяльність дозволяє моделювати реальні умови роботи веброзробника: аналіз потреб користувачів, планування структури сайту, розподіл завдань, тестування та презентація результатів. У процесі роботи над проєктом студенти навчаються застосовувати знання комплексно, приймати обґрунтовані рішення та відповідати за кінцевий продукт.

Практико-орієнтований підхід також передбачає використання реальних прикладів і кейсів. Аналіз уже існуючих вебсайтів, виявлення їхніх переваг і недоліків, порівняння різних дизайнерських і технічних рішень сприяють розвитку критичного мислення та формуванню професійного бачення. Студенти не лише копіюють готові зразки, а й навчаються аргументувати власний вибір технологій і способів реалізації.

Окрему увагу в межах практико-орієнтованого підходу приділяють формуванню навичок самостійної роботи та самоосвіти. Оскільки вебтехнології швидко змінюються, студенти вчаться користуватися документацією, онлайн-довідниками, навчальними платформами, що формує готовність до безперервного професійного розвитку.

Таким чином, практико-орієнтований підхід у навчанні створенню вебсайтів у коледжах забезпечує активну позицію студентів у навчальному

процесі, сприяє формуванню професійних компетентностей, підвищує мотивацію до навчання та дозволяє максимально наблизити освітній процес до реальних умов майбутньої діяльності.

Адаптація змісту навчання створенню вебсайтів у процесі вивчення інформатики в коледжах є важливою педагогічною умовою ефективного формування цифрових і професійних компетентностей студентів. Вона передбачає врахування початкового рівня знань і навичок здобувачів освіти, їхніх індивідуальних можливостей, а також особливостей обраної спеціальності та майбутньої професійної діяльності.

Насамперед адаптація здійснюється з урахуванням різного стартового рівня підготовки студентів. У групах коледжу часто навчаються здобувачі з неоднаковим досвідом використання інформаційних технологій: від базових користувачів до студентів, які вже мають початкові навички програмування чи вебдизайну. У зв'язку з цим доцільним є застосування диференційованого підходу, що передбачає варіативність завдань за рівнем складності, темпом виконання та обсягом самостійної роботи. Це дозволяє уникнути перевантаження менш підготовлених студентів і водночас стимулює розвиток більш сильних.

Важливою формою адаптації є модульна побудова навчального курсу. Модулі можуть включати обов'язковий базовий зміст (основи HTML і CSS, принципи побудови структури вебсайту) та варіативну частину, орієнтовану на поглиблене або прикладне вивчення окремих аспектів веброзробки. Такий підхід забезпечує гнучкість навчального процесу та дає можливість студентам обирати індивідуальну освітню траєкторію.

Суттєвим аспектом адаптації є врахування специфіки спеціальностей коледжу. Зміст і характер практичних завдань зі створення вебсайтів доцільно пов'язувати з професійною спрямованістю навчання. Адаптація змісту навчання також передбачає використання різноманітних форм і засобів навчання. Застосування онлайн-редакторів коду, візуальних конструкторів сайтів, навчальних відеоматеріалів і симуляторів дозволяє врахувати індивідуальні

стилі навчання студентів і сприяє більш доступному засвоєнню складних понять. Викладач має можливість поєднувати традиційні методи з цифровими інструментами, забезпечуючи підтримку та зворотний зв'язок.

Тому, адаптація змісту навчання створенню вебсайтів до рівня підготовки студентів і специфіки спеціальностей коледжу забезпечує індивідуалізацію освітнього процесу, підвищує ефективність засвоєння знань і сприяє формуванню професійно значущих компетентностей. Такий підхід дозволяє максимально наблизити навчання до реальних потреб студентів і вимог сучасного ринку праці.

Отже, навчання студентів створенню вебсайтів у процесі вивчення інформатики в коледжах є важливою складовою формування сучасних цифрових і професійних компетентностей. Ефективність цього процесу забезпечується поєднанням практико-орієнтованого підходу, активних та інтерактивних методів навчання, а також адаптацією змісту навчання до рівня підготовки студентів і специфіки їхніх спеціальностей. Така організація освітнього процесу сприяє усвідомленому засвоєнню знань, розвитку творчого й алгоритмічного мислення, умінню застосовувати вебтехнології для розв'язання реальних професійних завдань.

Застосування диференційованих і проєктних форм роботи, використання сучасних цифрових інструментів та орієнтація на практичний результат підвищують мотивацію студентів до навчання і формують готовність до самостійної діяльності та безперервного професійного розвитку. У підсумку навчання створенню вебсайтів у коледжах стає ефективним засобом підготовки конкурентоспроможних фахівців, здатних успішно діяти в умовах цифровізації суспільства та вимог сучасного ринку праці.

1.2 Аналіз інструментів програмного забезпечення для розробки сайтів

Одним з найбільш відомих інструментів для розробки вебсайтів мовою HTML є **Adobe Dreamweaver**. Це потужний редактор вебсторінок, який дозволяє створювати та редагувати вебсайти без необхідності написання власноруч HTML-коду.

Основна перевага Dreamweaver полягає у тому, що він надає можливість розробникам вебсайтів використовувати візуальний режим редагування, що дозволяє швидко та ефективно створювати вебсторінки без необхідності вчитися мови HTML. Завдяки код-редактору, Dreamweaver дозволяє швидко відкривати та редагувати HTML, CSS та JavaScript файли. Програма надає можливість переглядати вебсторінки в різних режимах: кодування, дизайну.

Adobe Dreamweaver є платним програмним забезпеченням, проте на офіційному сайті Adobe можна завантажити пробну версію програми для тестування перед придбанням.

Sublime Text – це текстовий редактор для розробників, який підтримує багато мов програмування, включаючи HTML, CSS, JavaScript, PHP, Python тощо. Ця програма має широкий спектр можливостей та дозволяє працювати з багатьма файлами одночасно.

Деякі з основних функцій редактора: автоматично підкреслює код залежно від його типу; автоматично доповнює код на основі контексту; можна працювати з багатьма файлами одночасно, кожен у своїй вкладці; має широкий спектр плагінів, що дозволяють розширювати його функціональність; підтримує багато мов, включаючи англійську, українську тощо; дозволяє використовувати швидкі клавіші для виконання різних задач.

Sublime Text є потужним інструментом для розробників, який дозволяє працювати зі зручним інтерфейсом та широким набором функцій.

Notepad++ є безкоштовним текстовим редактором, призначеним для редагування коду. Цей редактор має підсвічування синтаксису для більшості мов

програмування, що дозволяє більш ефективно працювати з кодом.

Має декілька корисних функцій, включаючи пошук та заміну тексту, автоматичне завершення коду, можливість створювати макроси, зберігати та відновлювати сеанси, використовувати вкладки для редагування кількох файлів та багато інших. До інших функцій Notepad++ входить можливість використовувати додаткові плагіни для розширення можливостей програми.

Підтримує роботу з багатьма мовами програмування, включаючи HTML, CSS, JavaScript, PHP, Python та інші. Доступний для встановлення на операційні системи Windows та Linux.

Atom – програмне забезпечення для редагування тексту та розробки програмного коду, розроблене командою GitHub. Atom працює на багатьох операційних системах, включаючи Windows, macOS та Linux.

Atom має багато корисних функцій, які роблять його популярним серед розробників. Деякі з них:

- розуміє багато мов програмування та надає підсвічування синтаксису, що допомагає зробити код більш читабельним та зрозумілим;
- має вбудований менеджер пакетів, який дозволяє встановлювати різноманітні плагіни для роботи з різними мовами програмування та різними інструментами;
- має вбудовану консоль, яка дозволяє виконувати команди, що полегшує роботу з файлами та іншими операціями;
- дозволяє відкривати кілька файлів одночасно та працювати з ними в окремих вкладках, що зручно при роботі з великими проєктами.

Bootstrap – безкоштовний фреймворк для розробки вебсайтів, створений для швидкої та простої розробки адаптивних та мобільних вебінтерфейсів. Він містить набір HTML, CSS та JavaScript компонентів та плагінів, що значно полегшують розробку сайтів.

Був розроблений командою Twitter і вперше був випущений у 2011 році. Він використовує систему сітки, що дозволяє розміщувати елементи вебсторінки відповідно до розміру екрану. Крім того, Bootstrap містить шаблони дизайну, які

можуть бути використані для створення стильних та сучасних вебсайтів.

Основні переваги:

- забезпечує адаптивний дизайн, що означає, що вебсайт буде підлаштовуватися під розміри екранів на різних пристроях, таких як настільні комп'ютери, планшети та мобільні телефони;
- підтримує основні веббраузери, включаючи Google Chrome, Mozilla Firefox, Safari, Opera та Internet Explorer;
- містить багато готових компонентів та шаблонів, що дозволяє розробникам швидко створювати професійні вебсайти без необхідності в написанні власного коду;
- містить багато готових плагінів та скриптів JavaScript, які можуть бути використані для розширення функціональності вашого вебсайту.

Visual Studio Code (VS Code) – безкоштовний редактор коду, розроблений компанією Microsoft для платформ Windows, Linux і macOS. Він включає в себе багато функцій, які допомагають програмістам створювати високоякісний код, збільшуючи його продуктивність [8].

VS Code має дуже широкі можливості з розширення, що дозволяє розробникам створювати свої власні розширення, або використовувати тисячі доступних для VS Code.

Деякі з основних функцій VS Code включають підсвічування синтаксису, автодоповнення коду, налаштування відступів, підтримку багатьох мов програмування, таких як HTML, CSS, JavaScript, Python, PHP, Java та інших. Також має можливість віддаленої розробки, що дозволяє розробникам підключатися до серверів, щоб редагувати та компілювати свій код на відстані.

Крім того, VS Code має вбудовану підтримку для популярних фреймворків, таких як React, Angular, Vue і Node.js, що дозволяє розробникам швидко розгортати та налагоджувати свої проєкти.

*У навчальному процесі, зокрема у фахових коледжах, доцільним є вивчення **Visual Studio Code**, адже цей редактор поєднує простоту використання з професійними можливостями та відповідає сучасним вимогам індустрії. Він*

безкоштовний і має багато розширень, що дозволяє студентам адаптувати середовище під різні освітні та практичні потреби. Робота з VS Code сприяє формуванню навичок програмування різними мовами, опануванню популярних фреймворків та систем контролю версій, а також навчає працювати у середовищі, яке широко використовується професійними розробниками. Це робить його ефективним інструментом для вивчення веброзробки та підготовки конкурентоспроможних фахівців, здатних швидко інтегруватися у сучасні команди розробників.

1.3 Характеристика основних засобів створення та стилізації вебсайтів

HTML – це мова, яку використовують для розмітки вебдокументів. Тобто вона дозволяє створювати вебсторінку, визначаючи роль кожного елемента [4].

HTML – документ – це файл, що складається з тегів розмітки, які обробляє браузер і в результаті відображаються користувачу, як вебсторінка.

Тег є елементом HTML, що обгортається в кутові дужки [6].

Виділяють **парні теги**, тобто відкриваючі і закриваючі, які обгортають наповнення:

```
<h1>Наповнення</h1>
```

У закриваючих використовують символ “/”.

Всередині парних тегів можна вкладати багато інших – це називається **вкладенням**. У тезі <head> вкладено теги <meta>, які визначають метадані – властивості документа і <title>, що містить ім’я документа. Також у тег <body> включено теги <h1>, <p>, , , які задають заголовок, підзаголовок, неупорядкований список і елемент списку відповідно (Рис. 1.1).

```

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>Заголовок</h1>
  <p>Підзаголовок</p>
  <ul>
    <li>Перший пункт списку</li>
    <li>Другий пункт списку</li>
    <li>Третій пункт списку</li>
  </ul>
</body>

```

Рис. 1.1. – Вкладення тегів

Одинарні теги, тобто тільки відкриваючі, не містять у собі тексту. Їх використовують лише для визначення властивостей і підключення різних файлів документа.

Прикладом є тег `<img>`, який відповідає за відображення зображень на вебсторінці. У ньому є атрибути: `src`, `alt`, `width`, `height`, що слугують для зміни властивостей зображення:

```

```

У кінці одинарних тегів також використовують символ `"/` перед кутовою дужкою.

Атрибут – це додаткове налаштування, яке завжди включають до початкового тегу, а значення записують після знаку дорівнює та у лапках. Якщо атрибутів декілька, то їх розділяють пробілом [3].

Розрізняють обов’язкові і необов’язкові атрибути. Наприклад, у тезі `<a>` обов’язковим атрибутом є `href`. У ньому міститься посилання, яке відобразиться користувачу після того, як він натисне на текст у тезі `<a>` – “Studio”:

```
<a href="./form.html">Studio</a>
```

Наступним атрибутом є `src`, що також вважається обов'язковим і вкладається в тег `<img>`. У цей атрибут прописується адреса розміщення зображення:

```

```

Тег `<img>` ще містить атрибути `width` і `height`, які відповідають за ширину і висоту зображення відповідно. Значення прописуються числом у лапках, що означає кількість пікселів:

```
width="370" height="294"
```

В тезі `<img>` також прописують атрибут `alt` – альтернативний текст, який буде відображатися якщо зображення не завантажиться на сторінці:

```

```

Атрибут `style` – вказує на стиль елемента, призначений для вказівки кольору тексту, розміру або шрифту:

```
<p style="color: purple">Студія</p>
```



Рис. 1.2. – Результат використання атрибута `style`

У тезі `<html>` оголошується атрибут `lang` для визначення мови документу. На вибір мови вказують перші дві літери. Наприклад, `uk` – українська:

```
<html lang="uk">
```

Атрибут `title` у тезі буде відображати підказку, якщо користувач наведе мишу на текст (Рис. 1.3).

Атрибут title

Я заголовок

Рис. 1.3. – Атрибут `title`

Всі HTML – документи складаються з **елементів** – це початковий і кінцевий теги та зміст, що міститься між ними. Елементи в яких немає змісту називаються **порожніми**. Вони мають тільки початковий тег.

Наприклад, тег `<h1>` – початковий, `</h1>` – кінцевий, а текст між ними, тобто “Студія” – це зміст елемента:

```
<h1>Студія</h1>
```

Прикладом **порожнього елемента** можна вважати елемент `<br>`, який задає розрив рядка:

```
<h1> Елементи <br> HTML – документів </h1>
```

Якщо запустити на виконання код, вказаний вище, то буде отримано заголовок, розміщений у двох рядках (Рис. 1.4).



**Елементи
HTML – документів**

Рис. 1.4. – Приклад використання порожнього тегу `<br>`

Структура HTML – документа складається із відповідного набору тегів.

Для початку, кожен HTML – документ повинен починатися з рядка:

```
<!DOCTYPE ..>
```

У ньому оголошується його тип, наприклад, `<!DOCTYPE html>` цей рядок означає, що розробник вказує на 5 версію HTML.

Так браузер зможе правильно “прочитати” версію HTML і відобразити веб сторінку.

Елемент `<html>` огортає весь документ, слугує контейнером у якому розташовується весь вміст сторінки:

```
<html>...</html>
```

Елемент `<head>` міститься між тегами `<html>` і `<body>`. Він призначений для збереження інформації про властивості сторінки, а саме заголовок документа, метадані, лінки тощо.

Тег `<meta>` також є фрагментом HTML – коду, який прописується у тезі `<head>`. Вони не відображаються на сторінці, а тільки забезпечують те, щоб браузер “зрозумів” структуру сторінки і правильно її відобразив. Таких тегів в одному документі може бути декілька.

В стандартній версії HTML першим мета-тегом є:

```
<meta charset="UTF-8">
```

Він відповідає за кодування символів, у цьому випадку воно відбувається за допомогою стандарту UTF-8.

Другий мета-тег:

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

Він вказує на версію браузера у якому буде відображатись сторінка. У цьому випадку має використовуватись остання версія браузера Internet Explorer.

Третій – це мета-тег:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

У ньому прописано, як браузер повинен змінювати розмір і масштаб сторінки при її відображенні. У цьому випадку розмір має дорівнювати ширині екрану пристрою, який використовується, а масштаб – початковий, а саме 1.0.

