

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ТЕХНОЛОГІЇ ІНТЕРНЕТ

Практикум

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітньою програмою «Інформаційно-комунікаційні технології»
спеціальності 172 Електронні комунікації та радіотехніка

Укладачі: С.В. Суліма, М.А. Скулиш

Електронне мережеве навчальне видання

Київ
КПІ ім. ІГОРЯ СІКОРСЬКОГО
2024

УДК 621.39
В19

Укладачі: Суліма Світлана Валеріївна, канд. техн. наук, доц.
Скулиш Марія Анатоліївна, д-р техн. наук, проф

Рецензент Кравчук, С.О., д-р техн. наук, професор,
НН ІТС КПІ ім. Ігоря Сікорського

Відповідальний редактор Суліма С.В., канд. техн. наук, доц.

*Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 1 від 26.09.2024 р.)
за поданням вченої ради навчально-наукового інституту телекомунікаційних систем
(протокол № 11 від 27.11.2023 р.)*

Технології Інтернет [Електронний ресурс] : практикум : навч. посіб. для здобувачів ступеня бакалавра за освіт. програмою «Інформаційно-комунікаційні технології» спец. 172 Електронні комунікації та радіотехніка / КПІ ім. Ігоря Сікорського ; уклад.: С. В. Суліма, М. А. Скулиш. – Електрон. текст. дані (1 файл). – Київ : КПІ ім. Ігоря Сікорського, 2024. – 63 с.

Посібник розроблений для студентів, які навчаються за освітньою програмою "Інформаційно-комунікаційні технології" за спеціальністю 172 "Електронні комунікації та радіотехніка", та містить методичні вказівки до виконання комп'ютерних практичних робіт і призначений для закріплення теоретичних знань з основ розробки веб-додатків та використання сучасних інтернет-технологій. Роботи, наведені в посібнику, дозволяють студентам отримати практичні навички з розробки та адміністрування веб-ресурсів, створення інтерактивних елементів на веб-сторінках, а також інтеграції мультимедійних і зовнішніх сервісів у веб-додатки. Посібник рекомендовано для використання у навчальному процесі студентів бакалаврату та може бути корисним також для інших спеціалістів, які прагнуть підвищити свою кваліфікацію у сфері інформаційно-комунікаційних технологій. ...

...

УДК 621.39

Реєстр. № НП XX/XX-XXX. Обсяг Х,Х авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Берестейський, 37, м. Київ, 03056
<https://kpi.ua>

Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

© КПІ ім. Ігоря Сікорського, 2024

Зміст

Вступ.....	4
Загальні методичні вказівки	5
Вимоги до оформлення звіту з практичної роботи	6
Інструкція з техніки безпеки	7
Практична робота №1 Конструктори сайтів.....	8
I. Теоретичні відомості	8
II. Завдання на практичну роботу.....	13
III. Контрольні питання	14
Практична робота №2 Мова гіпертекстової розмітки HTML. Структура HTMLдокумента. Теги для роботи з текстом. Теги для створення таблиць, малюнків та гіперпосилань.....	15
I. Теоретичні відомості	15
II. Завдання на практичну роботу.....	22
III. Контрольні питання	23
Практична робота №3 Мова гіпертекстової розмітки HTML. Теги для створення форм та елементів управління на веб-сторінках. Уставляння додатків з зовнішніх джерел ...	24
I. Теоретичні відомості	24
II. Завдання на практичну роботу.....	34
III. Контрольні питання	35
Практична робота №4 Стильове оформлення сторінок	36
I. Теоретичні відомості	36
II. Завдання на практичну роботу.....	42
III. Контрольні питання	44
Практична робота №5 Підключення бази даних до сайту	45
I. Теоретичні відомості	45
II. Завдання на практичну роботу.....	48
III. Контрольні питання	49
Практична робота №6 Мова програмування JavaScript	50
I. Теоретичні відомості	50
II. Завдання на практичну роботу.....	58
III. Контрольні питання	59
Додатки	60
Перелік літератури	63

Вступ

Сучасні вимоги до інженерів з телекомунікацій ґрунтуються на глибоких науково-технічних знаннях та вміннях розв'язувати практичні задачі, в тому числі задачі щодо створення та роботи з веб-додатками, а саме методологій розробки таких прикладних програм, написання програмного коду програмного продукту, роботи з засобами управління проєктами тощо.

Метою методичних вказівок є надання допомоги студентам в отриманні практичних навичок роботи з системами керування проєктами, контролю версій та тестування. Виконання практичних робіт забезпечить закріплення теоретичного та лекційного матеріалу.

Під час виконання практичних робіт студенти отримають навички щодо:

- підходів до проектування веб-додатків;
- організації процесу командної розробки програмного продукту;
- тестування веб-додатку на відповідність вимогам замовника.

Загальні методичні вказівки

1. У кожній практичній роботі визначено: мету роботи, рекомендації з підготовки до роботи, програму і порядок її виконання.

2. Напередодні кожної практичної роботи необхідно:

- вивчити теоретичний матеріал зазначений у розділі "Підготовка до практичної роботи";
- усвідомити мету, зміст і порядок виконання;
- у лабораторії перевірити наявне практичне устаткування.

3. До виконання практичної роботи допускаються тільки підготовлені студенти після тестування (усного чи письмового), що проводиться викладачем перед початком виконання роботи.

Практичні роботи виконуються самостійно кожним студентом.

Звіт з роботи акуратно оформлюється і подається під час захисту практичної роботи. Схеми, зображені в звіті, мають відповідати вимогам стандартів.

4. Студенти, відсутні на заняттях, виконують роботу у час, погоджений з викладачем і інженером лабораторії після тестування.

5. Перед початком робіт кожному студенту необхідно вивчити правила техніки безпеки, здати залік, за що розписатися в журналі.

Вимоги до оформлення звіту з практичної роботи

Звіт оформлюється на аркушах формату А4 і повинен містити:

1. Титульну сторінку з номером та назвою практичної роботи, прізвищем студента, номером групи.
2. Особливості завдання.
3. Опис виконаного завдання.
4. Висновки по роботі.

Інструкція з техніки безпеки

1. Вимоги з техніки безпеки перед початком роботи
 - 1.1 Провести огляд з зовні електророзеток, шнурів, вилок підключення до мережі живлення та заземлення (занулення).
 - 1.2 Забороняється працювати на несправному устаткуванні.
 - 1.3 За необхідності отримати додаткове устаткування у викладача та перевірити його справність.
2. Вимоги з техніки безпеки під час виконання робіт
 - 2.1 Необхідно виконувати лише ту роботу, з якої був проведений інструктаж, забороняється передоручати свою роботу іншим особам.
 - 2.2 Забороняється:
 - експлуатація кабелів та проводів з пошкодженою ізоляцією або такою, що втратила захисні властивості за час експлуатації;
 - залишати під напругою кабелі та проводи з неізольованими провідниками;
 - застосовувати саморобні подовжувачі, що не відповідають вимогам ПВЕ для переносних електропроводників;
 - користуватися пошкодженими розетками, розгалужувальними та з'єднувальними коробками, вимикачами та іншими електровиробами, а також лампами, скло яких має слід затемнення або випинання;
 - використовувати електроустаткування та прилади в умовах, що не відповідають інструкції з експлуатації підприємств-виробників;
 - залишати пристрої, що працюють без нагляду на тривалий час;
 - переносити пристрої, що підключені до електромережі;
 - забороняється самостійно ремонтувати апаратуру;
 - класти будь-які предмети на апаратуру комп'ютера, напої на клавіатуру або поруч з нею - це може вивести їх з ладу.
3. Вимоги з техніки безпеки після закінчення роботи
 - 3.1. Зберегти необхідні файли на жорсткий диск комп'ютера або на переносний носій.
 - 3.2 Виключити комп'ютер.
 - 3.3. Виключити додаткове устаткування та віддати його викладачу.
 - 3.4 Прибрати робоче місце.

Практична робота №1

Конструктори сайтів

I. Теоретичні відомості

З появою і швидким розвитком Інтернет перед користувачами відкрилося багато нових можливостей, зокрема, можливість спілкування. Тепер можна обговорювати різні теми на форумах і отримувати цінні поради, розповідати про себе у блозі, знаходити давніх друзів за допомогою соціальних мереж і багато іншого.

Значно ширше поле для діяльності відкриває для користувачів наявність власного сайту. Це можливість заявити про себе, свої захоплення, роботу, компанію. Але створення навіть простого за дизайном чи функціоналом сайту потребує базових знань мов HTML, CSS та JavaScript, навичок обробки зображень, уставляння мультимедійних об'єктів, розміщення веб-документів на серверах Інтернету.

В міру зростання інтересу до створення сайтів, з'явилися й особливі платформи – конструктори сайтів. З їх допомогою користувач будь якого рівня обізнаності може створити власний сайт-візитку, багатосторінковий сайт для компанії або онлайн-магазин.

Конструктор сайтів дозволяє сформувати і об'єднати веб-сторінки в цілісну структуру сайту, а також керувати ними, не володіючи спеціальними технічними знаннями і навичками. Створений в конструкторі ресурс розміщується на хмарі - віддаленому сервері-хостингу, збереження і працездатність якого підтримується командою адміністраторів конструктора без втручання користувача.

Конструктори мають дружній інтерфейс, що буде зрозумілим для непідготовленого користувача. Редагування сторінок, зовнішнього вигляду дизайну і загальне налаштування відбувається в онлайн-режимі за допомогою зручної панелі управління.

Переваги конструкторів сайтів

Для звичайного користувача, який не має особливих знань і навичок, конструктор є одним з кращих варіантів швидко і якісно створити власний інформаційний ресурс в Інтернеті, розпочати онлайн бізнес або презентувати власні досягнення.

Швидкість створення. Користувачі «конструюють» дизайн на основі блоків або обирають вже готові шаблони. Створити сайт можна за кілька годин.

Ціна. Вартість створення сайту за допомогою конструктора значно нижча, ніж звертатися до професіональних веб-фахівців, чия робота оцінюється погодинно і може коштувати чимало.

Простота. Жодних особливих знань для роботи в конструкторі не потрібно. В основному функціоналі можна розібратися за допомогою зрозумілих інструкцій.

Доступність. Конкуренція між розробниками конструкторів призвела до того, що значна частина базових рішень надається безкоштовно. Платними залишаються особливі функції і додаткові можливості.

Наявність шаблонів. Користувачу, що створює сайт самостійно, надається можливість вибрати один із готових дизайнів, який надається до редагування: зміна колірної схеми, гарнітури та розміру шрифтів, додавання або видалення інформаційних блоків тощо.