Стандартні мета-теги:

```
<head>
```

```
<meta charset="UTF-8" />
```

```
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

```
</head>
```

Для того, щоб відобразити заголовок документа потрібно його назву помістити всередину `<title>`, який у свою чергу розміщений у тезі `<head>`:

```
<head>
```

```
<title>Заголовок</title>
```

```
</head>
```

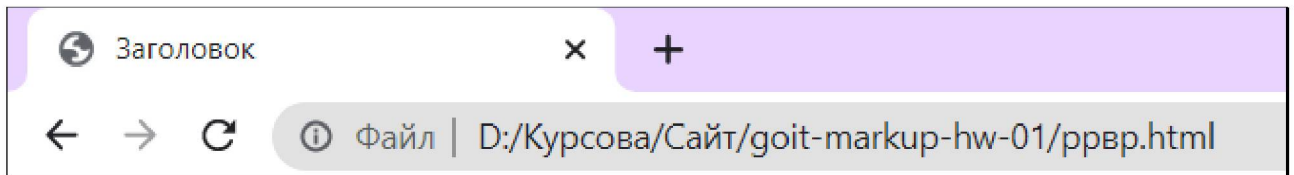


Рис. 1.5. – Вигляд заголовку у вкладці браузера

Елемент `<body>` також міститься всередині `<html>` після елемента `<head>` і використовується для відображення тіла документа, тобто того, що буде на сторінці у браузері:

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

У тезі `<body>` для форматування та створення змісту використовуються такі теги, як `<h1>` – відповідає за визначення заголовку першого рівня, `<h2>` другого, `<h3>` – третього і так далі. Тобто `<h1>` – це найважливіший заголовок на сторінці, `<h2>` – менш важливий тощо (Рис. 1.6). Таких рівнів може бути шість.

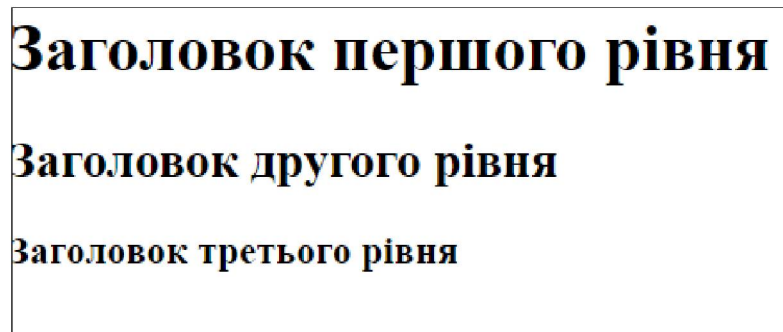


Рис. 1.6. – Заголовки першого, другого і третього рівнів

Наступний тег `<p>` – підзаголовок. Використовується для того, щоб поділити текст на параграфи чи абзаци і зробити його більш читабельним.



Рис. 1.7. – Різниця між заголовком і параграфом

Тег `<ul>` використовується для визначення нумерованого списку (Рис. 1.8). В якому кожен елемент позначається крапкою. Для цього використовують тег `<li>`, що визначає елемент списку:

```
<ul>
```

```
<li>перший пункт</li>
```

```
<li>другий пункт</li>
```

```
<li>третій пункт</li>
```

```
</ul>
```

- перший пункт
- другий пункт
- третій пункт

Рис. 1.8. – Ненумерований список

Тег `<ol>` використовується для визначення нумерованого списку. В якому кожен елемент пронумерований відповідно. Для цього теж використовують тег `<li>`, що визначає елемент списку:

```
<ol>
```

```
<li>перший пункт</li>
```

```
<li>другий пункт</li>
```

```
<li>третій пункт</li>
```

```
</ol>
```

Тег `<a>` існує для визначення посилання. Цей тег використовується для створення гіперпосилань на інші сторінки, документи або медіа-файли. Посилання можуть містити текст або зображення, які ведуть на іншу сторінку або ресурс в Інтернеті:

```
<a href="#">Текст</a>
```

“#” – знак решітки прописують у атрибуті `href`, щоб не залишати пробіл між лапками, якщо посилання ще не вказано:

```
<a href= "#">Посилання</a>
```

```

<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Назва документа</title>
</head>
<body>
  <h1>Заголовок</h1>
  <p>Підзаголовок</p>
  <p><a href="#">Посилання</a></p>
  <h2>Ненумерований список</h2>

  <ul>
    <li>перший пункт</li>
    <li>другий пункт</li>
    <li>третій пункт</li>
  </ul>

  <h2>Нумерований список</h2>

  <ol>
    <li>перший пункт</li>
    <li>другий пункт</li>
    <li>третій пункт</li>
  </ol>
</body>
</html>

```

Рис. 1.9. – Скелет HTML – документа із основними тегами

Після запуску коду HTML – документа із основними тегами, сторінка у браузері має такий вигляд (Рис. 1.10):

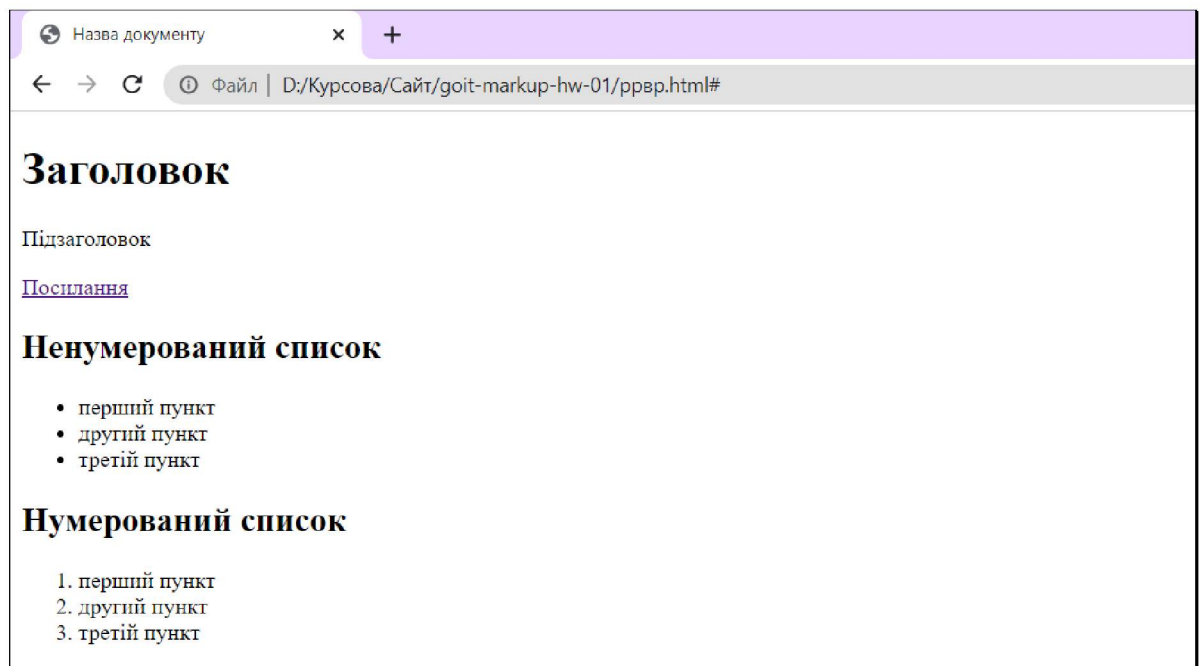


Рис. 1.10. – Вигляд сторінки у браузері

Способи підключення CSS для стилізації сайту

CSS – це мова таблиці стилів, що використовується для опису вигляду та форматування вебсторінок, написаних мовою HTML.

Синтаксис CSS складається з селекторів, властивостей та їх значень [1]. Наприклад, клас “.menu-link” – це селектор, “font-size:” – властивість, а “16px;” – значення:

```
.menu-link {
font-size: 16px;
}
```

Селектори вказують на елементи HTML, до яких будуть застосовані стилі. Вони можуть бути класами або тегами. Властивості вказують, які атрибути елементів повинні змінюватися, такі як колір, розмір, відступи тощо. Значення властивостей вказують, які конкретні параметри повинні бути застосовані до відповідної властивості.

Існує **декілька способів підключення стилів CSS** в документ HTML [7].

Наприклад, використовуючи теги <style>...</style> можна застосувати спосіб підключення **внутрішньої таблиці стилів** в документ. Їх треба помістити у тег <head> і звернутися до кожного тегу, якому потрібно надати стилі (Рис. 1.11).

Перевагою цього способу є те, що він може бути корисним для швидкого внесення невеликих змін у стиль, без створення окремого файлу. Недоліком – ускладнення підтримки стилів та те, що розмір HTML – документа збільшується і стає незручно редагувати файл. Силі вказані в коді (Рис. 1.11) будуть застосовуватись до всіх елементів <p> на сторінці.

```

<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Назва документа</title>
  <style>
    p{
      font-family:verdana;
      font-size:12px;
      line-height:20px;
      text-indent:25px;
      color:□#000000;
    }
  </style>
</head>
<body>
</body>
</html>

```

Рис. 1.11. – Підключення внутрішньої таблиці стилів

Наступним способом, найбільш розповсюдженим, є **зовнішня таблиця стилів**. Файл з розширенням .css підключають у секцію head за допомогою тегу <link>. У ньому містяться атрибути rel і href:

rel="stylesheet" – вказує на те, що підключений файл є файлом стилів;

href="style.css" – вказує на шлях до файлу стилів, який повинен бути використаний для форматування сторінки (Рис. 1.12).

```

!DOCTYPE html>
html lang="uk">
head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Назва документа</title>

  <link rel="stylesheet" href="style.css"/>
/head>
body>
|
/body>
/html>

```

Рис. 1.12. – Підключення зовнішньої таблиці стилів

Коли браузер зустрічає тег <link>, він завантажує файл стилів "style.css" та

застосовує його до HTML документу, що дозволяє задати зовнішній вигляд сторінки.

Перевагами цього способу є те, що можна відокремити стилі від контенту, забезпечуючи більш легку та ефективну роботу з ними і покращення швидкості завантаження сторінки у браузері.

Недоліки – придатний для використання тільки тоді, коли використовуються однакові стилі на кількох сторінках.

Третій спосіб підключення стилів називається **вбудованим**, тобто за допомогою атрибуту style до потрібного елементу.

Наприклад, стиль для фону background-color та кольору тексту color вказуються прямо в атрибуті style тегів <body> та <p> (Рис. 1.13).

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Назва документа</title>
</head>
<body style="background-color: lightblue;">
  <h1>Hello World!</h1>
  <p style="color: red;">This is a paragraph.</p>
</body>
</html>
```

Рис. 1.13. – Підключення вбудованого стилю

Перевагами цього способу може бути швидко доданий без необхідності редагування зовнішнього CSS файл; простий для зрозуміння, оскільки він знаходиться в тому тезі, де його використовують; придатний для задання унікальних стилів для окремих елементів, які не повторюються.

Недоліками способу є те, що він змінювати стиль тільки одного елемента, що важко підтримувати на сторінці з багатьма елементами; не дозволяє використовувати спільні стилі для кількох елементів; може призвести до дублювання коду, якщо його використовувати для багатьох елементів на сторінці.

Загалом, перший спосіб – внутрішня таблиця стилів, здійснюється через тег <style> у розділі <head>.

Другий спосіб – зовнішня таблиця стилів, підключається за допомогою тегу `<link>` і файлу з розширенням `.css`.

Третій спосіб – вбудований стиль, де атрибут `style` додається до потрібного елемента.

Серед трьох способів найкращим вибором буде використання зовнішньої таблиці стилів через те, що можна відокремити їх від контенту.

Висновок до першого розділу

У першому розділі кваліфікаційної роботи проаналізовано особливості навчання студентів створенню вебсайтів у фахових коледжах та виявлено розбіжності між методикою викладання інформатики та сучасними вимогами ринку праці. У навчанні часто приділяють недостатньо уваги практичній веброзробці. Знання студентів залишаються абстрактними та поверхневими, що призводить до втрати мотивації та відсутності необхідного досвіду для успішного старту в ІТ-сфері. З огляду на ці недоліки, виникла необхідність впровадження сучасних інструментів та підходів у навчальний процес.

Проаналізовано найпопулярніші інструменти для розробки вебсайтів. Вибір конкретного редактора залежить від потреб розробника та характеристик проєкту. В результаті аналізу, було обґрунтовано доцільність вивчення редактора VS Code у фахових коледжах, адже він має багато переваг, відповідає сучасним вимогам сфери ІТ та надає широкі можливості для підготовки студентів.

Охарактеризовано основні теги та атрибути мови HTML, що дозволяють створювати структуровані вебсторінки, визначаючи роль кожного елемента. Розглянуто основні поняття та способи підключення CSS. Структура CSS складається з селекторів, властивостей та їх значень, що дозволяє визначати вигляд, розташування та форматування вебсторінок. Виокремлено три основні способи підключення CSS: внутрішня, зовнішня таблиця стилів та вбудований стиль.

РОЗДІЛ II. МЕТОДИКА НАВЧАННЯ СТУДЕНТІВ КОЛЕДЖІВ СТВОРЕННЮ САЙТІВ

2.1 Аналіз змісту підготовки студентів коледжів до розробки сайтів

Навчальний план – нормативний документ, який визначає перелік і обсяг навчальних дисциплін (освітніх компонентів), логічну послідовність їх вивчення, вид семестрового контролю, розподіл навчальних годин за видами занять і форму атестації.

Фаховий коледж інформаційних технологій Львівської політехніки пропонує навчальний план для підготовки молодшого бакалавра за спеціальністю 122 Комп'ютерні науки. Він містить загальні відомості та складається з таких розділів: графік навчального процесу, зведені дані за бюджетом часу, практична підготовка, державна підсумкова атестація, атестація здобувачів фахової передвищої освіти, план навчального процесу, перелік навчальних кабінетів, лабораторій і майстерень, пояснення до навчального плану.

У першому розділі розміщений графік навчального процесу (Рис. 2.1) складений на весь період навчання. У ньому зазначено курси, на кожен з яких передбачено календарні терміни семестрів, види (навчальна, технологічна та переддипломна) та терміни практик, семестрового контролю, атестації здобувачів фахової передвищої освіти, у тому числі виконання кваліфікаційної роботи, канікул. Для відображення цієї інформації використано умовні позначення.

курс	1 семестр	вересень				жовтень				листопад				грудень				січень				лютий				березень				квітень				травень				червень				липень				серпень								
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	
I	17	23																	К	К	К																																	
II	17	18																	К	К	К																																	
III	15	15																	Н	Н	К	К	С	С																														
IV	17	13																	К	К	С	С																																

Теоретичні заняття

☐

Навчальна

☐

Практики

☐

Технологічна

☐

Переддипломна

☐

Семестровий контроль

☐

Виконання кваліфікаційної роботи

☐

Державна підсумкова атестація

☐

ДПА

☐

Атестація

☐

Канікули

☐

Рис. 2.1. – Графік навчального процесу

У наступному розділі наведено таблицю зведених даних за бюджетом часу (у тижнях) (Рис. 2.2) – тривалість теоретичного навчання, семестрового контролю, практик, виконання кваліфікаційної роботи, атестації здобувачів фахової передвищої освіти, канікул за курсами і за весь період навчання.

курс	Теоретичне навчання	Семестровий контроль	Практика			Виконання кваліфікаційної роботи	Атестація здобувачів фахової передвищої освіти	Канікули	Всього тижнів н.р.
			навчальна	технологічна	переддипломна				
I	40							12	52
II	35	1	3				2	11	52
III	30	4	4	3				11	52
IV	30	4			3	3	1	2	43
Разом	135	9	7	3	3	3	3	36	199

Рис. 2.2. – Зведені дані за бюджетом часу

Практична підготовка (Рис. 2.3), зазначена у третьому розділі, містить перелік усіх її видів (навчальна, технологічна, переддипломна) та виконання кваліфікаційної роботи, їх обсяг у кредитах ЄКТС, тривалість у тижнях та терміни проведення (семестри).

Назва практики			Семестр	Тижнів	Кредити ЄКТС
Навчальна практика	Технічне обслуговування комп'ютерної техніки	ТОКТ	4	1	1,5
	Основи програмування	ОП	4	2	3
	Системи управління та бази даних	СУБД	5	2	3
	Технології розробки прикладного програмного забезпечення	ТППЗ	6	2	3
	Технологічна		6	3	4,5
Переддипломна			8	3	4,5
Виконання кваліфікаційної роботи					4,5
Всього				13	24

Рис. 2.3. – Практична підготовка

Інформацію щодо атестації здобувачів фахової передвищої освіти (Рис. 2.4) розміщено у четвертому розділі. Зазначено форму підсумкового контролю – кваліфікаційна робота та її обсяг у кредитах ЄКТС (1,5) у восьмому семестрі.

Форма атестації	Семестр	Кредити ЄКТС
Кваліфікаційна робота	8	1,5

Рис. 2.4. – Атестація здобувачів фахової передвищої освіти

У наступному розділі розміщено план навчального процесу (Рис. 2.5). У ньому зазначено перелік та обсяг усіх освітніх компонентів ОПП у кредитах

ЄКТС та годинах, розподіл загального обсягу годин, закріплених за освітніми компонентами на аудиторні (лекції, лабораторні або практичні, семінарські) заняття та самостійну роботу здобувачів освіти, кількість навчальних тижнів у кожному семестрі, загальну кількість годин тижневого навантаження та обсяг кредитів ЄКТС, відведених на вивчення освітнього компонента за курсами та семестрами відповідно до структурно-логічної схеми, кількість екзаменів, заліків, курсових робіт.

Код освітньої компоненти	Назва освітньої компоненти / навчального предмета	Розподіл за семестрами		Загальний обсяг	Кількість годин				Розподіл навчальної роботи в кредитах ЄКТС та аудиторних годинах за курсами, семестрами, тижнями ^н																
		Експеримент	Варіант		ЄКТС	Години	Всього	аудиторна	у тому числі	самостійна	1-й курс				2-й курс				3-й курс				4-й курс		
				1-й семестр							2-й семестр		3-й семестр		4-й семестр		5-й семестр		6-й семестр		7-й семестр		8-й семестр		
				17 тижнів							23 тижнів		17 тижнів		18 тижнів		15 тижнів		15 тижнів		17 тижнів		13 тижнів		
				теоретична							практична	теоретична	практична	теоретична	практична	теоретична	практична	теоретична	практична	теоретична	практична	теоретична	практична	теоретична	практична

Рис. 2.5. – Шапка плану навчального процесу

Освітні компоненти у плані навчального процесу розподілено на обов'язкові: за програмою профільної середньої освіти та за освітньо-професійною програмою.

Обов'язкові освітні компоненти навчального плану за програмою профільної середньої освіти (Рис. 2.6) в свою чергу поділено на базові, вибірково-обов'язкові предмети, профільні та спеціальні курси.

ОБОВ'ЯЗКОВІ ОСВІТНІ КОМПОНЕНТИ НАВЧАЛЬНОГО ПЛАНУ ЗА ПРОГРАМОЮ ПРОФІЛЬНОЇ СЕРЕДНЬОЇ ОСВІТИ																										
2.1. БАЗОВІ ПРЕДМЕТИ																										
2.1.1.	Українська мова	140	140	140						2	2	2	2													
2.1.2.	Українська література	140	140	140						2	2	2	4													
2.1.3.	Зарубіжна література	74	74	74						1	1	2														
2.1.4.	Іноземна мова ¹ (іноземна мова (за професійною спрямованістю) ²)	180	180	180						2	2	2	2	2												
2.1.5.	Історія України	114	114	114						1	2	3														
2.1.6.	Всесвітня історія	74	74	74						1	1	2														
2.1.7.	Економіка України																									
	Основи права	34	34	34						2																
	Основи економічної теорії*	51	51	51						3																
2.1.8.	Математика (алгебра і геометрія) або математика та геометрія ³ (лінійна алгебра) ⁴	237	237	237						4	2	3	4													
2.1.9.	Біологія і екологія	114	114	96	8	10				1	2	3														
	Екологія	34	34	34						2																
2.1.10.	Географія	92	92	92						4																
	Фізика і астрономія ⁵ (фізика)*	253	253	225	16	12				4	5	3	2													
2.1.12.	Хімія	114	114	86	12	16				1	2	3														
2.1.13.	Фізична культура	114	114	114						2	2	2														
	Фізична культура	57	57	37						1	1	1														
2.1.14.	Землезнавство (Основи медичних наук)	114	114	114						2	2	2														
Резюме:		64,7	1924	1924	1859	26	39	0	37	6,0	30	0,0	37	0,0	70	0,0	0	0,0	0	0,0	0	0,0	0	0,0	0	0,0
2.2. ВИБІРКОВО-ОБОВ'ЯЗКОВІ ПРЕДМЕТИ																										
2.2.1.	Інформатика (Базовий модуль. Креслення (професійне)) Креслення (графіка)*	2,8	114	114	64	40	10			4	2,3	2	*													
2.2.2.	Технології (Алгоритми та методи обчислень)*	2,8	69	69	35	34					3	*														
	Технології (Основи програмування)*	2,8	68	68	34	34						4	*													
Резюме:		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2.3. ПРОФІЛЬНІ ПРЕДМЕТИ ТА СПЕЦІАЛЬНІ КУРСИ																										
	Кваліфікаційна спеціалізація та архітектура мовознавства*	2,6	108	108	68	40																				
2.3.1.	Електроніка та основи електроніки*	72	72	36	10	26																				
	Спеціальні курси																									
2.3.2.	Навчальна практика (ТОСТ і СІП)*	2,0	90	90																						
Резюме:		6,0	270	270	104	50	116	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Додаткові години (факультативні курси, вихідні на підтримку, відпочинок)*	7,2	215	215																						
Усього за профільною середньою освітою		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Рис. 2.6. – Обов'язкові освітні компоненти навчального плану за програмою профільної середньої освіти

Обов'язкові освітні компоненти навчального плану за освітньо-професійною програмою (Рис. 2.7) поділено на освітні компоненти, що

формують загальні компетентності, та освітні компоненти, що формують спеціальні компетентності.

За освітньо-професійною програмою																										
1. Освітні компоненти, що формують загальні компетентності																										
ОК.01	Історія державності та культури України (історія України, культурологія)	4		3,3	105	54	18	36	51															3	3,3	
ОК.02	Українська мова (за професійним спрямуванням)	4		3,3	105	54	8	46	51															3	3,3	
ОК.03	Іноземна мова (за професійним спрямуванням)/ Іноземна мова *	6	4,5	2,0	210	120	8	112	80															*	2,0	4
ОК.04	Основи філософських знань (філософія, релігієзнавство)		4		3,0	90	54	30	24	36														3	3,0	
ОК.05	Основи правління*	1		3,0	90	54	20	14	36																	
ОК.06	Основи економічних знань*	1		3,3	105	60	34	34	37																	
ОК.07	Математичний аналіз	5		3,0	90	60	30	30	30															4	3,0	
ОК.08	Висока математика/ Математика (алгебра і початки аналізу та геометрія)*	4		3,0	180	72	36	36	72															*	3,0	
ОК.09	Фізика/ Фізика і астрономія*	4		4,0	120	36	20	16	84															*	4,0	
ОК.10	Основи авторів праць та безпека життєдіяльності	8		4,0	120	78	48	30	42																	6
ОК.11	Фізичне виховання	4/4,7		10,0	300	276	10	216	74															2/2	2,0	2/2
Разом за всіма загальними підготовками		3	13		49,5	1485	856	262	0	594	629	0	4,5	0	0	0	0	11	23	10	9	4	5	2	2	6
2. Освітні компоненти, що формують спеціальні компетентності																										
КК.01	Дискретна математика	5		3,0	90	60	30	30	30															4	3,0	
КК.02	Теорія ймовірностей та математична статистика	6		3,0	90	60	34	26	30																4	3,0
КК.03	Електротехніка та основи електроніки*	3		4,0	120	72	36	10	26	48													*	4,0		
КК.04	Комп'ютерна систематика та архітектура комп'ютера*	4		6,0	180	108	48	40	72														*	6,0		
КК.05	Комп'ютерні мережі	5	3	3,0	180	60	30	30	90																4	3,0
КК.06	Алгоритми та методи обчислення/ Технології*	2		4,0	120	60	35	34	51														*	4,0		
КК.07	Основи програмування/ Технології*	4		4,0	120	60	34	34	52														*	4,0		
КК.08	Операційні системи та їх обслуговування	5		4,0	120	60	30	30	60																4	4,0
КК.09	Об'єктно-орієнтоване програмування	6	5	6,0	180	120	60	60	60																4	3,0
КК.10	Організація баз даних	6	5	3,3	165	120	60	30	30	45															4	3,0
КК.11	Комп'ютерна графіка/ Інформатика*	5		4,0	120	46	74	74	74														*	4,0		

Рис. 2.7. – Обов'язкові освітні компоненти навчального плану за освітньо-професійною програмою

План освітнього процесу також містить освітні компоненти за вибором здобувача освіти, призначені для забезпечення можливості поглибити професійні знання та здобути додаткові спеціальні компетентності. Вибіркові дисципліни сформовано у блоки. При виборі певного блоку всі освітні компоненти що входять до нього, стають обов'язковими для вивчення.

Наступним розділом навчального плану є перелік необхідних кабінетів, лабораторій і майстерень наявних у коледжі.

Розділ сьомий являє собою пояснення до навчального плану, де уточнено окремі положення та особливості організації освітнього процесу.

У плані навчального процесу серед переліку обов'язкових компонентів наявний предмет **«WEB-програмування. Інформатика»** в змістовому модулі **«Креативне програмування»**, який вивчають студенти коледжу третіх курсів у другому семестрі.

У календарному плануванні предмету пропонуються теми лекційних і практичних занять, спрямовані на формування практичних навичок у сфері

веброзробки. Лекційні заняття охоплюють такі ключові напрями, як принципи роботи веброзробки, основи HTML, CSS, сучасні технології верстки Flexbox і Grid, а також базові можливості JavaScript.

Таблиця 2.1 – Теми лекцій

№ з/п	№ теми	Назви теми	кількість годин
1	1	Вступ до вебпрограмування. Структура та принципи роботи веброзробки	2
2	2	HTML: структура вебсторінки та основні теги	1
3	3	CSS: основи стилізації вебсторінок	1
4	4	Розміщення елементів на сторінці: Flexbox та Grid	1
5	5	Основи JavaScript для веброзробника	2

Практичні заняття включають створення вебсторінок, роботу з мультимедійними елементами, застосування стилів, побудову макетів та виконання проєктів, що дозволяє студентам застосувати отримані знання на практиці.

Таблиця 2.2 – Теми лабораторних занять

№ з/п	№ теми	Назви теми	кількість годин
1	1	Створення простої HTML-сторінки	1
2	2	Додавання мультимедійних елементів у вебсторінку	1
3	3	Верстка сторінки за допомогою CSS	1
4	4	Практика Flexbox: побудова макету вебсторінки	1
5	5	Основи JavaScript: робота зі змінними та подіями	1
6	6	Проект: створення вебсторінки	2

У рамках програми предмету «WEB-програмування. Інформатика» в змістовому модулі «Креативне програмування» доцільним є вивчення редактора коду Visual Studio Code (Рис. 2.8). Цей редактор є популярним серед веброзробників завдяки своїй зручності та функціональності. VS Code легко інтегрується з мовами й інструментами, що активно використовуються у веброзробці, такими як HTML, CSS, JavaScript, а також підтримує велику кількість розширень, які розширюють функціональність робочого середовища.

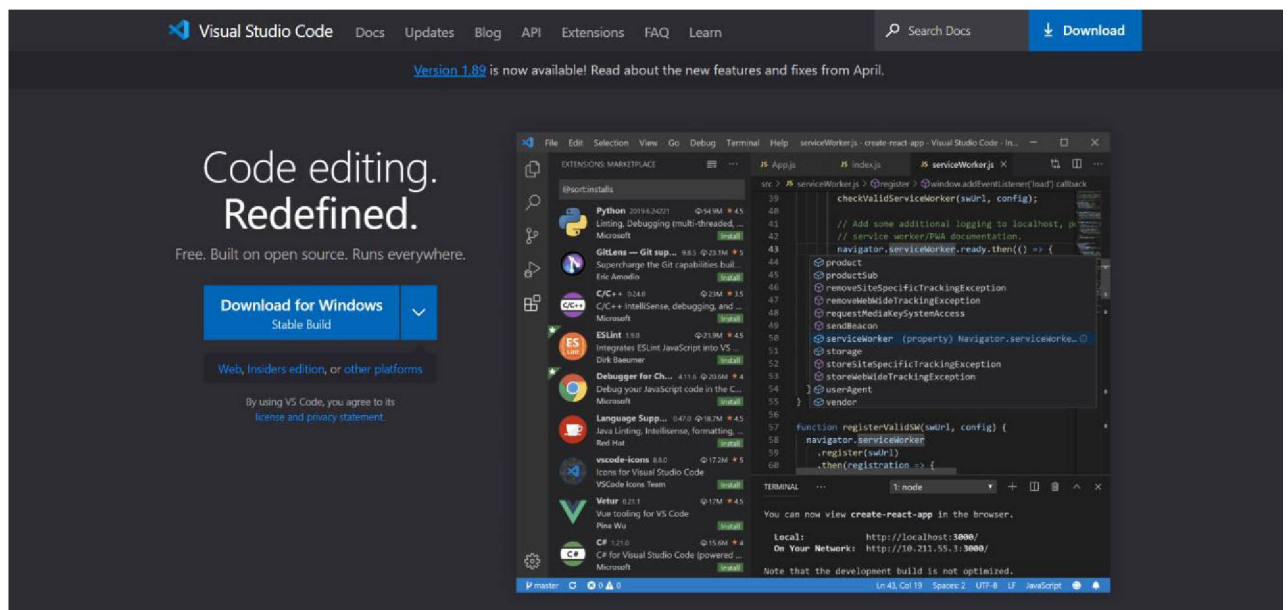


Рис. 2.8. – Офіційний сайт Visual Studio Code

Вивчення **VS Code** на заняттях дасть змогу студентам працювати над вебпроектами у професійному середовищі, використовуючи корисні інструменти, такі як автодоповнення коду, підсвічування синтаксису, вбудовану систему контролю версій Git, панель терміналу, зручну навігацію між файлами та інтеграцію з системами розробки. Також цей редактор допомагає формувати навички, які є необхідними у професійній діяльності веброзробника, оскільки VS Code є одним із ключових інструментів галузі.

2.2 Методика навчання студентів коледжів створенню вебсайтів у середовищі VS Code

Методика навчання студентів коледжів створенню вебсайтів у середовищі Visual Studio Code ґрунтується на поєднанні практичної спрямованості навчання, поетапного ускладнення навчального матеріалу та активного залучення студентів до реальної діяльності веброзробника. Середовище VS Code доцільно використовувати як базовий інструмент, оскільки воно є безкоштовним, кросплатформним і широко застосовується у професійній сфері, що сприяє формуванню в студентів актуальних цифрових і професійних компетентностей.

Навчання доцільно розпочинати з ознайомлення студентів з інтерфейсом VS Code, його основними можливостями та принципами організації робочого простору. На цьому етапі важливо сформувати вміння створювати та структурувати проєкт вебсайту, працювати з папками і файлами, використовувати вбудований термінал, а також підключати необхідні розширення для роботи з HTML, CSS і JavaScript. Пояснення супроводжується демонстрацією дій викладача та виконанням простих практичних завдань студентами.

Подальша методика передбачає вивчення основ вебтехнологій через практичні приклади. Засвоєння HTML здійснюється шляхом створення простих вебсторінок із поступовим ускладненням структури, використанням семантичних тегів та формуванням навичок читабельного коду. Робота з CSS організовується через стилізацію вже створених сторінок, що дозволяє студентам одразу бачити результат своєї діяльності. У середовищі VS Code активно використовуються підказки, автодоповнення коду та попередній перегляд, що знижує рівень помилок і підвищує мотивацію до навчання.

Важливим компонентом методики є навчання налагодженню та аналізу коду. Студенти поступово опановують роботу з помилками, вчать читати повідомлення середовища розробки та використовувати інструменти перевірки синтаксису. Це сприяє розвитку алгоритмічного мислення та самостійності у процесі створення вебсайтів. Значна увага приділяється формуванню культури

програмування, зокрема дотриманню стандартів оформлення коду та логіки побудови проєкту.