Недоліки конструкторів сайтів

Обмеженість. Конструктор надає можливості створювати сайт з вже готових блоків, і зазвичай, індивідуальні задуми реалізувати важко.

Продуктивність і швидкість роботи. Ресурси, що є невеликими за розміром та складністю працюють доволі швидко. Але, зі збільшенням потужності сайту, його функціонування помітно пригальмовується, що підштовхує власника обрати інший дорожчий тариф.

Зростання абонентської плати. Для простого лендингу або візитки можна обрати безкоштовний або мінімальний тариф. Але із збільшенням функціоналу доведеться змінювати тариф на дорожчий.

Складнощі інтеграції сервісів та систем. Багато сучасних конструкторів намагаються враховувати останні тренди, але існує ризик, що певні сервіси або платформи не будуть підтримуватися.

Залежність від розробників сервісу. Працездатність та доступність сайту залежить від того, як працює команда розробників конструктора. До цього відноситься технічна підтримка, сервісне обслуговування, потужності хостингової хмари, надійність серверів та інше. Також, ризик, що платформа конструктора буде закрита, існує завжди і тоді всі проблеми власник сайту має вирішувати самостійно.

Конструкторів для створення сайтів існує значна кількість, вони мають власні особливості, переваги, тому перед вибором варто ознайомитися з їх функціоналом та оцінити зручність роботи.

Конструктор сайтів Wix

Wix - популярний конструктор, що орієнтується, в першу чергу, на потреби початківців користувачів з нульовими знаннями щодо створення сайтів. Ідеально підходить для створення яскравих за формою і змістом візиток. Сервіс поставляється зі зручним візуальним редактором, в якому більшість дій виконується за допомогою мишки.

Інтерфейс конструктора сайтів Wix

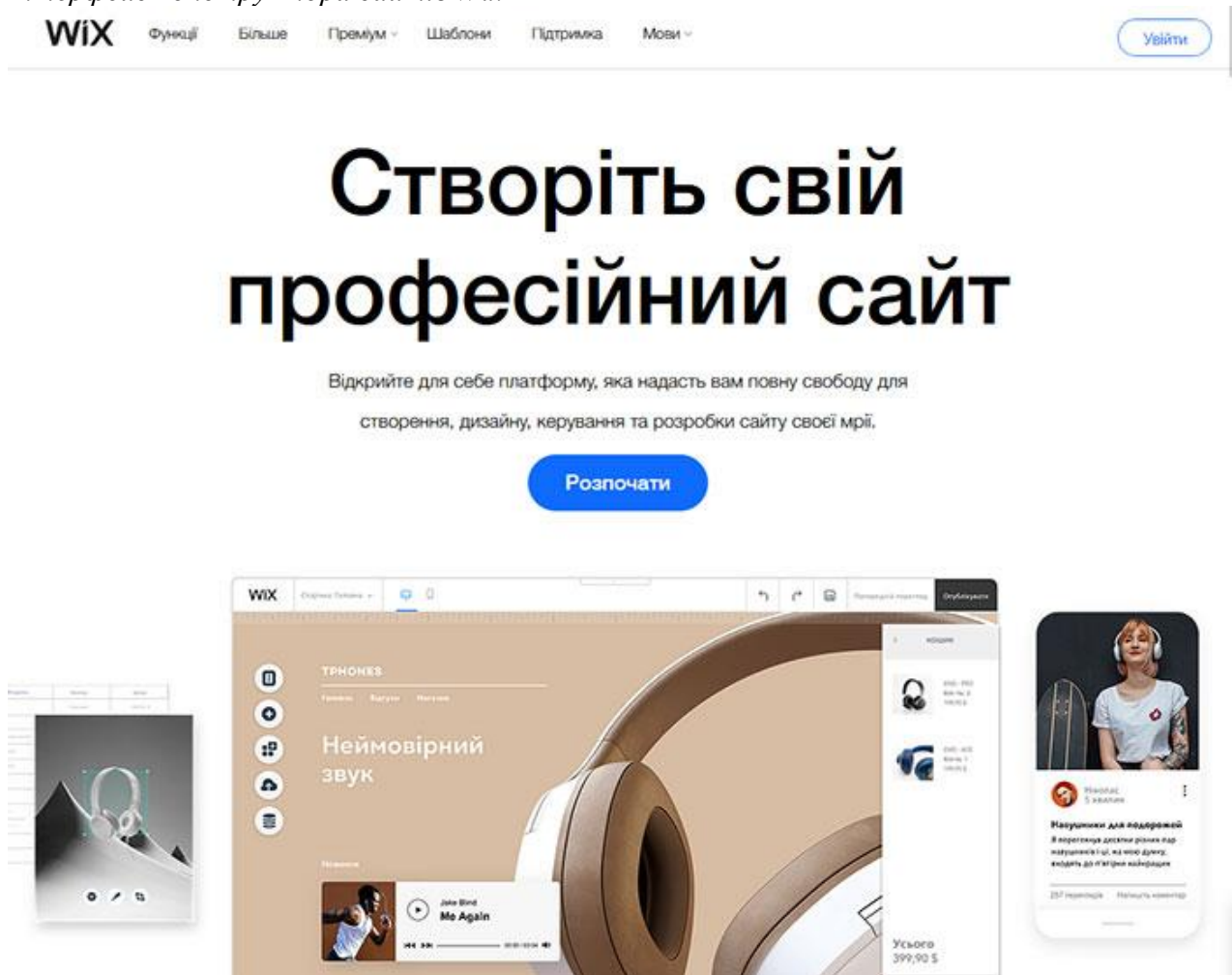


Рисунок 1.1

Конструктор надає надвелику збірку безкоштовних шаблонів, які розподілені за тематичними категоріями, що спрощує вибір та визначає сферу їх застосування. Хоча в процесі роботи над сайтом поміняти дизайн на інший не вийде, проте, є значні можливості щодо його зміни.

Панель управління зручна, функціонально добре продумана, реалізовано додавання віджетів, компонентів і різноманітних налаштувань. Серед цікавих можливостей: додавання відео на фон сайту, широкоформатні лендинги, геометричні форми, іконки, ефекти паралакса, анімація і багато іншого. Є один з кращих редакторів зображень міні-Photoshop. Перед публікацією можна телефону обробити фото (кадрування, розтягнення, зміна масштабу, накладення ефектів, корекція кольору) і розмістити у потрібному місці на сторінці.

У є вбудований магазин додатків, здатний посилити існуючий функціонал. Можна прикріпити форум, під'єднати онлайн платежі, встановити інтерактивного чат-бота (live-chat), додати можливість виставлення рахунків, розгорнуто працювати з налаштуваннями SEO та багато іншого. Асортимент додатків великий і постійно зростає.

Wix надає широкі можливості для творчих людей, які бажають просунути себе або свій бізнес: створення креативних візиток, портфоліо художника / фотографа / дизайнера, промо-сторінок, магазинів на невелику кількість товарів, ефектних лендингів, блогів.

Конструктор сайтів SITE123

SITE123 – сервіс, що призначений для професійних користувачів і початківців, які створюють клієнтські сайти. Конструктор пропонує інший підхід до формування сторінок ніж Wix. Замість віджетів використовуються модулі - готові блоки, що створені під конкретні завдання. Будь-який з модулів має кілька варіантів оформлення, які піддаються додатковому налаштуванню.

Інтерфейс панелі управління зручний і зрозумілий. Всі зміни видно в реальному часі, хоча відсутня можливість безпосередньо виділити елемент на сторінці і почати його редагувати. Зміна сторінок проходить централізовано по заданих напрямних - через налаштування в панелі управління для кожного конкретного модуля.

Інтерфейс конструктора сайтів SITE123

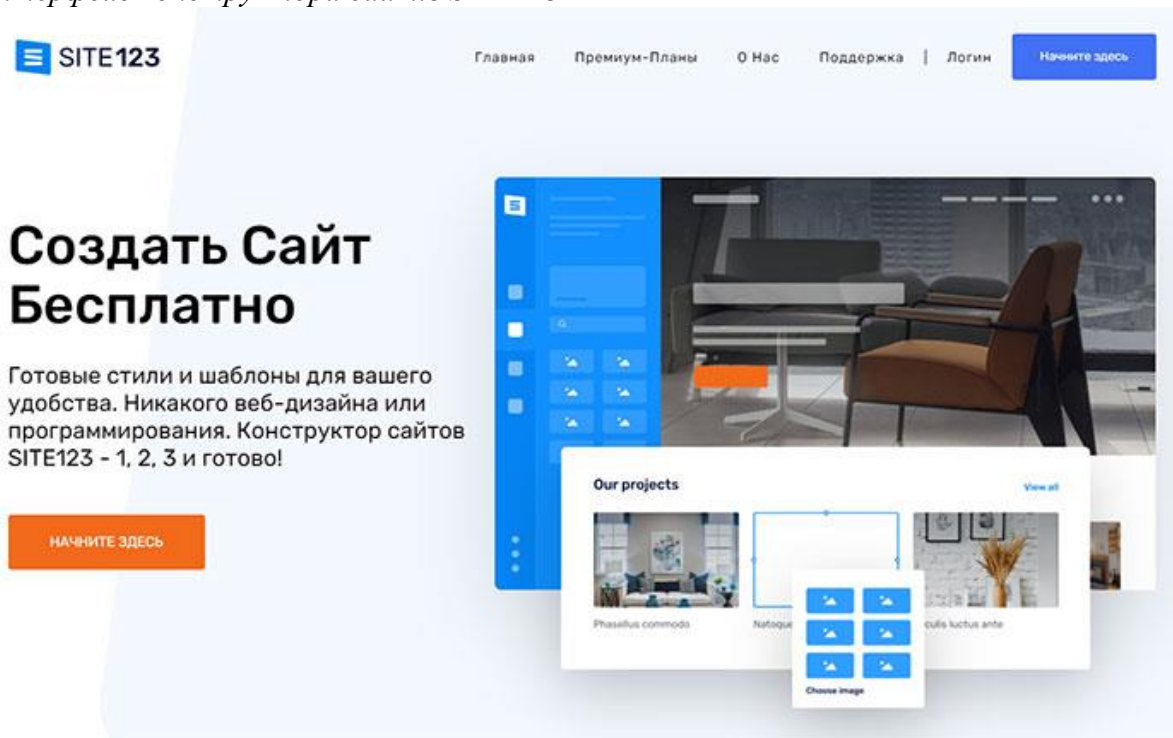


Рисунок 1.2

Конструктор надає до використання шаблони, які мають адаптивний дизайн і сучасно виглядають. Вони легко налаштовуються: 10 варіантів структур, гнучка робота зі схемами шрифтів і кольорів, тип сайту (одно- або багатосторінковий), фони. Використовуючи даний арсенал можна швидко отримати унікальний дизайн.

SITE123 добре підходить для створення магазину з невеликою кількістю товару. Присутня можливість налаштування оплати (PayPal), способів доставки, вибору валюти, можливість впровадити кілька мов на сайт, знижки та інше. Надано збірку плагінів, які допоможуть просунути сайт, зібрати клієнтську базу, аналізувати статистику, інтегрувати сервіси соціальних мереж і багато іншого.

Конструктор сайтів Tilda

Tilda - конструктор для тих, хто уважно ставиться до презентації контенту, щоб подати матеріали вигідно і красиво. Інноваційний блоковий механізм редагування надає можливості швидко і легко зібрати сторінку з готових блоків, спроектованих професіоналами. Бібліотека містить понад 400

блоків і постійно поповнюється. Розробники створюють нові, сучасні елементи відповідно до трендів веб-дизайну. В кожному блоці є гнучкі налаштування для індивідуального дизайну.

Інтерфейс конструктора сайтів Tilda

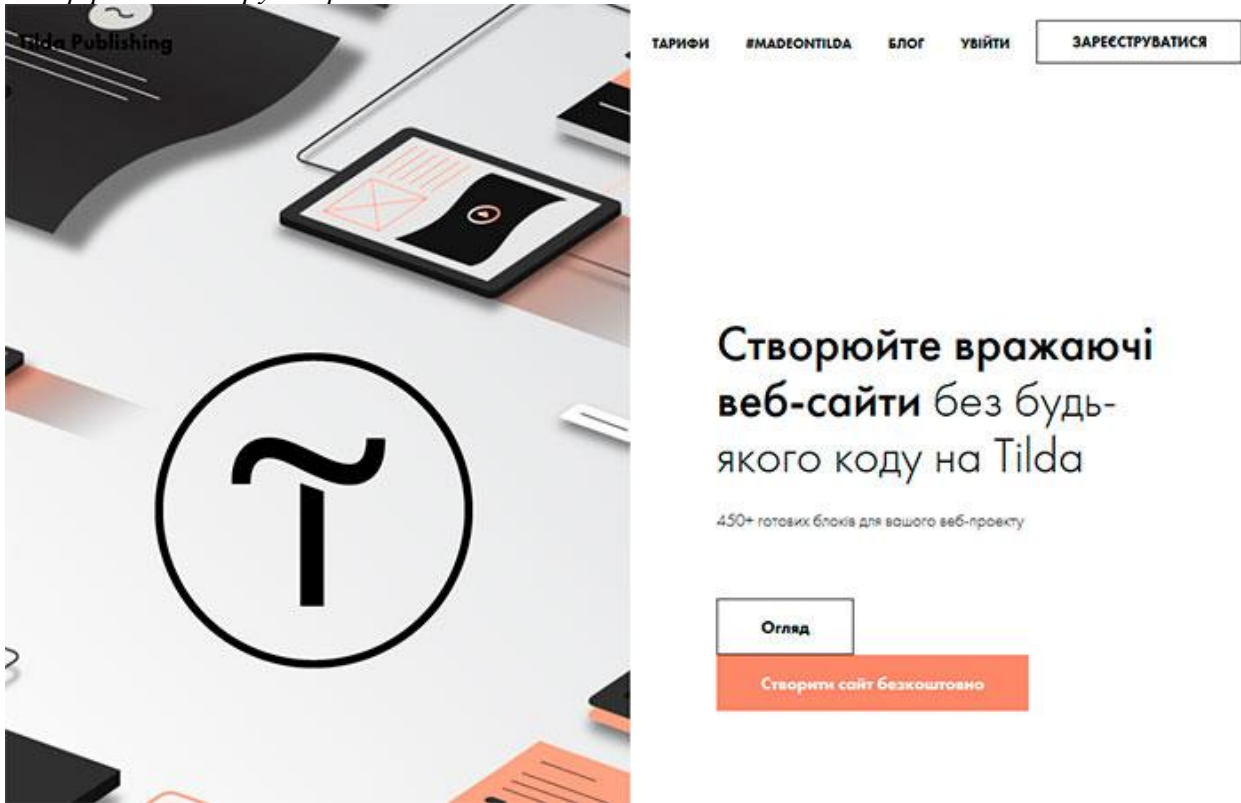


Рисунок 1.3

Надано велику збірку шаблонів готових сторінок: лендинги, тематичні сайти, магазини, промо-сторінки, блоги та інше. Увагу приділено до типографіки: гармонійні пропорції, зручна ширина рядків, правильний інтерліньяж, оптимальний розмір шрифту, комбінації заголовків.

Широкі налаштування в подачі візуального контенту: повноекранні фотографії, готові фотогалереї, комбінації фотографій і тексту. Вбудований редактор зображень надає можливості додавати до зображень написи, ефекти, тонування, змінити насиченість, яскравість, розмір і пропорції.

Представлено колекцію безкоштовних іконок та бібліотеку безкоштовних зображень. Інструменти роботи з мультимедіа (відео та аудіо контент) сприяють пошквалюванню сторінок: фонове відео (YouTube, MP4, WEBM), вбудовування відео з YouTube і Vimeo, готові поєднання відео та тексту, інтегровані сервіси SoundCloud і Soub, уставка відео та аудіо плеєра через блок html-embed.

Конструктор надає можливості безкоштовного користування, хоча там присутні обмеження: сайт міститиме не більше 50 сторінок, домен від конструктора, обмежена кількість блоків і граничний обсяг завантажених файлів в 50 Мб. Для повноцінної роботи варто скористатися платними тарифними планами.

Конструктор сайтів Ucoz

uCoz - це сервіс створення та обслуговування сайтів, що має велику кількість переваг та привертає увагу користувачів завдяки своїм характеристикам. Платформа пропонує потужний функціонал, має доступні тарифні плани з багатьма бонусами. Вона – універсальна та надає змогу використовувати багато можливостей абсолютно безкоштовно.

Інтерфейс конструктора сайтів uCoz

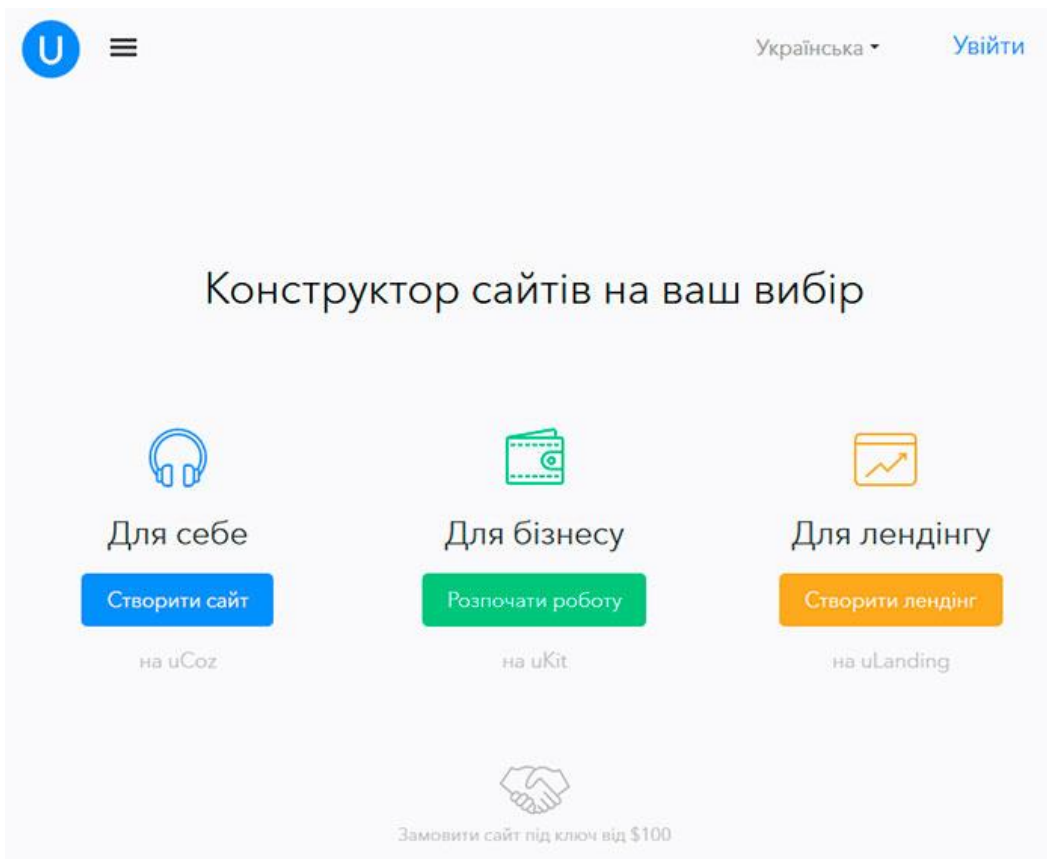


Рисунок 1.4

Єдиний аспект, що може стати перешкодою для початківців – це певна складність оволодіння всіма функціями сервісу. Професіональні розробники знайдуть тут все те, що зазвичай бракує іншим конструкторам сайтів.

Якщо розкрити весь потенціал системи, то можна створити будь-який тип сайту. Завдяки універсальності сервісу, не буде потреби у додаткових інструментах для створення якісного та функціонального ресурсу, що забезпечить суттєву економію часу.

Конструктор сайтів Weblium

Weblium - має естетичну панель керування. Робочий простір ділиться на особистий кабінет зі списком створених проектів і візуальний редактор. Майже вся функціональність сайту вибудована навколо різноманітності готових секцій і налаштування їх компонок. Інших параметрів мінімум - тільки найнеобхідніше, весь фокус сконцентрований на можливостях візуального редактора високого рівня якості.

Інтерфейс конструктора сайтів Weblium

Weblium – розумний конструктор сайтів

Створіть швидкий та ефективний сайт для
вашого бізнесу!

Створити сайт безкоштовно

Замовити сайт у студії

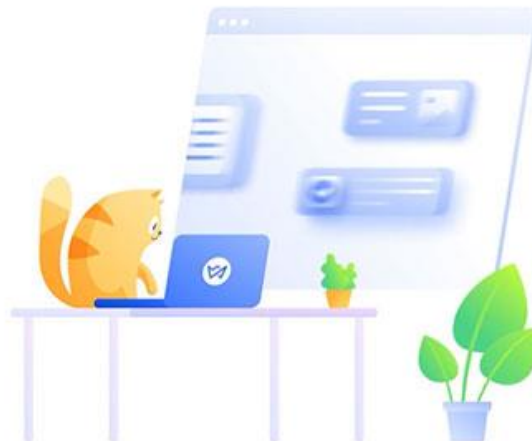


Рисунок 1.5

Однією із особливостей Weblium є вбудований штучний інтелект. Він спроможний підказувати найбільш вдале розміщення блоків, допомагати вибирати кольори, шрифти, правити дрібні недоліки заради досягнення оптимального результату. Користувач обирає секцію, блок, колір, шрифт з багатьох варіантів.