На завершальному етапі навчання доцільно застосовувати проєктну діяльність, у межах якої студенти створюють власні вебсайти за заданою або обраною тематикою. Робота над проєктом у VS Code дозволяє інтегрувати знання з різних тем, розвивати навички командної взаємодії та презентувати результати своєї роботи. Таким чином, методика навчання створенню вебсайтів у середовищі VS Code забезпечує поступовий перехід від базових умінь до комплексного використання вебтехнологій і сприяє підготовці студентів до реальних умов професійної діяльності.

Лекції відіграють важливу роль у процесі вивчення створення вебсайтів у середовищі Visual Studio Code, оскільки вони забезпечують теоретичну основу, на якій вибудовується подальша практична діяльність студентів. Саме під час лекцій формується цілісне уявлення про веброзробку як професійну діяльність, розкриваються базові поняття вебтехнологій, принципи клієнт-серверної взаємодії, логіка побудови вебпроєктів і місце середовища VS Code у цьому процесі.

У межах лекційного заняття викладач має змогу систематизувати навчальний матеріал, пояснити призначення та можливості VS Code як сучасного інструменту веброзробки, обґрунтувати вибір саме цього середовища для навчання. Лекція допомагає студентам зрозуміти структуру вебпроєкту, призначення HTML, CSS і JavaScript, а також логіку їх взаємодії, що є необхідним для усвідомленого виконання практичних завдань. Теоретичні пояснення, підкріплені демонстрацією коду та інтерфейсу середовища, сприяють кращому сприйняттю матеріалу та знижують складність подальшої самостійної роботи.

Важливою функцією лекції є орієнтаційна та мотиваційна складова. Під час лекційного заняття студенти отримують уявлення про практичну цінність знань зі створення вебсайтів, можливості їх застосування у майбутній професійній діяльності та актуальність володіння інструментами сучасної

розробки. Це формує внутрішню мотивацію до опанування матеріалу та підвищує зацікавленість у роботі з VS Code.

Крім того, лекція виконує інтегративну роль, поєднуючи теоретичні знання з практикою. Викладач може аналізувати типові помилки, звертати увагу на логіку написання коду, правила його оформлення та ефективної організації робочого середовища. Таким чином, лекція не обмежується лише передаванням інформації, а стає основою для формування професійного мислення майбутніх фахівців, забезпечуючи усвідомлене та результативне вивчення створення вебсайтів у середовищі VS Code. Розглянемо приклад лекції:

Тема: Огляд середовища Visual Studio Code як інструменту для створення вебсайтів.

Мета: Ознайомити студентів з процесом встановлення та базового налаштування редактора Visual Studio Code, а також навчити ефективно використовувати розширення (плагіни) для підвищення продуктивності при розробці сайту.

ПЛАН

1. Завантаження та встановлення редактора Visual Studio Code.
2. Початкове налаштування VS Code.
3. Огляд інтерфейсу редактора.
4. Підключення та використання плагінів. Огляд корисних плагінів: Prettier, Formatting Toggle, HTML CSS Support, Live Server, eCSStractor.
5. Переваги використання плагінів у VS Code.

Для того, щоб завантажити редактор Visual Studio Code, потрібно перейти на офіційний сайт (Рис. 2.9) <https://code.visualstudio.com/> і обрати інсталятор відповідно до вимог операційної системи.

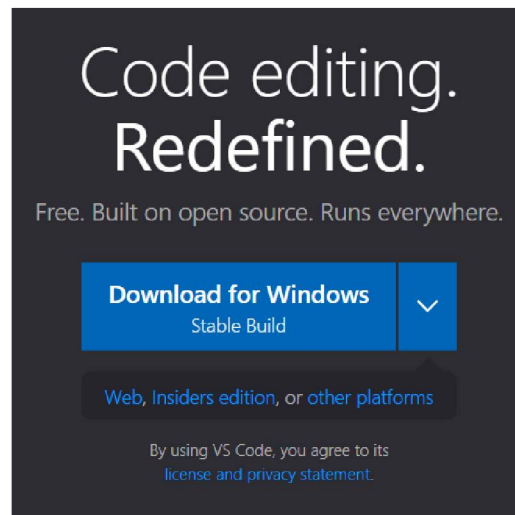


Рис. 2.9. – Інсталятор VS Code

Після того, як завантаження завершиться, у папці “download” з’явиться іконка інсталятора, на яку потрібно натиснути, щоб завершити процес встановлення.

Спочатку потрібно буде обрати мову. Для оптимальної роботи з редактором радять встановити англійську (Рис. 2.10).

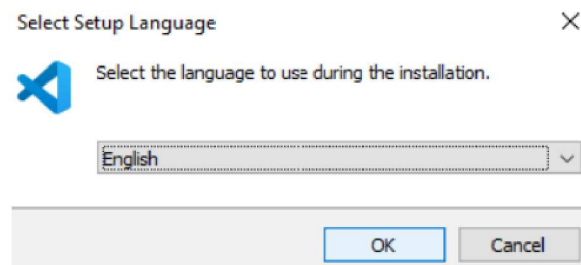


Рис. 2.10. – Вибір мови в редакторі VS Code

Після запуску інсталятора пропонується прийняти умови використання, без чого буде неможливо продовжувати роботу з програмою. Далі з’явиться віконце з місцем розташування файлів для запуску VS Code. Потрібно обрати папку в яку буде завантажено програму або залишити те, що запропоновано інсталятором. Також можна поставити “пташку” для того, щоб створити ярлик на робочому столі пристрою.

Після цих дій з'явиться вікно з пропозицією розпочати налаштування інсталяції, де потрібно натиснути “Install” і зачекати декілька хвилин поки програма встановиться.

На робочому столі з'явиться ярлик Visual Studio Code (Рис. 2.11).

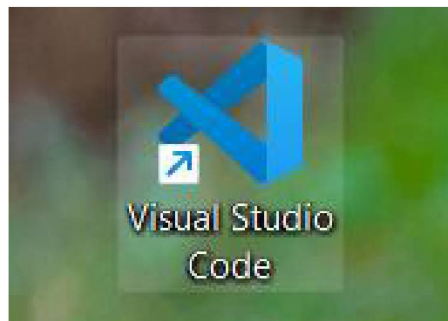


Рис. 2.11. – Ярлик VS Code

Важливим пунктом після встановлення редактора є підключення плагінів.

Плагіни – це додатки, які дають можливість розширювати функціональність редактора. Вони створені розробниками або спільнотою користувачів VS Code.

Плагіни дозволяють налаштувати редактор під свої потреби, розширити його функціональність, покращити продуктивність та зробити роботу з кодом більш зручною та ефективною.

VS Code має велику кількість плагінів, які можна встановити безпосередньо з магазину або з використанням командного рядка.

Prettier – плагін, який дозволяє автоматично формувати код. Особливо зручний при роботі з великими проєктами, де вимоги до форматування можуть бути строгими.

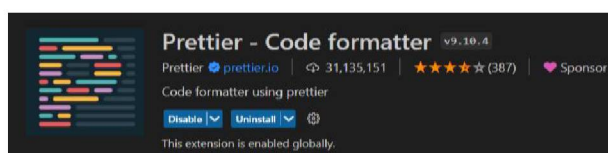


Рис. 2.12. – Плагін Prettier

Formatting Toggle – плагін, який дозволяє користуватися форматуванням лише тоді, коли це потрібно та працювати без нього, якщо так більш зручно.



Рис. 2.13. – Плагін Formatting Toggle

HTML CSS Support – плагін, який забезпечує підтримку CSS у HTML – файлах та забезпечує автодоповнення CSS – коду.

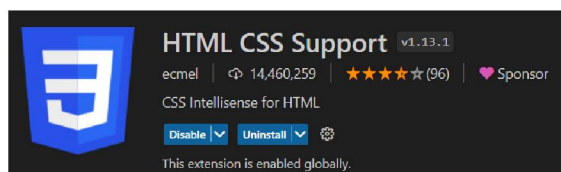


Рис. 2.14. – Плагін HTML CSS Support

Live Server – плагін, який дозволяє автоматично оновлювати відображення вебсторінки при внесенні змін у вихідний код, що забезпечує швидку перевірку змін і дає можливість ефективно використовувати час.

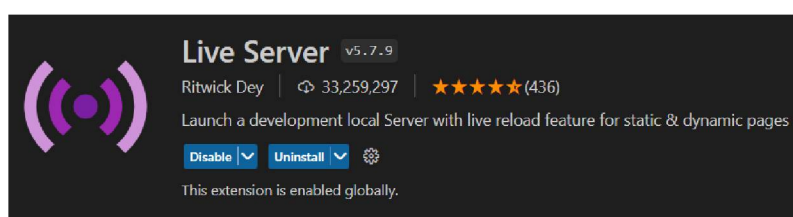


Рис. 2.15. – Плагін Live Server

eCSStractor for VS Code – плагін, який дозволяє швидко додавати селектори у файл CSS без необхідності вводити кожен з них вручну.

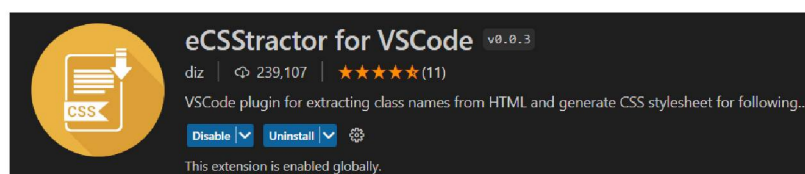


Рис. 2.16. – Плагін eCSStractor for VS Code

Переваги використання плагінів:

- допомагає підвищити продуктивність розробки, даючи можливість виконувати потрібні задачі та забезпечувати швидкий доступ до потрібних інструментів.
- робить процес написання коду більш зручним і зрозумілим.
- допомагає збільшити точність і якість коду, автоматично форматуючи його за заданим шаблоном.
- дозволяє зосередитись на процесі написання коду, забезпечуючи зручний та продуктивний робочий процес.

Ще одною важливою формою вивчення створення сайтів є лабораторні роботи, тому що відіграють ключову роль у вивченні створення вебсайтів у середовищі Visual Studio Code, оскільки саме в процесі їх виконання теоретичні знання перетворюються на практичні вміння та стійкі навички веброзробки. Вони забезпечують активну участь студентів у навчальному процесі й створюють умови для безпосередньої роботи з кодом у професійному середовищі, максимально наближеному до реальних умов діяльності веброзробника.

Під час лабораторних занять студенти навчаються створювати та організовувати структуру вебпроектів у VS Code, працювати з файлами HTML, CSS і JavaScript, використовувати розширення середовища та вбудований термінал. Виконання практичних завдань дозволяє поступово опановувати інструментарій редактора, розуміти логіку його роботи та формувати навички ефективної взаємодії з середовищем розробки. Важливо, що результати роботи студент одразу бачить у браузері, що сприяє усвідомленню причинно-наслідкових зв'язків між кодом і візуальним відображенням вебсторінки.

Лабораторні роботи також сприяють розвитку самостійності та відповідальності за результат власної діяльності. У процесі виконання завдань студенти стикаються з помилками в коді, навчаються аналізувати повідомлення середовища VS Code, знаходити та виправляти неточності. Це формує навички налагодження коду, логічного мислення та здатність до самоконтролю, що є надзвичайно важливими для майбутньої професійної діяльності.

Крім того, лабораторні заняття створюють умови для диференційованого та індивідуального підходу до навчання. Студенти з різним рівнем підготовки можуть працювати у власному темпі, ускладнювати завдання або розширювати функціональність вебсайту. Викладач у цьому випадку виконує роль консультанта, спрямовуючи діяльність студентів і надаючи допомогу в разі виникнення труднощів.

Таким чином, лабораторні роботи є основною формою формування практичних компетентностей зі створення вебсайтів у середовищі VS Code, забезпечуючи інтеграцію теорії та практики, розвиток професійних навичок і підготовку студентів коледжів до реальних завдань у сфері веброзробки. Розглянемо приклад лабораторної роботи.

Лабораторна робота

Тема: Створення та стилізація вебсторінки за допомогою HTML і CSS у редакторі Visual Studio Code.

Мета: Навчитися створенню та стилізації вебсторінок за допомогою мов HTML і CSS. Здобути знання про основні теги HTML, структуру HTML-документа, а також вміти застосовувати CSS для зміни вигляду елементів на вебсторінці.

План роботи

1. Вивчити теоретичні відомості.
2. Створити та стилізувати вебсторінку за допомогою HTML і CSS у редакторі Visual Studio Code.
3. Результати роботи оформити у вигляді звіту.

Теоретичні відомості

HTML – це мова розмітки, яка використовується для створення структури вебсторінок. Елементи HTML: `<header>`, `<nav>`, `<main>`, `<section>` та `<footer>` визначають різні частини сторінки.

Тег `<header>` використовується для представлення заголовка на сторінці. Може включати логотип та основне навігаційне меню.

Тег `<nav>` використовується для створення навігаційного меню, яке дозволяє користувачам переходити між різними сторінками або розділами сторінки.

`<main>` використовується для обгортання основного вмісту сторінки. Це може бути текст, зображення, відео або інші медіа-елементи.

`<section>` використовується для групування схожого вмісту на сторінці.

Елементи списку `<ul>` та `<ol>` використовуються для створення нумерованих та нумерованих списків.

`<footer>` використовується для представлення нижньої частини сторінки. Може включати інформацію про авторські права, посилання на соціальні мережі та кнопку для відправлення запиту на зворотний дзвінок.

CSS – це мова таблиці стилів, що використовується для опису вигляду та форматування вебсторінок, написаних мовою HTML. Кожне правило стилю в CSS має властивість та значення. Наприклад, *font-family: Arial, sans-serif;* задає шрифт для тексту, використовуючи Arial або sans-serif як резервний шрифт.

CSS використовує селектори для вибору елементів, до яких будуть застосовані стилі. Наприклад, *header* вибирає заголовок сторінки, а *header a* вибирає всі посилання всередині заголовка.

CSS розглядає кожен елемент як прямокутник, який має властивості, такі як *width*, *height*, *margin*, *padding*, *border*. Наприклад, *padding: 1em;* задає відступ всередині елемента.

Властивість *margin: 0 auto;* вирівнює елемент по центру горизонталі.

CSS дозволяє задавати колір фону *background-color*, колір тексту *color*, розмір шрифту *font-size* та інші параметри для оформлення тексту та фону елементів.

Використання *display: flex;* та інших властивостей *flexbox* дозволяє створювати гнучкі макети для розташування елементів на сторінці.

Завдання для виконання

1. Створити нову папку у редакторі VS Code. У цій папці створити файл `index.html` для головної вебсторінки та папку `css` для CSS-файлів.

2. Використовуючи HTML, створити основну структуру вебсторінки в файлі `index.html`. Вона повинна включати такі елементи, як заголовок, навігаційне меню, основний контент та нижній колонтитул.

3. Використовуючи CSS, стилізувати вебсторінку. Створити окремий CSS-файл у папці `css` та підключити його до HTML-файлу. Використовувати різні CSS-властивості для зміни вигляду елементів на сторінці, таких як кольори, шрифти, відступи тощо.

4. Відкрити вебсторінку в браузері, щоб перевірити результат. Переконалися, що всі елементи відображаються правильно.

5. Оформити звіт виконаної роботи.

Питання для закріплення матеріалу:

1. Як використовується тег `<img>` для вставки зображень в HTML?
2. Як створити навігаційне меню в HTML?
3. Як використовується тег `<a>` для створення посилань в HTML?
4. Наведіть приклади властивостей CSS.
5. Що означає властивість `line-height` в CSS і як вона впливає на відображення тексту?

Домашнє завдання

Створити та стилізувати вебсторінку за допомогою HTML і CSS у редакторі Visual Studio Code та оформити звіт.

Хід роботи

Розробка структури та контенту вебсайту засобами HTML

Для початку відкриваємо редактор Visual Studio Code на комп'ютері. У лівій панелі натискаємо на значок папки, що знаходиться у верхньому куті. Відкриється вікно “Explorer”, натискаємо на кнопку “New Folder” або використаємо комбінацію гарячих клавіш `Ctrl + Shift + N`, щоб створити нову папку. Далі вводимо назву та натискаємо “Enter”.