Змінити дизайн можна в один клік. Система супроводжує робочий процес підказками у форматі чек-листа, що є корисним для початківців - вони нічого не пропустять, виконуючи інструкції покроково.

II. Завдання на практичну роботу

1. Для виконання практичної роботи студент створює сайт відповідної тематики, що містить 3-4 змістовних сторінок та сторінку автора сайту. У додатку А наведено перелік тем, які можна використати, або можна реалізувати власну ідею.
2. Ознайомитися з наданими теоретичними відомостями щодо безкоштовних конструкторів.
3. Визначитися щодо тематики та типу сайту. Під час створення сайту робити скрінні відповідних етапів.
4. Обрати онлайн-конструктор для створення сайтів. Зареєструвати акаунт і ознайомитися з можливостями редактора. Зберегти ідентифікатори доступу, зокрема поштову адресу реєстранта, оскільки на неї надходитимуть листи зі створеного сайту.
5. Відповідно до теми сайту віднайти доречний шаблон. Відкрити шаблон для редагування. Уважно дослідити можливості адміністративної панелі, наявні інструменти та їх функції.
6. Відповідно до теми сформулювати інформацію: тексти і зображення. Визначитися з переліком розділів, колірною схемою, стилем подання інформації для різних типів пристроїв, наявністю мультимедійних об'єктів та зворотнього зв'язку.
7. Користуючись інструментами адміністративної панелі або прямим редагуванням на сторінці, змінити назви розділів/підрозділів, додати/видалити відповідні сторінки. Наповнити сторінки сайту відповідною інформацією. Мова сайту - українська! Оздобити текст доречними зображеннями та мультимедійним контентом (відео, аудіо об'єкти, карти).
8. Обов'язковим є наявність авторської сторінки, де розміщується фотографія чи аватар студента, коротка інформація (Ім'я, нік, е-мейл тощо), карта із зазначенням домівки чи іншого місця.
9. Після завершення редагування зберегти зміни та отримати доменну адресу для опублікування сайту. Опублікувати сайт в Інтернеті і переглянути результати через браузер. За потребою внести зміни у сайт.

10. По результатах роботи оформити звіт. Звіт повинен включати доменну адресу, назву і короткий опис створеного сайту, скріншоти послідовності створення сайту з коротким описом кожного з етапів, у висновку оцінити зручність використання конструктора і адміністративної панелі.

III. Контрольні питання

1. Перелічити популярні конструктори сайтів і зазначити сервіси, які надають можливості безкоштовного користування.
2. Вказати особливості конструкторів, зокрема можливості реалізації індивідуальних змін.
3. Навести перелік позитивних характеристик обраного конструктора, а також відмітити обмеження стосовно певних функціональностей.
4. Яким чином можна отримати хостинг та доменну адресу від конструктора.
5. Назвати основні причини, за якими користувач може обрати створення сайту за допомогою конструктора.

Практична робота №2

Мова гіпертекстової розмітки HTML. Структура HTML-документа. Теги для роботи з текстом. Теги для створення таблиць, малюнків та гіперпосилань

I. Теоретичні відомості

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

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

Розмітка означає, що код пишеться з допомогою ключових слів.

Гіпертекст означає, що він використовує HTTP як частину Інтернет.

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

Загальний синтаксис написання тегів:

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

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

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

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

Структура HTML-документа:

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

Наведемо призначення деяких основних та часто використовуваних тегів HTML.

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

<html> ... </html> – вказують на те, що веб-сторінка створена засобами HTML.

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

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

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

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

`<p> ... </p>` – все, що знаходиться між ними, сприймається як один абзац.

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

`_{...}` – перетворює текст між тегами у нижній індекс.

`^{...}` – перетворює текст між тегами у верхній індекс.

`<!-- ... -->` – призначений для коментарів, ігнорується браузером і не відображається на веб-сторінці.

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

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

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

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

`<i> ... </i>` – відображає текст курсивом, призначений для зміни вигляду тексту.

` ... ` – відображає текст курсивом, призначений для акцентування тексту.

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

` ... ` – відображає текст напівжирним, без акценту на його важливість.

` ... ` – відображає текст напівжирним, призначений для акцентування тексту.

`<small> ... </small>` – зменшує розмір шрифту на одиницю по відношенню до звичайного тексту.

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

` ... ` – відображає текст перекресленим, використовується для виділення тексту, який був видалений в новій версії документа.

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

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

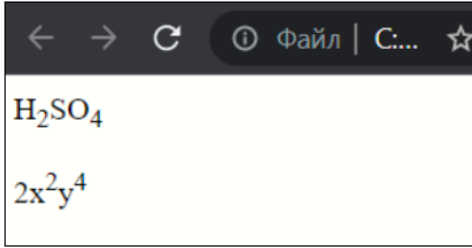
`<mark> ... </mark>` – виділяє колір фону тексту жовтим, призначений для виділення тексту в довідкових цілях.

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

`<wbr>` (або `­`) – вказує браузеру місце, де можна зробити перенесення слова в тексті.

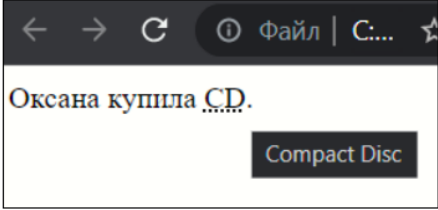
Різниця в тому, що `­` додає при переносі дефіс, а `<wbr>` – ні.

Наприклад:

Код	Результат
<pre><p> H<sub>2</sub>SO<sub>4</sub> </p> <p> 2x<sup>2</sup>y<sup>4</sup> </p></pre>	

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

Наприклад:

Код	Результат
<pre><p> Оксана купила <abbr title="Compact Disc"> CD </abbr>. </p></pre>	

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

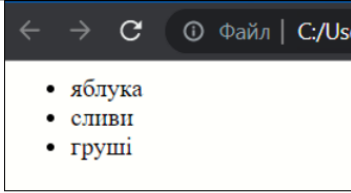
Ім'я	Код	Вид	Опис
©	©	©	знак копірайт
®	®	®	знак зареєстрованої торгової марки
<	<	<	знак «менше»
>	>	>	знак «більше»
&xi	ξ	ξ	грецька мала буква ксі

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

Нелінійний список. Синтаксис:

```
<ul>
  <li> перший елемент списку </li>
  <li> другий елемент списку </li>
  <li> третій елемент списку </li>
  ...
</ul>
```

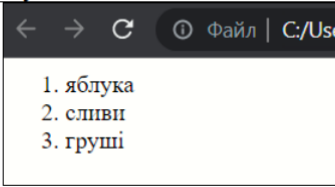
Наприклад: _____

Код	Результат
<pre> яблука сливи груші </pre>	

Нумерований список. Синтаксис:

```
<ol>
  <li> перший елемент списку </li>
  <li> другий елемент списку </li>
  <li> третій елемент списку </li>
  ...
</ol>
```

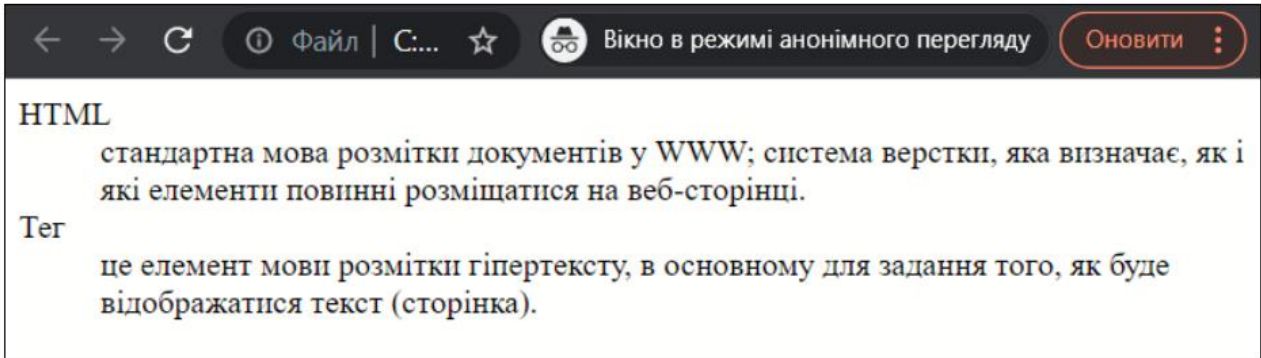
Наприклад: _____

Код	Результат
<pre> яблука сливи груші </pre>	

Списки визначень використовуються для опису визначень і термінів. Синтаксис:

```
<dl>
  <dt>Термін 1</dt>
    <dd>Визначення першого терміну</dd>
  <dt>Термін 2</dt>
    <dd>Визначення другого терміну</dd>
  ...
</dl>
```

Наприклад:

Код
<pre><dl> <dt>HTML</dt> 10 <dd> стандартна мова розмітки документів у WWW; система верстки, яка визначає, як і які елементи повинні розміщатися на веб-сторінці.</dd> <dt>Тег</dt> <dd> це елемент мови розмітки гіпертексту, в основному для задання того, як буде відображатися текст (сторінка).</dd> </dl></pre>
HTML


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

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

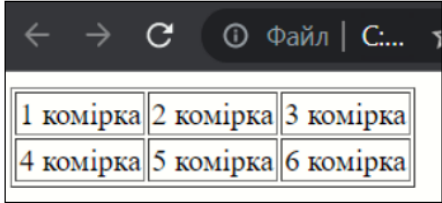
Таблиця в HTML задається з допомогою парного тегу `<table> ... </table>` та повинна витримувати певну структуру. Додатково для таблиці тег абзацу писати не потрібно. Структура таблиці в HTML:

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

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

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

Наприклад:

Код	Результат
<pre><table border="1"> <tr> <td>1 комірка</td> <td>2 комірка</td> <td>3 комірка</td> </tr> <tr> <td>4 комірка</td> <td>5 комірка</td> <td>6 комірка</td> </tr> </table></pre>	

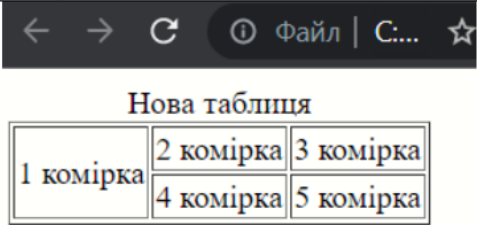
Для того, щоб над таблицею відображався заголовок слід написати тег `<caption> ... </caption>` відразу після `<table>`. Додатково `<caption> ... </caption>` огортати тегом абзацу не потрібно.

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

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

`colspan` – об'єднує комірки по горизонталі.

Наприклад:

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

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

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

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

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

Зображення (фото, картинка, схеми, рисунки) на веб-сторінках розміщують з допомогою тегу . Обов'язковими атрибутами цього тегу є наступні:

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

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

Відносні адреси вказуються від кореня сайту або поточного документа:

1. означає завантажити файл pic.png, який розміщений в тій же папці, що й сама веб-сторінка;

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

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

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

Для того, щоб зображення на веб-сторінці відображалось з нового рядка, потрібне додаткове використання тегу абзацу або тегу переходу на новий рядок. Наприклад:

Код	Результат
<pre><p> </p></pre>	

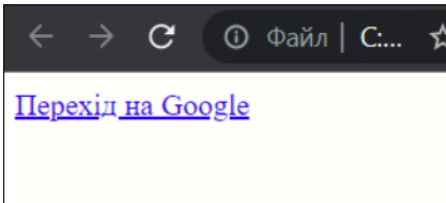
Веб-сторінки, як правило, завжди містять гіперпосилання для забезпечення можливості переходу на інші веб-сторінки. Гіперпосилання використовують у трьох випадках:

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

Синтаксис:

** текст гіперпосилання **

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

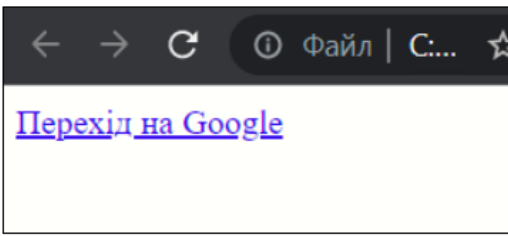
Код	Результат
<pre><p> Перехід на Google </p></pre>	

Гіперпосилання з якорем призначені для навігації в межах однієї сторінки. Якорем називають закладку з унікальним іменем в певному місці вебсторінки, яка призначена для переходу на неї за гіперпосиланням. Якорі зручно застосовувати в великих документах для швидкого переходу до певного розділу.

Створення якоря:

1. Зробити закладку у відповідному місці і дати їй унікальне ім'я з допомогою атрибута id.
2. При створенні гіперпосилання в якості значення href для переходу до цього якоря використовується значення id з символом решітки (#) попереду.

Наприклад:

Код	Результат
<pre><p> Перехід на Google </p></pre>	

Верстка веб-сторінок — процес формування веб-сторінок у текстовому або WYSIWYG-редакторі, наступний етап після веб-дизайну за допомогою HTML та CSS.

HTML (HyperText Markup Language — «мова гіпертекстової розмітки») — стандартна мова розмітки документів у Всесвітній павутині. Більшість веб-сторінок містять опис розмітки на мові HTML (або XHTML). Мова HTML інтерпретується браузерами та відображається у вигляді документа у зручній для людини формі.

CSS (англ. Cascading Style Sheets – каскадні таблиці стилів) – формальна мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки.

Основні елементи структури сайту (рис. 2.1):

- a. Header (заголовок) – це найперше, що впадає у вічі відвідувачу сайту. Зазвичай містить у собі логотип та навігаційне меню.
- b. Sidebar (бічна панель) – визначає блок збоку від контенту для розміщення рубрик, посилань на архів, міток та іншої інформації.
- c. Content (вміст) – в цій частині сайту міститься основна інформація, для представлення якої створена сторінка.
- d. Footer (колонтитул) – тут може розташовуватися ім'я автора, дата документа, контактна та правова інформація.

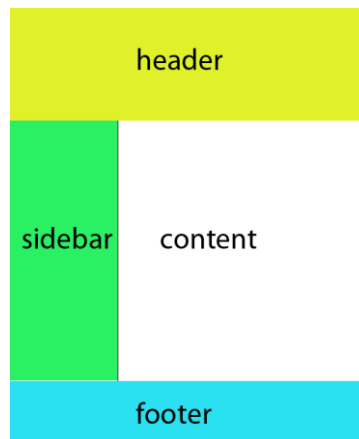


Рисунок 2.1 – Структура сайту

Структура блоків html (рис. 2.2):

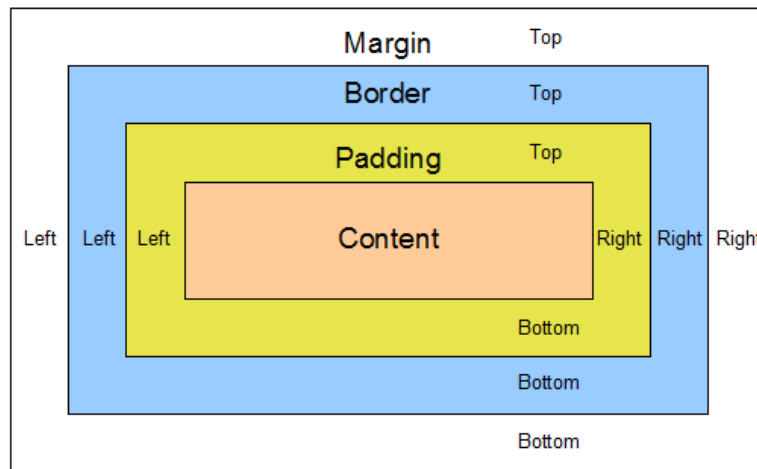


Рисунок 2.2 – Блок html

a. Margin.

Визначає відстань навколо елемента. Margin звільняє відстань навколо елемента (зовні від border). Margin не має кольору фону і завжди залишається прозорим.

b. Border.

Дозволяє визначати стиль та колір межі елемента.

c. Padding.

Визначає відстань між границею елемента та його вмістом.

d. Content.

Містить елементи цього блоку.

II. Завдання на практичну роботу

1. Створити веб-сторінку за зразком тільки засобами HTML, використовуючи теги з теоретичного матеріалу. Приблизний варіант сторінки наведено на рис. 2.3.

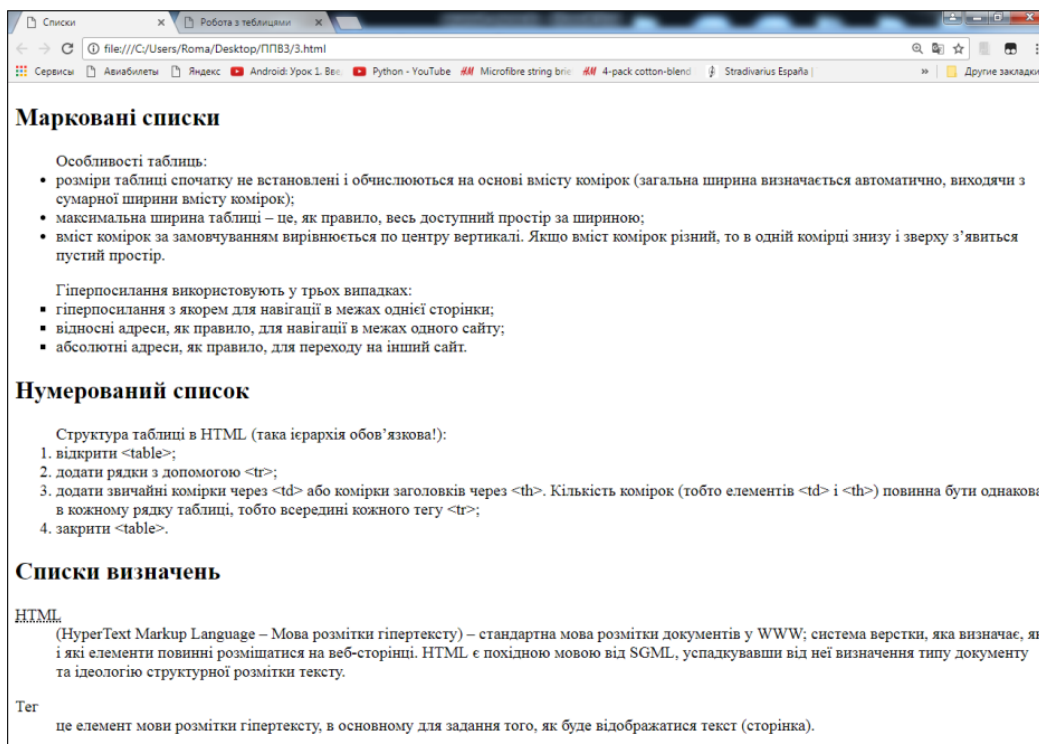


Рисунок 2.3 – Знімок екрану зразка веб-сторінки «Списки»

2. Створити таблицю лише з допомогою тегів та атрибутів.
3. Створити веб-сторінку зі списком гіперпосилань на інші сторінки. Зробити гіперпосилання на зовнішній ресурс за Вашим бажанням. Зробити гіперпосилання у вигляді зображення на довільний ресурс.

III. Контрольні питання

1. Що таке HTML?
2. Що таке тег?
3. Навести синтаксис написання тегу.
4. Навести структуру веб-сторінки.
5. Які теги використовуються для відображення абзацу на веб-сторінці?
6. Який тег використовується для задання заголовків на веб-сторінці?
7. Чим відрізняються теги, які відображають текст на веб-сторінці курсивом?
8. Чим відрізняються теги, які відображають текст на веб-сторінці напівжирним шрифтом?
9. Який тег використовується для задання аббревіатури на веб-сторінці?
10. Як задають списки на веб-сторінці?
11. Які теги використовуються для задання верхніх та нижніх індексів на вебсторінці?
12. Як можна відобразити спеціальні символи, яких немає на клавіатурі, на вебсторінці?
13. Який тегом позначають початок таблиці на веб-сторінці?
14. Які теги використовують для позначення рядків та стовпців таблиці?
15. Які атрибути призначені для об'єднання рядків та стовпців таблиці?
16. Яким тегом позначають гіперпосилання?
17. Який тег використовують для відображення малюнків на веб-сторінці?
18. Назвати обов'язковий атрибут тегу <a>.
19. Назвати обов'язковий атрибут тегу .
20. Що таке якір і як його створити?