У новій папці створимо “New File” або використаємо комбінацію клавіш `Ctrl + N`, щоб створити новий файл. Називаємо його “`index.html`” та натискаємо “Enter”.

Тепер можна розпочинати наповнення сторінки за допомогою HTML – коду.

Структура **першої** сторінки сайту складається з *шапки, основного змісту і нижнього колонтитулу*.

Шапка сайту – верхня частина вебсторінки, яка відображає інформацію про сайт. Містить логотип, меню навігації і контактну інформацію.

У тег `<header>`, який вказує на те, що вся інформація яка міститься в ньому є шапкою сторінки, прописано тег `<nav>` в якому містяться елементи навігаційного меню.

Створений логотип, за допомогою елементу `<a>`, містить назву студії “WebStudio”. Атрибути `href` у якому вказано посилання, яке відкриється користувачу при кліку на логотип та `lang`, який вказує на мову, якою відображається вміст в елементі, тобто англійська.

Елемент `<ul>` містить список елементів `<li>`, які містять посилання на інші сторінки сайту. Наприклад, перше веде на головну сторінку “WebStudio”, тобто при натисканні на логотип або “Студія” користувача буде направляти на початок сайту.

Наступний елементі `<ul>` містить елементи `<li>`, які мають посилання на електронну пошту та телефон компанії. Атрибут `href` вказує на адресу, на яку буде відправлено лист або здійснено дзвінок при кліку на посилання.

```
<header>

  <nav>
    <a href="./index.html" lang="en">WebStudio</span></a>
    <ul>
      <li><a href="./index.html">Студія</a></li>
      <li><a href="#">Контакти</a></li>
    </ul>
  </nav>

  <ul>
    <li><a href="mailto:info@devstudio.com">info@devstudio.com</a></li>
    <li><a href="tel:+380961111111">+38 096 111 11 11</a></li>
  </ul>

</header>
```

Рис. 2.25. – Шапка сайту наповнена HTML – кодом



Рис. 2.26. – Вигляд шапки у браузері

Основний зміст – це частина сайту між шапкою і нижнім колонтитулом, яка складається з блоків, обгорнених в елементи `<section>`, а ті в свою чергу знаходяться в `<main>`.

Перший блок – відображає заголовок першого рівня – тег `<h1>` з текстом “Ефективні рішення для вашого бізнесу” і кнопкою – тег `<button>`, перейшовши за якою браузер відкриває нову сторінку де можна замовити послугу.

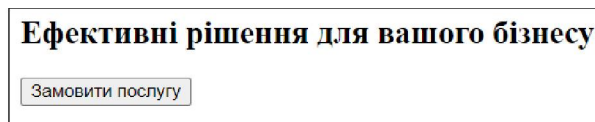


Рис. 2.27. – Вигляд першого блоку сторінки у браузері

Другий блок – особливості студії, відображає нумерований список, за допомогою тегу `<ul>`. Створено заголовок третього рівня і підзаголовки до кожного.

```

<ul>
  <li>
    <h3>УВАГА ДО ДЕТАЛЕЙ</h3>
    <p>
      Ідейні міркування, і навіть початок повсякденної роботи з
      формування позиції.
    </p>
  </li>
  <li>
    <h3>ПУНКТУАЛЬНІСТЬ</h3>
    <p>
      Завдання організації, особливо рамки і місце навчання кадрів тягне
      у себе.
    </p>
  </li>
  <li>
    <h3>ПЛАНУВАННЯ</h3>
    <p>
      Так само консультація з широким активом значною мірою зумовлює.
    </p>
  </li>
  <li>
    <h3>СУЧАСНІ ТЕХНОЛОГІЇ</h3>
    <p>
      Значимість цих проблем настільки очевидна, що реалізація планових
      завдань.
    </p>
  </li>
</ul>

```

Рис. 2.28. – Приклад другого блоку

• УВАГА ДО ДЕТАЛЕЙ Ідейні міркування, і навіть початок повсякденної роботи з формування позиції. • ПУНКТУАЛЬНІСТЬ Завдання організації, особливо рамки і місце навчання кадрів тягне у себе. • ПЛАНУВАННЯ Так само консультація з широким активом значною мірою зумовлює. • СУЧАСНІ ТЕХНОЛОГІЇ Значимість цих проблем настільки очевидна, що реалізація планових завдань.
--

Рис. 2.29. – Вигляд другого блоку сторінки у браузері

Третій блок – зображення роботи команди, який також розроблений за допомогою нумерованого списку. Заголовок другого рівня, створений за допомогою тегу `<h2>`, має назву “Чим ми займаємося”. Кожен елемент списку містить тег `<img>`, в якому відображаються зображення з файлу. За допомогою атрибуту `src` вказано їх розташування, альтернативний текст, якщо зображення не завантажиться на сторінці, а також ширина – 370 і висота – 294.

```

<h2>Чим ми займаємося</h2>
<ul>
  <li>
    
  </li>
  <li>
    
  </li>
  <li>
    
  </li>
</ul>

```

Рис. 2.30. – Розробка третього блоку

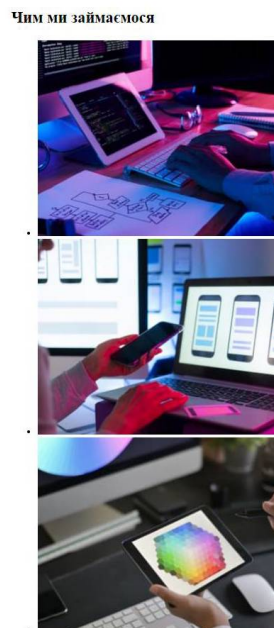


Рис. 2.31. – Вигляд третього блоку сторінки у браузері

Перед тим, як додавати зображення до свого сайту, потрібно їх оптимізувати, що зменшить вагу сторінки і пришвидшить її завантаження.

Потрібно пам'ятати, що при стисненні зображень якість погіршується та для користувача це буде зовсім непомітно.

Одним з найкращих ресурсів для оптимізації є *Squoosh* (Рис. 2.32), який дозволяє стискати зображення без сильної втрати якості.

Squoosh також має корисні функції, такі як можливість порівняти оригінальне зображення з оптимізованим, зберегти налаштування стиснення в певних форматах та використовувати вебкамеру для стиснення зображень прямо в браузері.

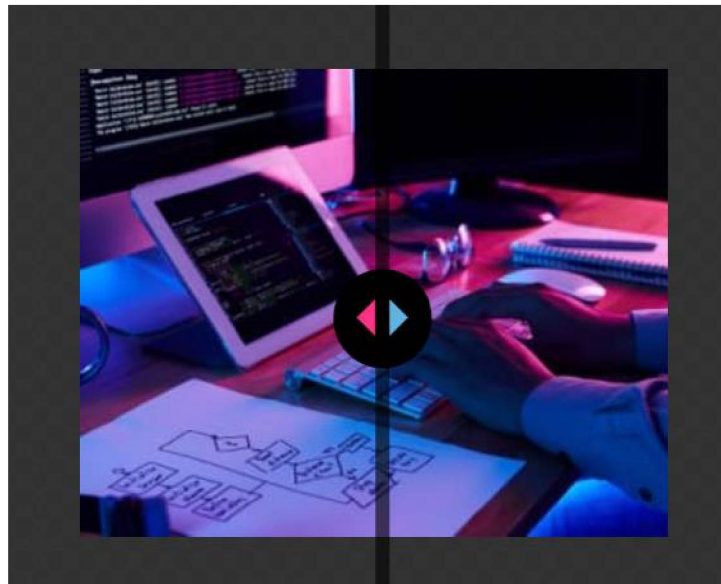


Рис. 2.32. – Squoosh – ресурс для оптимізації зображень

Четвертий блок містить інформацію про команду студії. Він складається із заголовку другого рівня – “Наша команда”, нумерованого списку, в якому є зображення працівників із шляхом до файлу, їх описи в атрибуті alt і розміри, а саме висота і ширина в пікселях.

Далі є заголовки третього рівня, в яких прописані імена працівників та підзаголовки, що містять посади працівників в команді.

```
<h2>Наша команда</h2>
<ul>
  <li>
    
    <h3>Ігор Дем'яненко</h3>
    <p lang="en">Product Designer</p>
  </li>
</ul>
```

Рис. 2.33. – Розробка четвертого блоку



Рис. 2.34. – Фото працівника на сторінці сайту

Нижній колонтитул – це частина, яка розташована внизу вебсторінки і містить логотип, адресу, де розташована студія, електронну пошту та номер телефону і є обгорненою у тег `<footer>`.

Перший рядок – логотип сайту, прописаний за допомогою тегу `<a>` так само, як і в шапці сторінки, тобто вони однакові. Наступним йде нумерований список контактної інформації, обгорнений тегом `<address>`.

У першому тезі `<a>` міститься назва вулиці студії, також у атрибутах `href` вказано посилання на карту *Google*, а у `target` – значення `_blank`, яке означає, що посилання буде відкрито у новому вікні браузера.

У другому тезі `<a>` міститься посилання на електронну пошту. У третьому – посилання на телефон студії.

```
<footer>
  <a href="/index.html" lang="en">Web<span>Studio</span></a>
  <address>
    <ul>
      <li><a href="https://goo.gl/maps/CPtrU1FHBa2aMyZL9" target="_blank">
      <li><a href="mailto:info@devstudio.com">info@devstudio.com</a></li>
      <li><a href="tel:+380961111111">+38 096 111 11 11</a></li>
    </ul>
  </address>
</footer>
```

Рис. 2.35. – Код нижнього колонтитулу



Рис. 2.36. – Вигляд нижнього колонтитулу у браузері

Структура **другої** сторінки сайту також складається з *шапки*, *основного змісту* і *нижнього колонтитулу*. Вона відкривається при натисненні кнопки “Замовити послугу” на головній сторінці та створена для зв’язку клієнтів зі студією.

Шапка і *нижній колонтитул* залишаються такими ж як на головній сторінці. **Шапка сайту** – верхня частина вебсторінки, яка відображає інформацію про сайт. Містить логотип, меню навігації і контактну інформацію.

Нижній колонтитул – це частина, яка розташована внизу вебсторінки, містить логотип, адресу, де розташована студія, електронну пошту та номер телефону.

Основний зміст сторінки містить форму з двома текстовими полями (ім'я, телефон) і кнопку для замовлення послуги.

Тег `<form>` використовується для створення форми та міститься в тезі `<section>`. Тег `<h3>` виводить заголовок третього рівня, `<p>` – опис, що буде коли користувач введе особисті дані.

Далі, тег `<div>` створює блок, який містить наступні елементи форми – `<label>` та `<input>`. `<label>` виводить на сторінку назву першого поля – Ім'я, а `<input>` створює комірку типу `text` для введення даних. Атрибут `required` означає на те, що поле є обов'язковим для заповнення. Наступний тег `<div>` створює ще один такий блок з полем для вводу номера телефону.

Внизу форми міститься кнопка з текстом “Замовити послугу”, створена за допомогою тегу `<input>` з атрибутами `type` і `value`. Перший вказує на те, що елемент є кнопкою. Другий встановлює текст, який відображатиметься у ній. Після того, як користувач натисне цю кнопку, його дані зберуться і через деякий час студія зв'яжеться з ним.

```
<section>

  <form action="#">

    <h3>Залишіть свої контакти</h3>
    <p>Ми вам зателефонуємо протягом 30 хвилин</p>

    <div>
      <label for="user-email">Ім'я</label>
      <input type="text" name="Ім'я" required/>
    </div>

    <div>
      <label for="user-email">Телефон</label>
      <input type="phone" name="Телефон" required/>
    </div>

    <input type="submit" value="Замовити послугу">
  </form>

</section>
```

Рис. 2.37. – Код форми для введення даних

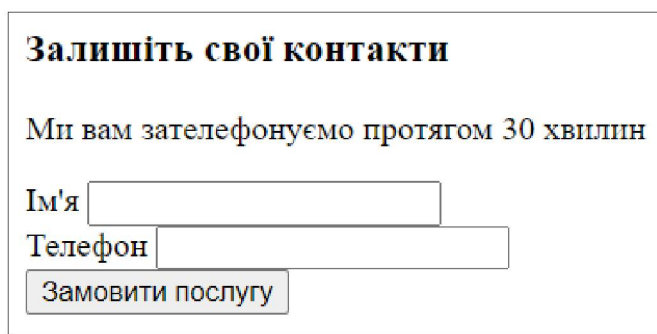


Рис. 2.38. – Вигляд форми для введення даних у браузері

Стилізація вебсайту з використанням CSS

Основні принципи використання CSS полягають у тому, що спочатку створюється CSS-файл, де описується, яким чином повинен відображатися вміст вебсторінки. У ньому можна встановлювати такі параметри як колір тексту, фону, розмір і тип шрифту, відступи, рамки тощо. Зв'язування CSS-файлу з вебсторінкою здійснюється через підключення файлу за допомогою тегу `<link>` до `<head>` вебсторінки:

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

Наступним кроком є підключення шрифтів з *Google Fonts*, для цього переходимо за посиланням <https://fonts.google.com/>, обираємо – “Relaway” та “Roboto”, різної товщини, копіюємо HTML – код, який з’явиться у вікні “Embed Font” та вставляємо його у `<head>`.

Далі до кожного елементу в HTML – файлі створюється атрибут “class” із назвою, який використовується для зазначення стилів CSS, що відповідатимуть певним групам елементів.

Усі назви класів в CSS – файлі починаються з крапки:

```
.logo {
```

Підключаючи стилі до частини логотипу “Web” у шапці профілю використовуємо такий код:

```
.logo {
font-family: 'Raleway', sans-serif;
font-weight: 700;
```

```
font-size: 26px;
line-height: 1.19;
letter-spacing: 0.03em;
color: #2196F3; }
```

Перша властивість `font-family` встановлює шрифт – “Raleway”. Значення “sans-serif” вказує на те, що якщо обраний шрифт не буде доступний, то браузер використає інший шрифт. Властивість `font-weight` встановлює товщину шрифту – “700”, `font-size` – розмір шрифту в пікселях, `line-height` – міжрядковий інтервал – 1.19, де значення вказано множителем для того, щоб при зміні розміру шрифту текст не накладався, `letter-spacing` – інтервал між буквами та `color` – колір, в нашому випадку – синій відтінок.

Для частини логотипу – “Studio” змінюємо колір на чорний:

```
.logo-header {
color: #000000; }
```

Змінюємо стилі для “студіо”, “контакти”, номери телефону і пошти таким чином:

```
.menu-link {
font-family: 'Roboto', sans-serif;
font-weight: 500;
font-size: 14px;
line-height: 1.14;
letter-spacing: 0.02em;
color: #212121; }
```

Для всіх елементів в шапці вебсайту задаємо селектори, використовуючи псевдокласи “hover” і “focus”, щоб при наведенні курсору миші на якийсь з елементів його колір змінився на синій відтінок і мав фокус при використанні клавіші Tab на клавіатурі, наприклад:

```
.menu-link:hover,
.menu-link:focus {
```

```
color: #2196F3; }
```

Такий вигляд буде мати шапка вебсайту:



Рис. 2.39. – Шапка вебсайту

Надалі властивості будуть повторюватися, змінюються лише їх значення.