Практична робота №3

Мова гіпертекстової розмітки HTML. Теги для створення форм та елементів управління на веб-сторінках. Уставляння додатків з зовнішніх джерел

I. Теоретичні відомості

Теги для створення форм та елементів управління на веб-сторінках

Тег `<form> ... </form>` є контейнером для елементів управління і дає можливість вводити інформацію і відправляти її на сервер.

Атрибути тегу:

- `name` – визначає ім'я форми, унікальне для даного документа.

Використовується, якщо в документі є кілька форм;

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

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

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

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

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

- `value` – значення елемента;

- `placeholder` – виводить текст-підказку;

- `size` – встановлює кількість видимих символів в `text`;

- `name` – ім'я елемента;

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

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

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

• `type = "text"` – створює рядок для введення тексту;

• `type = "password"` – створює рядок для введення тексту, при цьому відображає символи, які вводяться у вигляді зірочок;

• `type = "checkbox"` – створює перемикач "прапорець";

• `type = "radio"` – створює перемикач, який дозволяє вибрати одне значення з кількох альтернативних. Оскільки перемикачі є груповими елементами, то ім'я (`name`) у всіх елементів групи повинно бути однаковим;

• `type = "file"` – створює елемент для вибору локальних файлів;

• `type = "hidden"` – створює невидимий для користувача елемент. Може використовуватись для відправки додаткової службової інформації (наприклад, пароль);

• `type = "button"` – створює стандартну кнопку;

• `type = "submit"` – створює кнопку «Надіслати»;

• `type = "reset"` – створює кнопку «Скинути». При натисканні всі поля форми приймуть значення, дані їм за замовчуванням.

• `type = "color"` – віджет для вибору кольору;

• `type = "number"` – для введення чисел;

• `type = "range"` – повзунок для вибору чисел у вказаному діапазоні;

• `type = "date"` – поле для вибору календарної дати;

• `type = "datetime"` – дата і час;

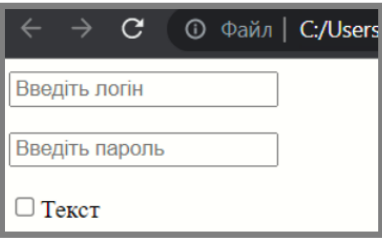
• `type = "datetime-local"` – місцеві дата та час;

• `type = "time"` – для часу;

• `type = "month"` – вибір місяця;

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

Наприклад.

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

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

```
<select>
  <option> пункт 1</option>
  <option> пункт 2</option>
  ...
</select>
```

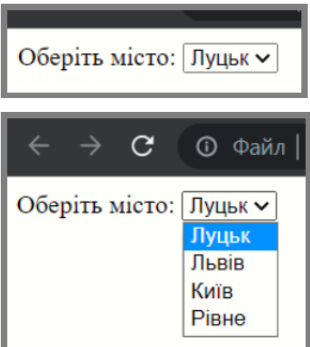
Атрибути тегу `<select> ... </select>`:

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

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

- selected – визначає пункт списку, який буде вибраний першим при завантаженні документа;
- value – задає даному пункту унікальне ім'я, яке буде використовуватись разом із іншими відомостями про вміст заповненої форми.

Наприклад.

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

`<optgroup> ... </optgroup>` дає змогу об'єднати групу елементів

`<option> ... </option>` із загальним надписом, яка, як правило, виділяється напівжирним шрифтом.

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

Атрибути:

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

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

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

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

`<fieldset>`

`<legend> Текст </legend>`

`</fieldset>`

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

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

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

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

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

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

Синтаксис:

1 спосіб

`<input id="<ідентифікатор>">`

`<label for="<ідентифікатор>"> Текст </label>`

2 спосіб

`<label><input> Текст </label>`

Уставлення додатків з зовнішніх джерел

Інтерфейс програмування додатків (API, application programming interface) це набір готових класів, процедур, функцій, структур і констант, що надаються додатком (бібліотекою, сервісом) для використання у зовнішніх програмних продуктах. Використовується програмістами для написання різноманітних додатків.

API для сайту - це скрипт, який надсилає GET/POST запити до певного сервісу, отримує від нього результат, що використовується для автоматизації певних дій.

Онлайнові картографічні сервіси

Картографічний сервіс створює карти, об'єкти і дані атрибутів всередині багатьох типів клієнтських додатків. Одним з традиційних методів використання картографічного сервісу є показ бізнес-даних поверх листів базової карти в Google Maps, Bing Maps або ArcGIS Online. Популярні геоінформаційні сервіси пропонують окрім перегляду карт пошук об'єктів, прокладання маршрутів і багато соціальних функцій. Кожна з популярних систем має свої сильні і слабкі сторони, зумовлені спеціалізацією або історією розвитку.



Google Maps

Google Maps



Bing Maps



Карты «Мета»

Уставляння карти на сайт

Як додати GOOGLE карту на сайт

Google maps – це безкоштовний картографічний додаток, який дозволяє в лічені секунди знайти будь-який об'єкт і прокласти до нього найкоротший маршрут. Картами можна користуватися не тільки в самому додатку, але і встановити їх до себе на сайт.

Оформлення сторінки контактів впливає на рівень довіри потенційних клієнтів, тому намагайтеся наповнити її максимально. Просто написана текстом адреса може ні про що не говорити відвідувачам, особливо якщо вони не орієнтуються в місцевості.

За допомогою інтерактивної карти проїзду можна:

- побачити, де знаходиться компанія;
- збільшити або зменшити масштаб;
- прокласти маршрут;
- подивитися кількість відгуків і рейтинг компанії в Google My Business.

Якщо ви поставите її на сайт, користувачам буде набагато зручніше зорієнтуватися, і ймовірність того, що вони до вас прийдуть, підвищиться. Якщо у вас кілька філій, Google карта дозволить наочно продемонструвати, де знаходиться кожен з них. І користувачам не потрібно буде шукати найближче відділення. Вони зможуть це зробити відразу на сайті.

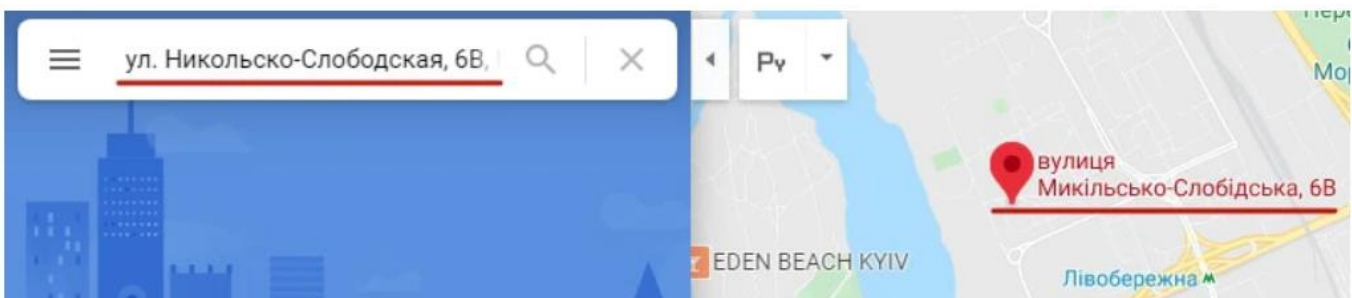
У Google Maps є можливість вбудувати на сайт:

- карту проїзду з встановленою на ній міткою;
- прокладений маршрут;
- зображення перегляду вулиць.

Розглянемо, як отримати HTML-код кожного з цих варіантів і додати його на сайт.

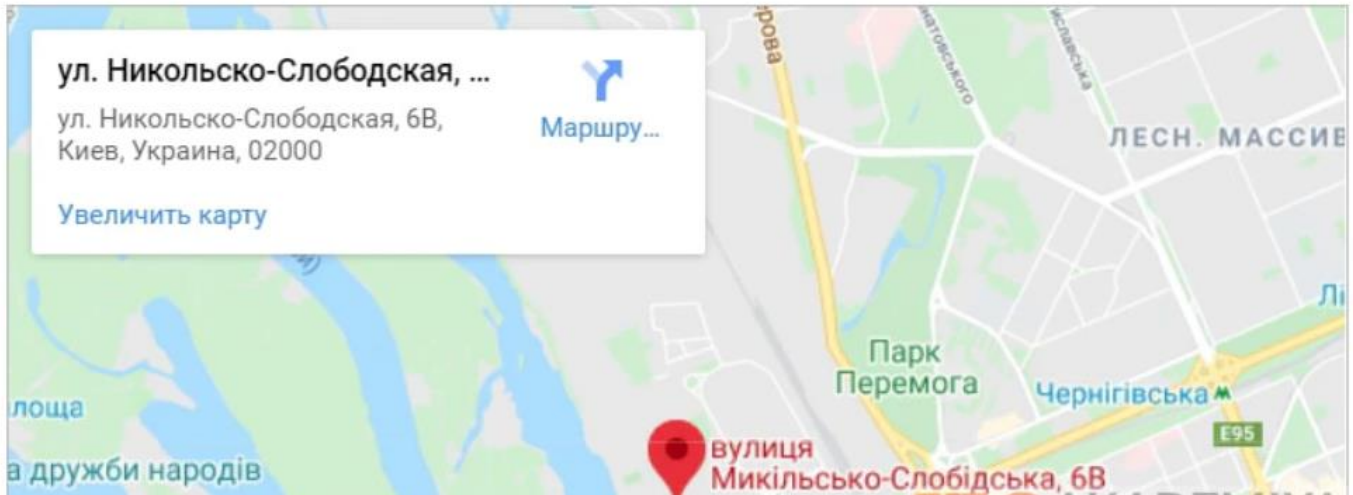
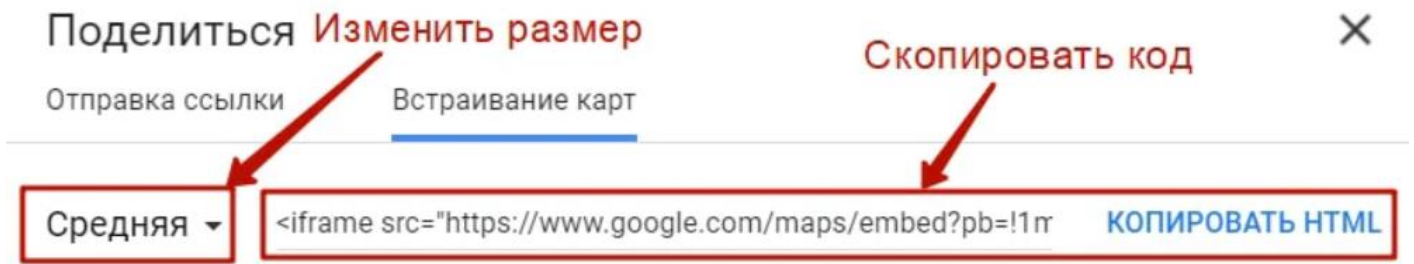
Карта з міткою це найпоширеніший і зручний варіант віджета. Щоб підключити його, виконайте наступні дії.