Для першого блоку основної частини вебсайту задаємо фон – темно-сірого відтінку, розміщуємо заголовок і кнопку посередині:

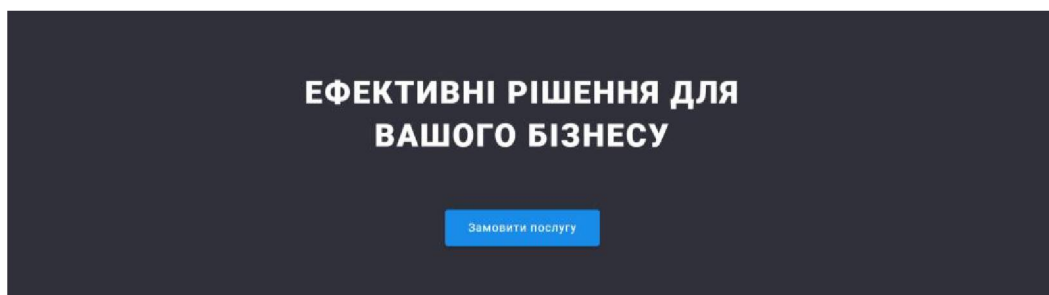


Рис. 2.40. – Перший блок основної частини вебсайту

У другому блоці в якому описуються особливості студії, елементи розміщуються посередині, один біля одного:

УВАГА ДО ДЕТАЛЕЙ	ПУНКТУАЛЬНІСТЬ	ПЛАНУВАННЯ	СУЧАСНІ ТЕХНОЛОГІЇ
Ідейні міркування, і навіть початок повсякденної роботи з формування позиції.	Завдання організації, особливо рамки і місце навчання кадрів тягне у себе.	Так само консультація з широким активом значною мірою зумовлює.	Значимість цих проблем настільки очевидна, що реалізація планових завдань.

Рис. 2.41. – Другий блок основної частини вебсайту

У третьому і четвертому блоці заголовки, зображення та підзаголовки також розміщуємо посередині:

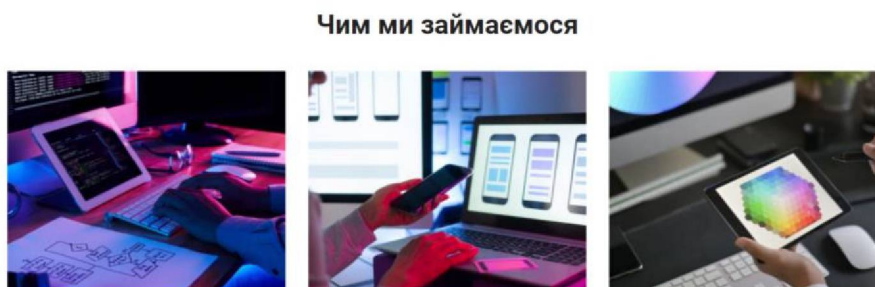


Рис. 2.42. – Третій блок основної частини вебсайту

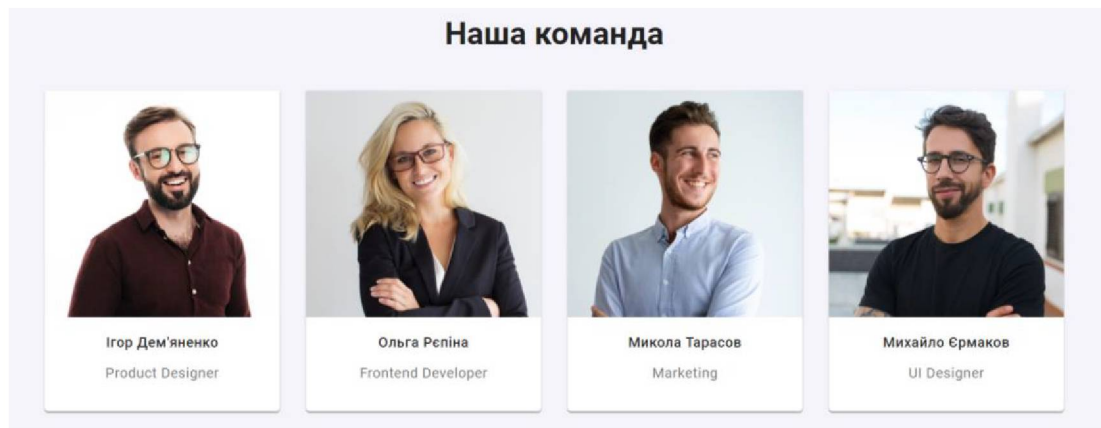


Рис. 2.43. – Четвертий блок основної частини вебсайту

У нижньому колонтитулі вебсайту до логотипу застосовуються такі ж стилі, як у шапці. Для адреси, пошти і номера телефону змінюємо значення властивостей:

```
font-family: 'Roboto', sans-serif;
font-weight: 400;
font-size: 14px;
line-height: 1.71;
letter-spacing: 0.03em;
color: rgba(255, 255, 255, 0.6);
```



Рис. 2.44. – Нижній колонтитул вебсайту

Друга сторінка сайту відкривається при натисненні кнопки “Замовити послугу” на головній сторінці та створена для зв’язку клієнтів зі студією.

Форма, яка розташована на вебсторінці, має фон світлого кольору, а сам контейнер темного. В середині нього знаходиться заголовок та опис, а також поля для введення даних користувачем і кнопка. До тексту застосовуються такі властивості, які змінюються:

```
font-family: 'Roboto', sans-serif;
font-weight: 700;
font-size: 36px;
line-height: 24px;
padding-top: 107px;
margin-bottom: 32px;
color: #FFFFFF;
```

Рис. 2.45. – Форма для зв'язку з клієнтами

Для розміщення елементів на сторінці використовувалися властивості “margin” та “padding”. “Margin” встановлює відстань між зовнішнім краєм елемента та іншими елементами на сторінці. “Padding” – відстань між вмістом елемента та його внутрішнім краєм.

2.3 Експериментальне дослідження ефективності запропонованої методики

Проведення педагогічного експерименту стало ключовим етапом у дослідженні ефективності запропонованих навчальних тем зі створення вебсайтів у середовищі Visual Studio Code. Його мета полягала у виявленні змін у рівні сформованості професійних компетентностей студентів спеціальності «Комп'ютерні науки». Експеримент був організований у межах освітнього

процесу коледжу та охоплював дві навчальні групи: експериментальну та контрольну.

Студенти контрольної групи навчалися за традиційною програмою, яка передбачала переважно теоретичне вивчення HTML і CSS, обмежене виконанням окремих фрагментів коду без створення повноцінного вебпроєкту. Натомість студенти експериментальної групи працювали за практико-орієнтованим підходом, що включав покрокову розробку сайту – від створення структури HTML-документа до стилізації елементів за допомогою CSS, оптимізації графічних ресурсів та реалізації інтерактивної форми зворотного зв'язку.

Навчання в експериментальній групі здійснювалося у середовищі Visual Studio Code, яке було обране як зручний, сучасний та функціональний інструмент для веброзробки. Студенти самостійно створювали проєктну папку, додавали HTML-файл, поступово наповнювали його контентом, структурували сторінку за допомогою семантичних тегів, таких як `<header>`, `<main>`, `<section>`, `<footer>`, та реалізовували навігаційне меню, блоки з текстом, зображеннями, списками і кнопками. Особливу увагу було приділено створенню другої сторінки сайту, яка відкривалася через кнопку «Замовити послугу» і містила форму для введення контактних даних користувача. Це дозволило студентам не лише закріпити знання про HTML-структуру, а й зрозуміти логіку взаємодії елементів на сторінці.

Після завершення етапу створення структури сайту студенти переходили до стилізації за допомогою CSS. Вони створювали окремий файл стилів, підключали його до HTML-документа, працювали з класами, шрифтами, кольорами, відступами, псевдокласами `:hover` і `:focus`, а також адаптували елементи для різних розмірів екранів. Важливим компонентом навчального процесу стала оптимізація зображень, яку студенти здійснювали за допомогою онлайн-ресурсу Squoosh. Це дозволило їм зрозуміти значення ефективного використання графіки для швидкого завантаження сайту та покращення користувацького досвіду.

У процесі навчання здійснювався супровід, надання зворотного зв'язку студентам, виявлення труднощів, з якими вони стикалися. Після завершення практичного етапу було проведено підсумкове оцінювання, яке включало тестування теоретичних знань, аналіз створених сайтів за визначеними критеріями (структура, стилізація, функціональність, оптимізація), а також анкетування студентів щодо їхнього ставлення до навчального процесу, рівня мотивації та самооцінки.

Комплексний аналіз результатів дозволив оцінити ефективність впроваджених тем з HTML і CSS у професійну підготовку. У процесі оцінювання враховувалися результати тестування, які показали, що студенти експериментальної групи продемонстрували значно вищий рівень теоретичної обізнаності щодо синтаксису HTML і CSS, розуміння структури вебдокументів, знання основних тегів, атрибутів, властивостей стилів, а також логіки побудови вебсторінки. Це свідчить про те, що поетапне виконання завдань, робота в середовищі Visual Studio Code та створення реального сайту сприяли глибшому засвоєнню матеріалу.

Окрім тестування, важливим джерелом інформації стали практичні роботи студентів – створені ними вебсайти. Аналіз цих робіт здійснювався за кількома критеріями: логічність і повнота HTML-структури, коректність стилізації елементів, використання класів і шрифтів, адаптивність сторінки, оптимізація зображень, реалізація форми зворотного зв'язку. У більшості випадків студенти експериментальної групи успішно реалізували всі компоненти сайту, дотримуючись технічного завдання. Їхні роботи були структурованими, естетично оформленими, функціональними та відповідали сучасним вимогам до веброзробки. У контрольній групі спостерігалася фрагментарність виконання, відсутність стилізації або її спрощений варіант, а також труднощі з реалізацією інтерактивних елементів.

Для підтвердження ефективності навчання студентів за темами HTML і CSS було проведено порівняльний аналіз результатів двох груп. Середній бал за тестування в експериментальній групі становив 8,7 з 10 можливих, тоді як у

контрольній –лише 6,2. Якість HTML-структури в роботах експериментальної групи оцінена як висока – 90% робіт були побудовані коректно, з дотриманням семантичної розмітки та логіки побудови сторінки. У контрольній групі цей показник становив лише 65%, що свідчить про фрагментарне засвоєння теми.

Ще більш виразною є різниця у використанні CSS для стилізації. Студенти експериментальної групи не лише застосовували стилі, а й реалізовували адаптивність, використовували шрифти з Google Fonts, псевдокласи `:hover` і `:focus`, що забезпечувало сучасний вигляд сайту. У контрольній групі стилізація була частковою, обмеженою базовими властивостями без глибокого опрацювання.

Усі студенти експериментальної групи використовували онлайн-ресурс Squoosh для стиснення графіки, що позитивно вплинуло на швидкість завантаження сайту та загальну продуктивність. У контрольній групі лише частина студентів застосовувала оптимізацію, а більшість – ігнорувала цей етап, що призводило до перевантаження сторінки та зниження її ефективності.

Реалізація форми зворотного зв'язку також стала показником рівня технічної підготовки. У 95% студентів експериментальної групи форма була повністю функціональною, містила поля для введення імені та телефону, а також кнопку для надсилання даних. У контрольній групі лише 40% студентів змогли реалізувати цей функціонал повністю, решта – частково або з помилками в коді.

Рівень самостійності студентів експериментальної групи оцінено як високий. Вони демонстрували здатність самостійно працювати з кодом, приймати рішення щодо структури та дизайну, активно використовували довідкові ресурси, консультувалися з викладачем лише у складних випадках. Це свідчить про розвиток не лише технічних навичок, а й критичного мислення, творчого підходу та професійної відповідальності. У контрольній групі спостерігалася залежність від викладача, труднощі з самостійним виконанням завдань, що негативно впливало на загальний результат.

Мотивація до навчання, за результатами анкетування (зразок анкети – додаток А), також була значно вищою серед студентів експериментальної групи

– 92% учасників відзначили, що навчання було цікавим, практичним і корисним. Вони висловили задоволення від можливості працювати з реальним проєктом, бачити результат своєї роботи, використовувати сучасні інструменти та технології. У контрольній групі лише 58% студентів дали позитивну оцінку навчальному процесу, решта вказували на складність матеріалу, відсутність практичного застосування та недостатню підтримку.

Результати перевірки знань у контрольній та експериментальній групах

На діаграмі (Рис. 2.46) показано середній відсоток отриманих оцінок студентами контрольної групи: початкові – 60,83%, кінцеві – 67,5%.



Рис. 2.46. – Засвоєння знань студентами контрольної групи

Результати засвоєння знань студентами експериментальної групи (Рис. 2.47): початкові оцінки – 65%, кінцеві – 72,5%.

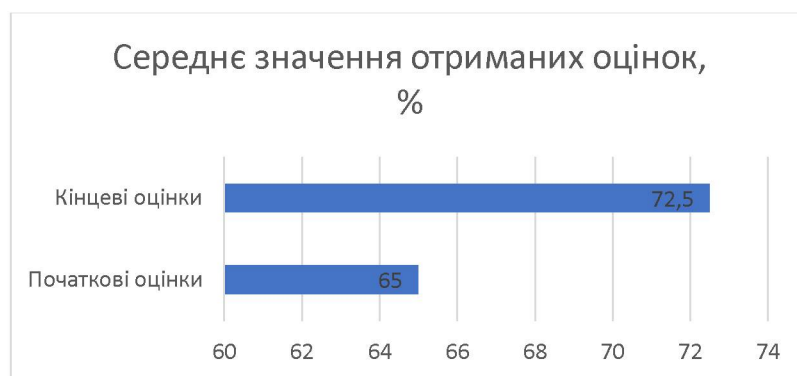


Рис. 2.47. – Засвоєння знань студентами експериментальної групи

Таким чином, аналіз результатів педагогічного експерименту підтверджує, що навчання за темами HTML і CSS у форматі практичної розробки сайту є ефективним засобом формування професійних компетентностей. Воно сприяє

глибшому засвоєнню матеріалу, розвитку самостійності, підвищенню мотивації студентів та формуванню готовності до роботи з реальними проєктами.

Висновок до другого розділу

У другому розділі кваліфікаційної роботи у результаті проведеного аналізу навчального плану та календарно-тематичного планування фахового коледжу встановлено, що дисципліна «WEB-програмування. Інформатика» в змістовому модулі «Креативне програмування» має значний потенціал для розвитку цифрових компетентностей студентів. Теми HTML і CSS займають важливе місце у формуванні базових навичок веброзробки, проте їх практична реалізація потребує оновлення методичних підходів та інтеграції сучасних інструментів.

Розроблена методика навчання створення вебсайтів у середовищі Visual Studio Code базується на практико-орієнтованому підході, який передбачає активне залучення студентів до процесу розробки реального вебресурсу. Даний підхід дозволяє не лише засвоїти теоретичні знання, а й сформувати професійні навички, що відповідають вимогам сучасного IT-ринку.

У межах методики було реалізовано поетапну розробку вебсайту. Студенти ознайомлювались із принципами побудови HTML-структури, семантичними тегами, здійснювали стилізацію вебсторінки за допомогою CSS, опрацьовували селектори, властивості, адаптивність та графічну оптимізацію. Такий розподіл забезпечує поступове засвоєння матеріалу та закріплення навичок у контексті реального завдання.

Проведений педагогічний експеримент підтвердив ефективність запропонованої методики. Студенти експериментальної групи, які працювали з VS Code та виконували завдання на основі розробленого сайту, продемонстрували вищий рівень самостійності, креативності та технічної грамотності. Аналіз результатів показав покращення якості виконаних робіт, глибше розуміння структури вебсторінки та позитивне ставлення до використання сучасних інструментів.

ЗАГАЛЬНІ ВИСНОВКИ

У кваліфікаційній роботі проаналізовано теоретичні основи навчання студентів створенню сайтів у коледжах. Детально розглянуто основні аспекти веброзробки, включаючи використання HTML та CSS у редакторі Visual Studio Code. Проаналізовано основні елементи HTML і CSS, способи їх застосування для створення та форматування вебсторінки, а також можливості сучасних редакторів коду. Особливу увагу приділено Visual Studio Code як ефективному інструменту для реалізації практико-орієнтованого підходу в навчанні.

У межах роботи здійснено аналіз навчального плану та календарно-тематичного планування дисципліни «WEB-програмування. Інформатика» в змістовому модулі «Креативне програмування» та обґрунтовано доцільність оновлення методичних підходів до викладання тем HTML і CSS.

Розроблено методику навчання студентів створенню сайтів в коледжах під час вивчення інформатики. Створено лекційне заняття та лабораторну роботу, що сприяють формуванню професійних навичок студентів. Поетапна розробка вебсайту у середовищі Visual Studio Code дозволила студентам закріпити теоретичні знання через практичну діяльність.

Проведено експериментальну перевірку ефективності запропонованої методики в освітньому процесі. Результати педагогічного експерименту свідчать про доцільність впровадження сучасних цифрових інструментів у навчальний процес та оновлення змісту освітніх програм відповідно до вимог ІТ-галузі.

Запропоновані навчальні матеріали можуть бути використані як методичний ресурс для викладачів інформаційних дисциплін, а також як основа для подальшого вдосконалення освітніх програм з вебпрограмування. Робота демонструє, що поєднання теоретичних знань із практичною розробкою реального вебсайту у сучасному середовищі сприяє формуванню конкурентоспроможних фахівців, здатних ефективно працювати в умовах цифрової економіки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. CSS Синтаксис. [Електронний ресурс]. Режим доступу: URL: https://w3schoolsua.github.io/css/css_syntax.html (дата звернення: 20.04.2023).
2. HTML Reference – A free guide to all HTML elements and attributes. [Електронний ресурс]. Режим доступу: URL: <https://htmlreference.io/> (дата звернення: 20.04.2023).
3. HTML Атрибути. [Електронний ресурс]. Режим доступу: URL: https://w3schoolsua.github.io/html/html_attributes.html#c1&gsc.tab=0 (дата звернення: 20.04.2023).
4. HTML Елементи. [Електронний ресурс]. Режим доступу: URL: https://w3schoolsua.github.io/html/html_elements.html (дата звернення: 20.04.2023).
5. HTML Редактори коду та IDE. [Електронний ресурс]. Режим доступу: URL: https://w3schoolsua.github.io/html/html_editors.html#gsc.tab=0 (дата звернення: 20.04.2023).
6. Meta tags. [Електронний ресурс]. Режим доступу: URL: <https://aboutseo.blog/seo-dictionary/meta-tags/> (дата звернення: 20.04.2023).
7. Безпечне використання сучасних інформаційних технологій / Ящик О. Б., Рак В.І., Лісовий В.В. //The 2nd International scientific and practical conference «Progressive research in the modern world» (November 2-4, 2022) BoScience Publisher, Boston, USA. 2022. 328-334 p.
8. Безпечне користування сучасними інформаційно-комунікативними технологіями: методичні рекомендації. – К.: Україна, 2021. – 72 с.
9. Биков В.Ю., Мадзігон В.М. Руденко В.Д. Яким бути базовому курсу інформатики в загальноосвітніх навчальних закладах//Комп'ютер у школі та сім'ї. – 2001. – №6. – С.3-6.
10. Вбудовані таблиці стилів. [Електронний ресурс]. Режим доступу: URL: <https://studfile.net/preview/1624162/page:3/> (дата звернення: 20.04.2023).

11. Встановлення та налаштування Visual Studio Code. [Електронний ресурс]. Режим доступу: URL: <https://cases.media/article/vstanovlennya-ta-nalashtuvannya-visual-studio-code> (дата звернення: 20.04.2023).
12. Дисертація «використання Веб-орієнтованих технологій у навчанні інформатики в закладах загальної старшої освіти» Національний педагогічний університет ім. Драгоманова – Ворожбит А.В., 2019. – 302с.
13. Етапи створення вебсайтів [Електронний ресурс]. Режим доступу: URL: <https://webtune.com.ua/statti/web-rozrobka/etapy-stvorenniya-veb-sajtiv/> (дата звернення: 20.04.2023).
14. Інформатика: під. Для 10 класу загальноосв. І-74 навч. / Й.Я. Ривкінд [та інш.]. – Київ : Генеза, 2017. – 288 с.
15. Комп'ютерні інформаційні технології в освіті і науці URL:<https://sites.google.com/site/navcalnapraktikakitvoin/lekciie/lekciahttps://sites.google.com/site/navcalnapraktikakitvoin/lekciie/lekcia-veb-tehnologiiie-veb-tehnologiiie>
16. Лещенко М. П., Ястребов М. М. Теорія і практика використання веб-орієнтованих технологій у здоров'язбережувальному навчанні учнів початкових класів. Інформаційні технології і засоби навчання. 2016. № 4(54). С. 53–71. URL:<http://journal.iitta.gov.ua/index.php/itlt/article/view/1416/1067>
17. Мамченко С. Д., Греков В.П. , Коваль Т.І. Інформатика та комп'ютерна техніка (Навчально – К.: ІММБ, 2003. – с. 124)
18. Методика застосування інформаційно-комунікаційних технологій у діяльності педагога професійного навчання (на прикладі професій галузі зв'язку): методичні рекомендації / Л.А. Майборода. – К. : ФО-П Поліщук О.В., 2012. – 104 с.
19. Ю. Туранов, І. Луцик, Я. Замора, Р. Горбатюк, Т. Сорока, Б. Струганець, Ю. Франко Кваліфікаційні роботи : методичні рекомендації до виконання магістерських досліджень студентами інженерно-педагогічного факультету. Тернопіль : Вид – во ТНПУ ім. В. Гнатюка, 2024. 69 с.

ДОДАТКИ

Додаток А

Анкета для оцінювання ефективності навчання студентів за темами HTML і CSS у середовищі Visual Studio Code

Мета: з'ясувати ставлення студентів до навчання за темами HTML і CSS, оцінити мотивацію, зручність використання інструментів та рівень самостійності.

1. Наскільки вам було цікаво працювати над створенням вебсайту?
 - ☐ Дуже цікаво
 - ☐ Помірно цікаво
 - ☐ Нецікаво
2. Чи зрозумілою була структура HTML-документа після пояснення та практики?
 - ☐ Повністю зрозуміло
 - ☐ Частково зрозуміло
 - ☐ Залишилися труднощі
3. Наскільки зручно вам було працювати у середовищі Visual Studio Code?
 - ☐ Дуже зручно
 - ☐ Зручно
 - ☐ Незручно
4. Чи змогли ви самостійно створити структуру сайту без допомоги викладача?
 - ☐ Так
 - ☐ Частково
 - ☐ Ні
5. Наскільки складним для вас було стилізувати сайт за допомогою CSS?
 - ☐ Легко
 - ☐ Помірно складно

☐ Дуже складно

6. Чи використовували ви онлайн-ресурси (наприклад, Squoosh) для оптимізації зображень?

☐ Так

☐ Ні

☐ Не знали про такі ресурси

7. Наскільки корисною була практика створення форми зворотного зв'язку?

☐ Дуже корисна

☐ Корисна

☐ Не корисна

8. Чи відчували ви мотивацію до навчання під час виконання практичних завдань?

☐ Так, високу

☐ Помірну

☐ Низьку

9. Які труднощі виникали у вас під час роботи над сайтом?

(Відкрите запитання)

10. Чи хотіли б ви продовжити навчання у сфері веброзробки?

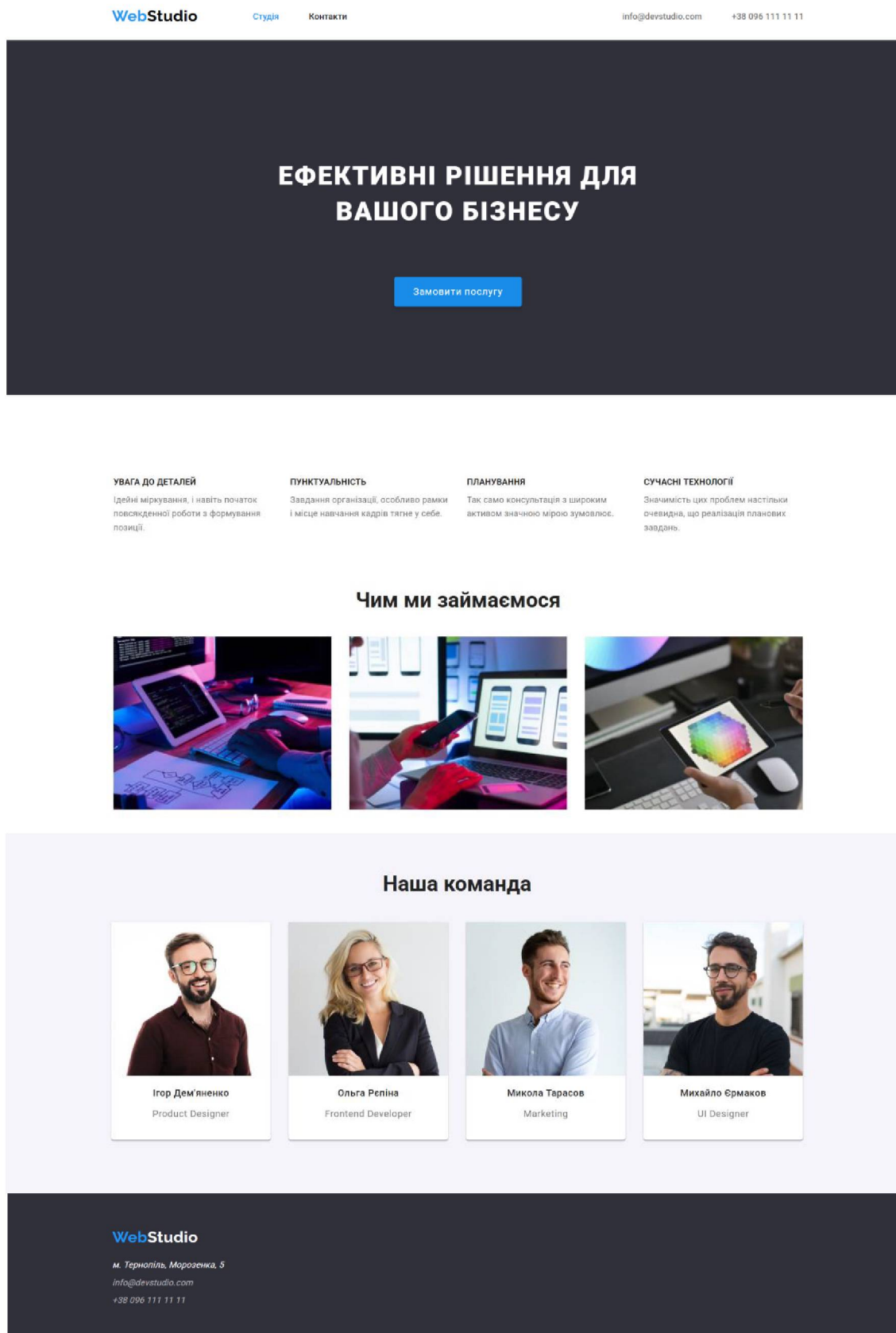
☐ Так

☐ Не впевнений(а)

☐ Ні

Розроблений сайт

url: <file:///C:/Users/user/Desktop/%D0%A1%D0%B0%D0%B9%D1%82/Website/index.html>



Форма

WebStudio

Студія

Контакти

info@devstudio.com

+38 096 111 11 11

Залишіть свої контакти

Ми вам вателефонуємо протягом 30 хвилин

Ім'я

Телефон

Замовити послугу

WebStudio

м. Тернопіль, Морозенка, 5

info@devstudio.com

+38 096 111 11 11



Лекція на тему:

Інтерфейс редактора коду Visual Studio Code. Основи HTML і CSS.

Мета: Знайомство із основними поняттями та можливостями мов розмітки HTML і CSS, необхідними для створення сучасних вебсайтів.

План

1. Інтерфейс редактора коду Visual Studio Code
2. Теги та атрибути мови розмітки HTML
3. Скелет HTML-документа
4. Зображення
5. Структурна розмітка
6. Вступ до CSS
7. Оформлення тексту
8. Властивості шрифту

Інтерфейс VS Code може виглядати складним на перший погляд але насправді є дуже зручним. Він складається з меню розташованого у верхній частині інтерфейсу та містить різноманітні опції, такі як відкриття/збереження файлу, управління розширеннями, довідку та інше (Рис. 2.17). Панель інструментів містить іконки, що надають швидкий доступ до деяких функцій редактора.

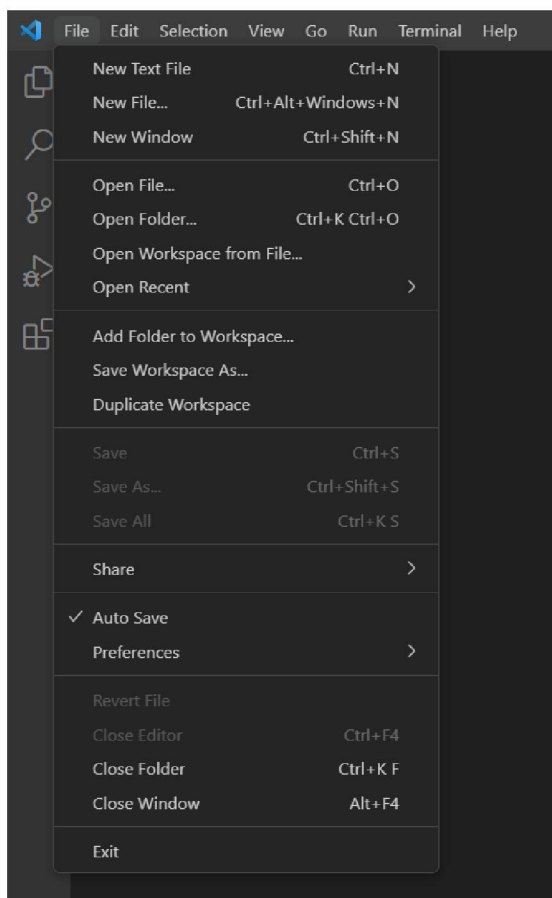


Рис. 2.17. – Меню редактора VS Code

У вікні "Explorer" кожен відкритий файл відображається у своїй вкладці, що дає можливість легко перемикатися між ними, натиснувши на потрібну:

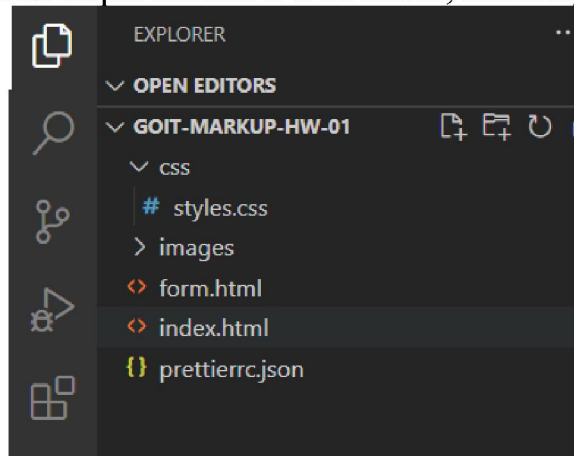


Рис. 2.18. – Вкладки файлів редактора VS Code

Бічна панель (Рис. 2.19) містить багато корисних інструментів та вікон, які допомагають управляти проєктами, файлами та розширеннями. Наприклад, вікно "Search" дозволяє шукати текст у вашому проєкті.



Рис. 2.19. – Бічна панель редактора VS Code

VS Code також має декілька вікон виводу, які використовуються для відображення різноманітних повідомлень та інформації про проєкт. Наприклад, у вікні "Terminal" (Рис. 2.20) можна взаємодіяти з командним рядком, а у вікні "Output" – переглядати вивід програми або іншу інформацію.

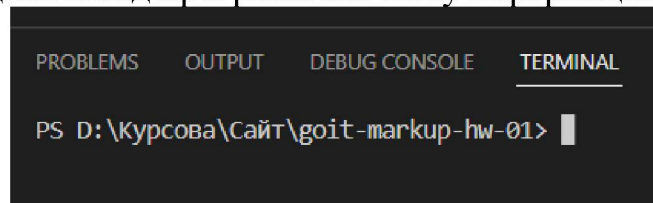


Рис. 2.20. – Вікно виводу "Terminal"

Панель команд дозволяє виконувати різноманітні команди за допомогою гарячих клавіш без використання миші (Рис. 2.21). Щоб викликати її потрібно використати Ctrl+Shift+P в Windows та Linux, або Cmd+Shift+P на macOS.