- Перейдіть в сервіс Google Maps
- У рядку пошуку напишіть адресу компанії, або ж знайдіть необхідну точку на карті і клікніть по ній.



- Натисніть « Поділитися ».
- Перейдіть у вкладку « Вбудовування карт ». Тут ви побачите код іframe карти і превью, як вона буде виглядати на сайті.

- Клікнувши по стрілці, розташованій зліва від коду, ви зможете налаштувати розмір: маленький, середній, великий або власний.



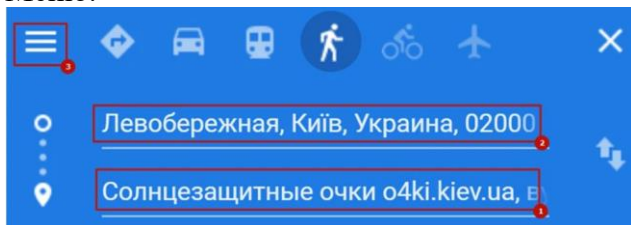
- Скопіюйте отриманий код.
важливо: Якщо ви зареєстровані в Google My Business, У встановленій на сайті карті буде відображатися назва компанії, зірки рейтингу і кількість відгуків. Якщо ж ні, тоді тільки зазначена мітка.

Якщо ви хочете показати користувачам, як дістатися до вашого офісу або магазин з певної точки, Наприклад, від найближчої станції метро, тоді:

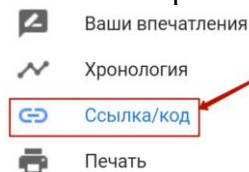
- Аналогічно поставте мітку там, де знаходиться ваша компанія.
- Натисніть “ Прокласти маршрут ”.



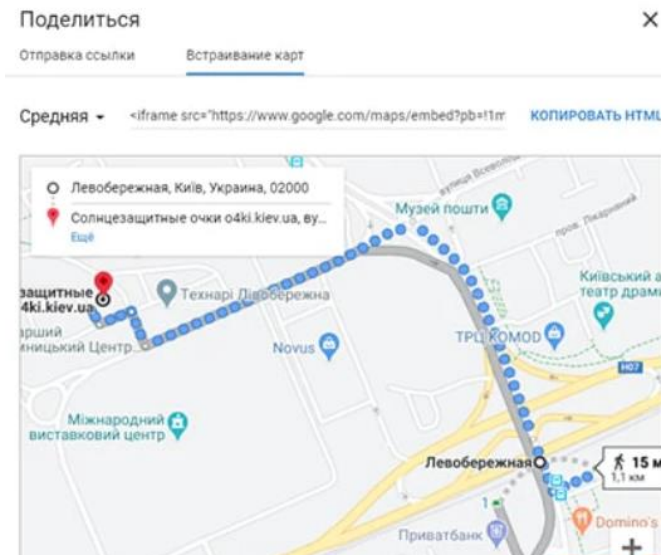
- Напишіть адресу вихідної точки або вкажіть її на карті.
- У лівому верхньому кутку сторінки клікніть на значок з 3-х горизонтальних смуг і перейдіть в Меню.



- У вікні виберіть пункт « Посилання / код ».

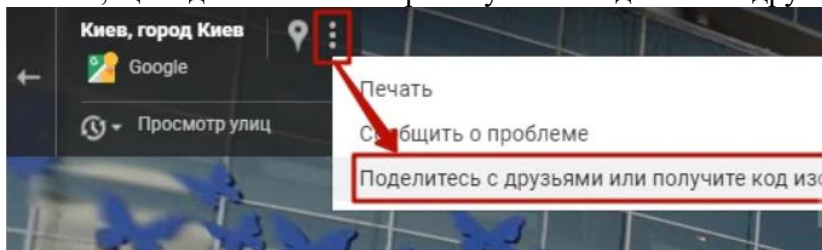


- Перейдіть в розділ « Вбудовування », встановіть необхідний розмір і скопіюйте отриманий код.

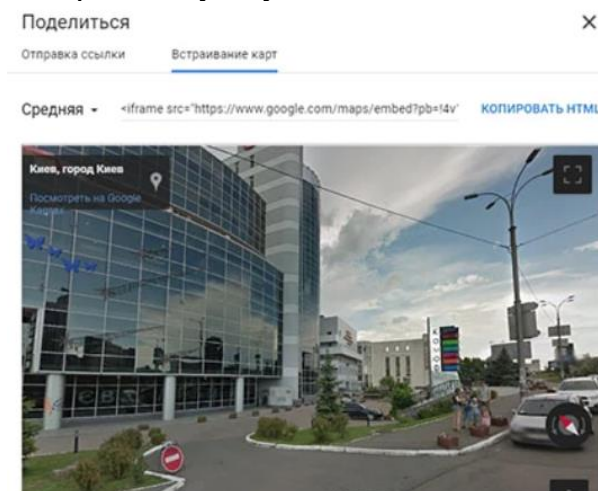


Якщо ви хочете вбудувати не просто схему проїзду, а ще й панорамний перегляд вулиць, Тоді:

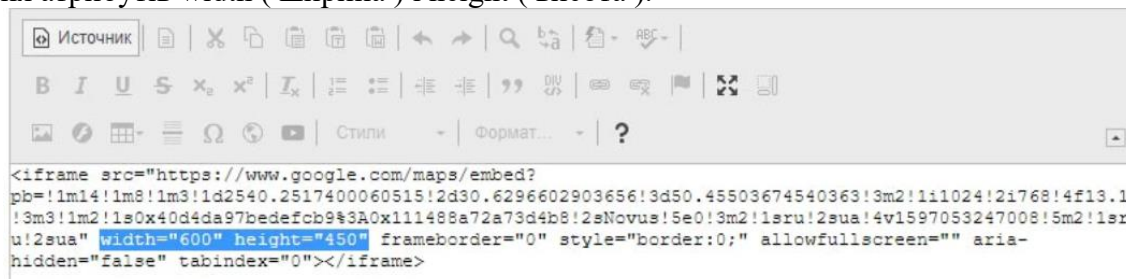
- Відкрийте необхідне зображення в Google Maps.
- Натисніть на 2 вертикальні точки, розташовані біля адреси.
- З вікна, що відчинилося виберіть пункт « Поділіться з друзями або отримаєте код ».



- Виберіть « Вбудовування » і скопіюйте код.



Тепер, щоб показати Google Maps користувачам, вам залишається тільки встановити iframe в вихідному коді сайту і зберегти зміни. У разі необхідності, ви можете підкоригувати розміри, змінивши значення атрибутів width (ширина) і height (висота).



Щоб карта була адаптивною і змінювалася в залежності від розміру екрану, вкажіть ширину 100%.

Мультимедійні сервіси

Відео-сервіс YouTube

YouTube - сервіс, що надає послуги відеохостингу. Користувачі можуть додавати, переглядати і коментувати відеозаписи. На сайті представлено як професійно зняті фільми і кліпи, так і любительські відеозаписи, включно з відеоблогами.

На сайті YouTube.com: користувачі можуть завантажувати відео в кількох поширених форматах, в тому числі. Mpeg і. Avi. YouTube автоматично конвертує їх у Flash Video і робить доступними для перегляду в онлайн.

За межами YouTube.com: кожне відео супроводжується готовою HTML-розміткою для вставки відео на інші веб-сторінки. Проста техніка CopyPaste зробила поширення відео з YouTube надзвичайно популярним, адже посилання на відео може бути впроваджено в HTML-код любої веб-сторінки.

Скачування відео: крім спеціальних сайтів, перетворюють посилання на сторінку з відео в посилання на файли, це дозволяють робити ряд сторонніх додатків (напр., SaveTube) і розширень для браузера (напр., UnPlug або інші розширення для Firefox або доповнення для Opera). Деякі менеджери завантажень також можуть зберігати відео з цього порталу (наприклад Download Master). Є також ряд спеціальних сайтів, які можуть завантажувати відео з YouTube.

Як вбудувати відео або плейлист:

1) Перейдіть на сайт youtube.com на комп'ютері та відкрийте ролик або плейлист, який ви хочете вбудувати.

2) Натисніть кнопку "Поділитися".

3) Виберіть Вбудувати.

4) Скопіюйте код.

5) Вставте код на веб-сайт.

6) Для адміністраторів мережі: додайте youtube.com до списку дозволених сайтів у налаштуваннях брандмауера.

Важливо! Якщо ви вбудовуєте матеріали з YouTube на сайт або додаток для дітей, вам необхідно вказати цільову аудиторію. На таких ресурсах відключаються деякі функції вбудованого програвача і не відображається персональна реклама.

HTML5 <audio>. Відтворення музики на сайті

Поява тегу <audio> в HTML5 надало можливості легко відтворювати звук.



На тепер існує 3 живих формати для <audio> - ogg, .mp3, wav. Файл з розширенням .ogg буде програватися в Firefox, Opera и Chrome, формат .mp3 або .wav для Internet Explorer і Safari.

- [Конвертор медіа-файлів он-лайн](#)
- [Уроки застосування HTML5 Audio](#)
- [Audio.js музика в будь-якому браузері](#)
- [Яндекс-віджет: Програма для редагування музики](#)
- [Аудіоплеєр для сайту на HTML5](#)

HTML5 <video>. Відтворення відео на сайті

- [Відео на сайті](#)
- [HTML5 відео](#)
- [11 ефективних HTML5 відео-плеєрів](#)

- [Власний відео-плеєр на HTML5 Video](#)

Віджети

Віджет це невеликий незалежний програмний модуль, що зроблено за допомогою технології API і який працює в деякому середовищі (напр. сайті, браузері, мобільному телефоні) та виконує, як правило, одну певну функцію.

Віджети також називають гаджетами, інформерами, а англійською gadget, badge, module, webjit, capsule, snippet, mini або навіть flake.

Веб-віджет (web widget) - це фрагмент коду, який може бути вбудований користувачем в HTML сторінку і використовуватися без значної модифікації. Як правило, при створенні веб-віджетів використовуються технології DHTML, JavaScript і Adobe Flash.

Веб-віджети можна умовно розділити на:

- Неінтерактивні, вміст і робота яких не залежить від дій користувача, що переглядає сторінку. Неінтерактивні віджети ще часто називають інформерами. Класичний приклад інформера - погодні інформер.
- Інтерактивні з якими користувач може взаємодіяти, наприклад, відправляти SMS або шукати маршрут на карті.