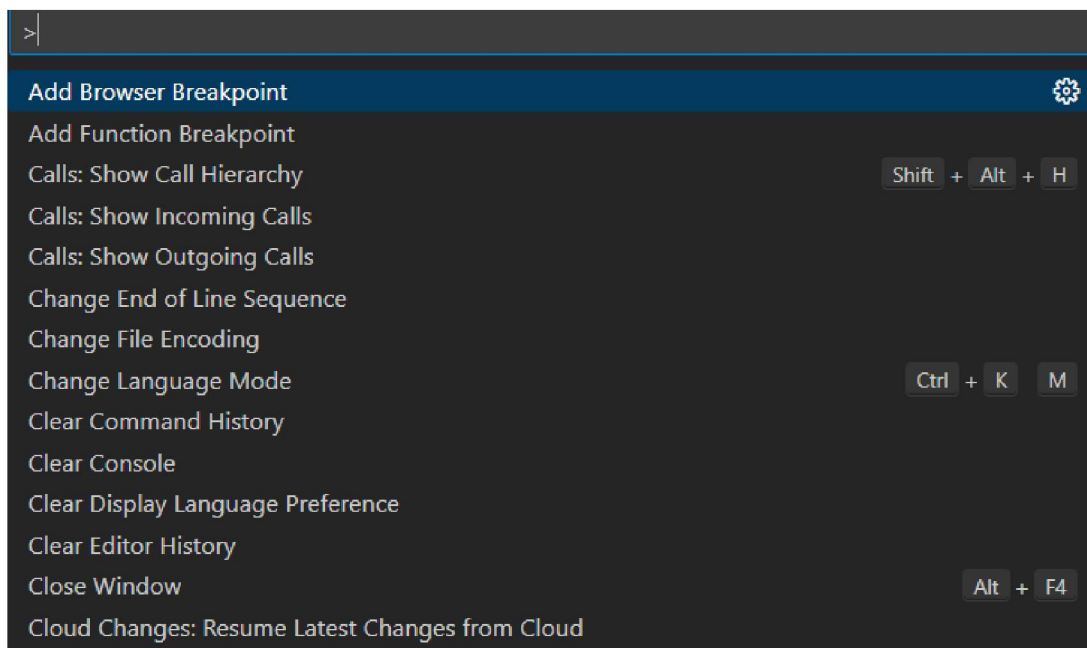


Рис. 2.21. – Панель команд

Налаштування редактора (Рис. 2.22) доступні в декількох місцях, включаючи меню “File” та панель налаштувань. У вкладці “Settings” також можна змінити різноманітні налаштування, такі як розмір шрифту, кольорову схему, функції редактора та інші.

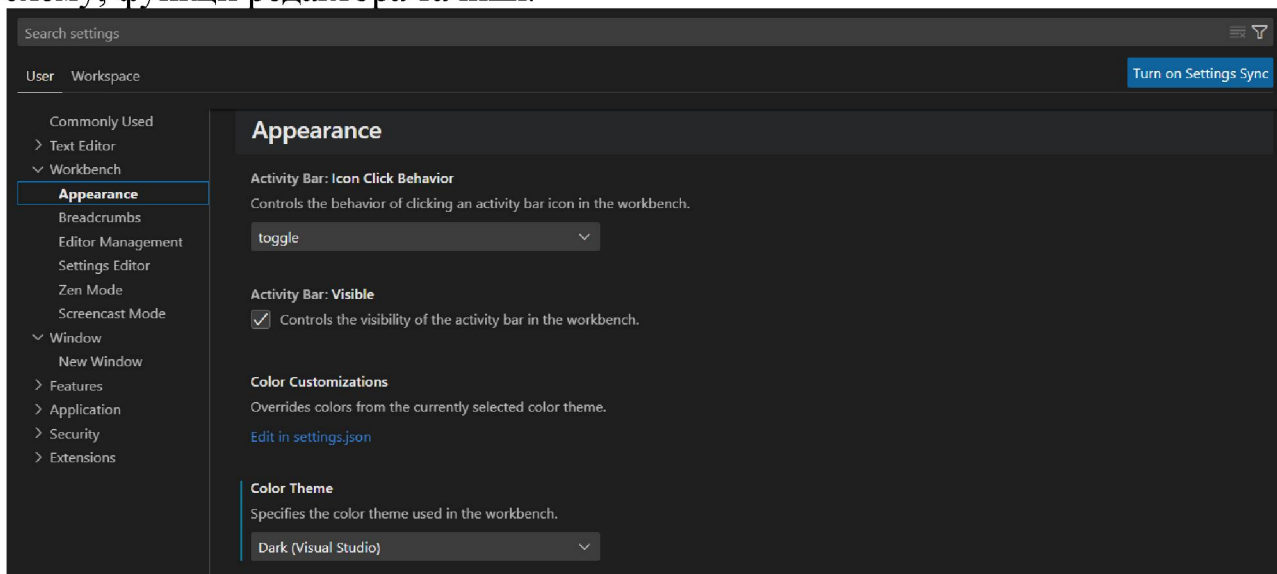


Рис. 2.22. – Налаштування редактора

Редактор має багато корисних функцій, які допомагають поліпшити продуктивність та швидкість написання коду. Це лише деякі з основних елементів інтерфейсу VS Code але їх достатньо для розробки вебсайту та інших проєктів.

Теги та атрибути

Тег – елемент мови розмітки гіпертексту. Це найменші будівельні блоки, з яких складається будь-яка вебсторінка. Кожен тег позначає якусь сутність: заголовок, список, абзац тексту, зображення.

Для виділення тегів серед тексту документа використовуються кутові дужки, в яких вказується ім'я тегу і його атрибути:

```
<ім'я_тегу>...</ім'я_тегу>
```

Відкриваючий тег вказує, де починається елемент, закриваючий – де закінчується. Закриваючий тег утворюється за допомогою додавання знаку “/” перед іменем тегу. Між відкриваючим і закриваючим тегами знаходиться вміст тегу – контент.

Атрибути – це додаткові налаштування тегів, за допомогою яких можна змінювати властивості і поведінку елемента. У кожного тегу є обов'язкові і необов'язкові атрибути, їх може бути декілька або зовсім не бути.

Атрибути записуються всередині відкриваючого тегу, а їх значення розташовуються всередині подвійних лапок. Декілька атрибутів розділяються пробілом.

Розглянемо деякі атрибути тегу `<img>`:

```
<img src=https://picsum.photos/640/480 alt="Картинка"/>
```

`src` – адреса зображення

`alt` – альтернативний текст, який буде показаний, якщо зображення не завантажилося

Скелет HTML-документа

HTML-документ складається з «дерева» тегів:

```
<!DOCTYPE html>
<html lang="en">
  <head></head>
  <body></body>
</html>
```

`<!DOCTYPE>` – це не тег, а обов'язкова інструкція оголошення типу документа. Вона потрібна для того, щоб повідомити браузеру, в якій версії HTML написаний документ. Браузер визначає версію HTML і правильно відображає сторінку. Оголошення типу документа повинно бути найпершим, що бачить браузер під час обробки HTML-документа.

Тег `<html>` – кореневий елемент документа як контейнер, який містить в собі увесь вміст сторінки. Все, що знаходиться за його межами, не сприймається браузером як HTML-код і не обробляється.

Атрибут `lang` вказує, якою мовою написаний текст сторінки.

Тег `<head>` призначений для зберігання службової інформації про сторінку: заголовок, опис, кодування. Вся ця інформація не відображається у вікні браузера, однак, містить дані, які вказують браузеру, яким чином потрібно обробляти сторінку.

Тег `<body>` містить вміст майбутньої вебсторінки. Контент, який повинен відображатися на сторінці, варто розташовувати саме всередині цього тегу.

Заголовок сторінки – текст, розміщений всередині тегу `<title>`, відображається у вкладці браузера. Довжина заголовка повинна бути не більше 60 символів.

Метадані – тег `<meta>` використовується для зберігання інформації, призначеної для браузера і пошукових систем. Цих тегів може бути декілька, тому що, залежно від використаних атрибутів, вони несуть різну інформацію.

Кодування сторінки необхідно вказати для того, щоб браузер коректно відобразив текст. Якщо цього не зробити, або задати невірне кодування, замість символів браузер може відобразити ієрогліфи.

Зображення

Тег `<img>` призначений для розмітки зображень у різних графічних форматах.

- `src="шлях"` – обов'язковий атрибут, вказує адресу зображення. Шлях до зображення може бути абсолютним, або відносним
- `alt="опис"` – обов'язковий атрибут, альтернативний опис
- `width="значення"` і `height="значення"` – задають розміри зображення в пікселях

Абсолютні і відносні шляхи

Вебсайти містять безліч файлів, які розміщують в окремі папки, щоб ними було легше керувати. Для того щоб створити зв'язок між різними файлами, наприклад в HTML-документі підключити зображення або файл стилів, використовуються абсолютні або відносні шляхи, що описують розташування файлу, який підключається.

Абсолютний шлях вказує точне місце розташування файлу в структурі папок на сервері. Абсолютний шлях дозволяє отримати доступ до файлу зі сторонніх ресурсів:

<https://images.pexels.com/photos/583842/pexels-photo-583842.jpeg>

Відносний шлях – описує шлях до ресурсу поточного файлу. Використовується для складання шляхів до зображень, файлів стилів або створення навігації на інші сторінки сайту.

Для того щоб відтворити зображення логотипу з папки `images`, в `index.html` в атрибуті `src` потрібно вказати відносний шлях:

```
<a href=""></a>
```

Символ `“/”` означає перехід на один рівень нижче. Такий шлях браузер розуміє як: «В папці `images`, на одному рівні з поточним файлом `index.html`, взяти файл `logo.png`».

Оптимізація зображень

Зображення складають близько 90% загальної ваги ресурсів вебсайту, тому їх необхідно оптимізувати в першу чергу.

Нижче наведені спеціальні онлайн-сервіси для стиснення зображень:

Squoosh для оптимізації растрової графіки

SVGOMG для оптимізації векторної графіки

Структурна розмітка

Тег `<header>` виділяє вступну частину всієї сторінки, складної секції або статті. Містить логотип, навігацію, вступний текст, складний заголовок.

Тег <footer> – це фінальна частина всієї сторінки, її розділу або статті. Часто містить копірайт, список посилань на соціальні мережі, контактну інформацію тощо.

Тег <main> – основний (унікальний) вміст, що не повторюється на інших сторінках сайту. Може бути тільки один на сторінці.

Смислові розділи

Тег <nav> визначає розділ основної навігації з посиланнями на секції поточної або інших сторінок. Використовується тільки для основної навігації, а не для будь-якої групи посилань в документі.

Тег <article> – незалежна, відокремлювана, і така, що має самостійне значення, частина документа.

Тег section – великий розділ, що об'єднує вміст за смисловим значенням. Не відокремлюється від основного документа. Наприклад секція списку товарів, блок особистої інформації в профілі користувача, розділ контактної інформації.

Тег <div> – універсальний контейнер без семантичного значення. Використовується як блок-обгортка для подальшої стилізації контенту.

Вступ до CSS

CSS – це мова таблиці стилів, що використовується для опису вигляду та форматування вебсторінок, написаних мовою HTML.

Синтаксис CSS складається з селекторів, властивостей та їх значень. Наприклад, клас “.menu-link” – це селектор, “font-size:” – властивість, а “16px;” – значення:

```
.menu-link {font-size: 16px;}
```

Селектори вказують на елементи HTML, до яких будуть застосовані стилі. Вони можуть бути класами або тегами. Властивості вказують, які атрибути елементів повинні змінюватися, такі як колір, розмір, відступи тощо. Значення властивостей вказують, які конкретні параметри повинні бути застосовані до відповідної властивості.

Оформлення тексту

Текст – це основний контент вебсторінок, і для його оформлення існують різноманітні CSS-властивості.

Властивість text-decoration задає різні декоративні підкреслення та інші ефекти тексту:

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

Властивість text-transform управляє регістром символів у тексті, наприклад, зробити всі літери великими або малими:

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

Властивість text-align управляє вирівнюванням текстового вмісту елемента за горизонталлю:

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

Властивість text-indent встановлює величину відступу першого рядка блоку тексту, наприклад абзацу:

```
text-indent: значення | відсотки | inherit
```

Властивість `line-height` встановлює міжрядковий інтервал тексту. За замовчуванням відстань між рядками залежить від виду шрифту і його розмірів, і визначається браузером автоматично:

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

Властивість `letter-spacing` визначає інтервал між символами:

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

Властивість `word-spacing` визначає інтервал між словами. За замовчуванням відстань встановлюється з огляду на тип і вид шрифту:

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

Властивості шрифту

Властивість `font-size` задає розмір шрифту елемента в абсолютних – це пікселі або відносних одиницях виміру:

`.text {font-size: 20px;}`

За замовчуванням розмір шрифту абзаців, який встановлений в таблиці стилів браузера, дорівнює 16px.

Властивість `font-weight` контролює товщину накреслення літер шрифту (Рис. 2.23).



Рис. 2.23. – Товщина накреслення літер шрифту

Властивість `font-style` задає накреслення тексту, наприклад можна зробити його похилим:

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

Властивість `font-family` задає шрифт тексту елемента. Через кому можна перерахувати довільну кількість шрифтів, які браузер, в порядку зліва направо, спробує знайти в системі користувача і застосувати перший знайдений.

Шрифти ділять на кілька основних груп (сімейств): шрифти із зарубками (`serif`), шрифти без зарубок (`sans-serif`), моноширинні (`monospace`), декоративні та рукописні. За замовчуванням, якщо жоден інший шрифт не заданий, браузер використовує Times New Roman з сімейства із зарубками (`serif`).

Подивитися, яким шрифтом відображений текст елемента, можна в інструментах розробника на вкладці `Computed`:

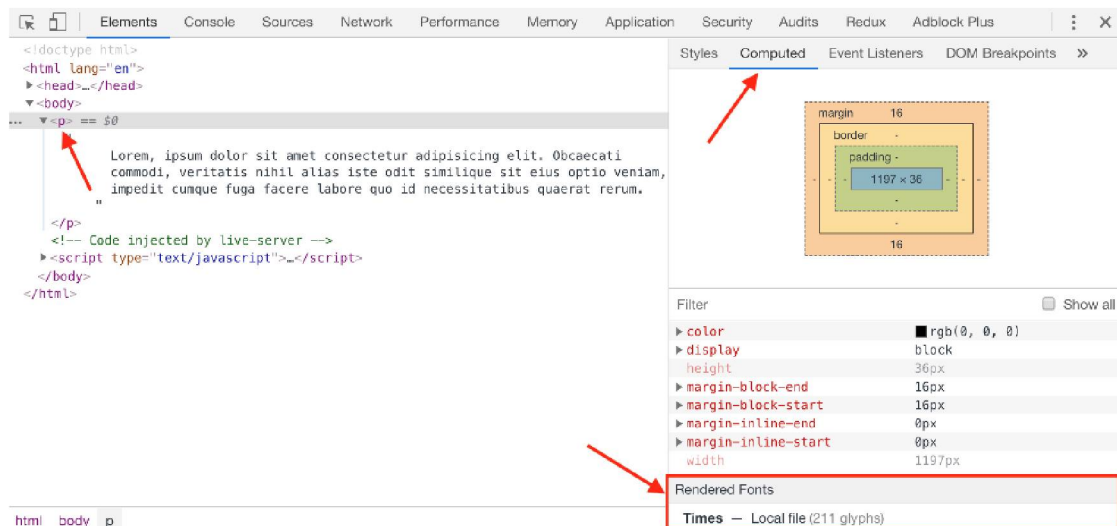


Рис. 2.24. – Інструменти розробника, вкладка Computed

Питання для закріплення лекційного матеріалу:

1. Що означає `<!DOCTYPE>` в HTML і чому він важливий?
2. Яка роль тега `<html>` в структурі HTML-документа?
3. Для чого використовується атрибут `lang` в HTML?
4. Які атрибути є обов'язковими для тега `<img>` і що вони означають?
5. Яка різниця між абсолютними та відносними шляхами до файлів на вебсайті?
6. Що означає властивість `line-height` в CSS і як вона впливає на відображення тексту?
7. Як використовується властивість `font-family` для встановлення шрифту тексту? Які основні групи шрифтів існують?

Домашнє завдання

Встановити редактор Visual Studio Code з офіційного сайту <https://code.visualstudio.com/> для виконання завдань на лабораторній роботі та підключити плагіни розглянуті у лекції I.