Кнопки соціальних мереж



Facebook. Кнопка "Поділитись"

Ця кнопка дозволяє додавати повідомлення до посилань, якими люди хочуть поділитися у хроніці, групах або повідомленні Facebook.

Якщо програма розроблена для iOS або Android, рекомендуємо використовувати не цю кнопку, а нативний діалог "Поділитися" в iOS та Android.

Якщо на веб-сайті не потрібна кнопка для відкриття діалогу "Поділитися" або кнопка Facebook не вписується в його дизайн, для публікації посилань можна використовувати діалог "Поділитися" для веб-платформи. Майте на увазі: щоб використовувати цей плагін, не потрібно інтегрувати вхід через Facebook або вимагати додаткових дозволів за допомогою перевірки програми.

Покрокові інструкції:

1. Виберіть URL-адресу або сторінку.

Виберіть URL-адресу веб-сайту або Сторінки Facebook, яким ви хочете поділитися.

2. Скористайтеся конфігуратором коду.

Вставте URL-адресу в [конфігуратор коду](#) і налаштуйте параметр layout кнопки "Поділитися".

Щоб створити код кнопки "Поділитися", натисніть Get Code.

3. Скопіюйте та вставте фрагмент коду HTML.

Скопіюйте та вставте фрагмент коду в HTML-код сайту.

Конфігуратор кнопки "Поделиться"

URL-адрес для публикации

Композиция

Размер кнопки

Facebook. Вбудовані публікації

Вбудовані публікації – зручний спосіб впровадити загальнодоступні публікації (від Сторінки або користувача Facebook) до контенту вашого сайту чи веб-сторінки. Можна вбудовувати лише загальнодоступні публікації від імені сторінок та профілів Facebook.

Meta for Developers Документи Інструменти Піддержка 🔍 Поиск в документации для раз

Генератор кодов

Кнопка "Сохранить"
Кнопка "Поделиться"
oEmbed
Старые конечные точки oEmbed
Сайты для детей
Best Practices
Часто задаваемые вопросы
Больше не используется

URL-адрес публикации:

Ширина публикации в пикселях (от 350 до 750):

Включить публикацию полностью

[Получить код](#)

Інформери

Щоб додати інформер до сайту не потрібно бути програмістом. Це невеликий фрагмент HTML коду, який вставляється в код сторінки сайту. Після додавання коду інформера, він почне відображатися на сторінці. Вся необхідна та актуальна інформація завантажуватиметься з серверів ресурсу, з якого отримано інформер. Код інформера жодним чином не впливає і не гальмує роботу сайту, оскільки це простий HTML код розмітки.

Увага!!! Деякі інформери відображаються лише після завантаження на хостинг, перегляд сторінки на локальному комп'ютері не надасть бажаних результатів.



Веб-хостинг

Веб-хостинг - це фізичне розміщення веб-сторінок на сервері. Це віртуальний аналог оренди приміщення, але орендується місце на диску, яке обчислюється мегабайтами. Від того, де буде розміщено сайт, залежить багато якісних характеристик, тому важливо вибрати оптимальний майданчик для сайту, що відповідає критеріям надійності та стабільності.

Хостинг умовно можна поділити на безкоштовний і платний. Для початківців, які починають перші кроки в області веб-розробки вірним рішенням буде реєстрація майданчика на безкоштовному хостингу.

Плюси безкоштовного хостингу

- Не потрібно платити. Розробник початківець тільки починає робити перші кроки і мало обізнаний в нюансах розміщення, показниках якості хостинга, налаштуваннях панелі та акаунту. Тому, варто, безкоштовно опанувати прийоми і зауважити ті критерії, які важливі для ефективної роботи: простота налаштувань, зручність адміністративної панелі, наявність вбудованих редакторів та файлових менеджерів.
- Безкоштовний тариф пропонує послуги аналогічні до платних. Надаються всі основні можливості (з певними функціональними обмеженнями), що потрібні для розміщення сайту: дисковий простір, підтримка мов програмування, під'єднання до баз даних, встановлення поширеної CMS - системи управління контентом, створення поштового акаунту та інше.

- Технічна підтримка. Користувачам надаються конструктори та інші інструменти, що полегшують створення, підтримку та просування сайту. Повсюдно присутня довідкова система, яка охоплює багато загальних та специфічних проблем, з якими стикаються початківці. Наявність спільнот так само містить багато форумів та обговорень, де можна завжди з'ясувати шляхи вирішення найпоширених проблем.
- Можливість отримання безкоштовної доменної адреси 3 рівня. Безкоштовні хостинги надають можливості для вибору будь якого під домену до основного домену хостингу. Початківець позбувається проблеми щодо реєстрації, оплати та підтримки доменної адреси.

Обмеження безкоштовного хостингу

- Обмежений дисковий простір, але цілком достатній для розміщення нескладного сайту. На сьогодні цей показник складає від 1Гб і більше.
- Немає гарантій щодо захисту та доступності сайту. На таких площадках не роблять резервні копії сайтів і тому, у разі збоїв серверів сайт може бути пошкоджений або знищений без можливості відновлення. Створення резервних копій є турботою власника сайту.
- Деякі безкоштовні хостинги мають дохід за рахунок реклами. Де, скільки і за якою тематикою буде реклама на сайті користувача вирішують власники сервісу. Як правило, заборонено розміщувати власну рекламу і будь-яку іншу комерційну інформацію.
- Можуть бути присутні обмеження на тримання файлів певного формату (аудіо, відео, скрипти), а також на просте складування документів.
- При явній популярності сайту власники хостингу можуть висунути умови перенесення ресурсу на платний тариф, погрожуючи знищенням сайту. Прикрий випадок, коли вкласти багато часу, сил і фінансових коштів в створення сайту, в його наповнення контентом і просування. Закрити сайт, коли він вже став популярним і має чималу аудиторію відвідувачів не завжди просто.

Поради для вибору безкоштовного хостингу

- Віддавати перевагу більш популярним сервісам, бажано з давньою історією та хорошою репутацією.
- Заздалегідь почитати відгуки про даний хостинг в Інтернеті: чи достатньо там функціональних можливостей, наскільки зручно користуватися цим сервісом (способи редагування файлів і завантаження на сервер, кількість і різноманітність готових шаблонів дизайну).

На сьогодні розробникам початківцям надається великий вибір безкоштовних майданчиків, як українських так і іноземних. Вони дещо різняться за своїми показниками та умовами розміщення, але в загальному мають приблизно однаковий набір функціоналу.



000webhost.com



byet.host



awardspace.com



freehostingeu.com/



1gb.ua

II. Завдання на практичну роботу

1. Створити веб-сторінку за поданим зразком (рис. 3.1);

Мozilla Firefox

Файл Правка Вигляд Історія Закладки Інструменти Довідка

file:///C:/Documents and Settings/teacher/Робочий стол/forma.html

file:///C:/Docume...0стол/forma.html

Регістрація

Введіть своє ім'я:

Введіть своє прізвище:

Введіть пароль:

Виберіть стать: чол. жін.

Виберіть дату народження:

Виберіть мови програмування, які Ви знаєте:

- Pascal
- PHP
- Delphi
- JavaScript
- Basic

За потреби, можете залишити свій коментар тут:

Зроблено

Рисунок 3.1 – Знімок екрану прикладу форми

2. Ознайомитися з наведеними картографічними та мультимедійними сервісами та можливостями їх API.
3. Користуючись сервісом Google Maps втілити карту, що відповідає певному місцю: будинок студента, навчальний корпус Київської політехніки, улюблене кафе чи пам'ятка. На карті проставити мітки, прокласти маршрути, змінити фон, схему відображення. Додатково розмістити панорамний перегляд цієї або іншої місцевості.

4. Ознайомитися з наведеними мультимедійними ресурсами. Відредагувати засобами YouTube власне чи довільне відео. Додати звукове супроводження, зробити нарізку, додати субтитри. Втілити відредаговане відео у відповідну сторінку сайту.
5. За допомогою тегу <audio> або audio.js додати музичний файл на сайт.
6. Поставити на сторінку кнопки плагінів соціальних мереж та інформери.
7. Розмістити тестову сторінку на хостингу. Результат виконання демонструвати в Інтернеті.
8. Під час виконання роботи робити скріншоти основних етапів роботи. Оформити звіт.

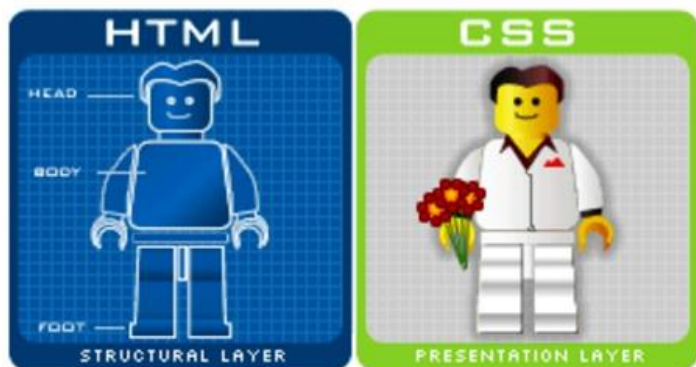
III. Контрольні питання

1. Яким тегом задають форму на веб-сторінці?
2. Як відобразити рядок для введення на веб-сторінці?
3. Як відобразити рядок для введення паролю на веб-сторінці?
4. Як відобразити прапорець на веб-сторінці?
5. Як відобразити перемикач на веб-сторінці?
6. Як відобразити кнопку на веб-сторінці?
7. Як відобразити випадаючий список на веб-сторінці?
8. Як відобразити елемент для вибору локальних файлів на веб-сторінці?
9. Як відобразити кнопку-зображення на веб-сторінці?
10. Як відобразити область для введення багаторядкових коментарів на вебсторінці?
11. Які є типи випадаючих списків?
12. Для чого розробляються API додатків, яку функціональність вони надають?
13. Яким чином можна використати сторонній сервіс з доступним API на власному сайті?
14. В чому переваги картографічних сервісів? Яким чином можна вставити онлайн карту на сайт?
15. Які обмеження у використанні накладено картографічним сервісом GoogleMaps?
16. На які особливості відтворення аудіо файлів на сайті слід вважати: спосіб уставляння, формати файлів, режими відтворення?
17. Які особливі елементи присутні в мові HTML5, для уставляння аудіо та відео. Чи цей підхід є зручним?
18. Особливості уставляння віджетів соціальних мереж на користувацький сайт. Які переваги надає їх наявність?
19. Які послуги надають інформери, як їх можна підключити?

Практична робота №4 Стильове оформлення сторінок

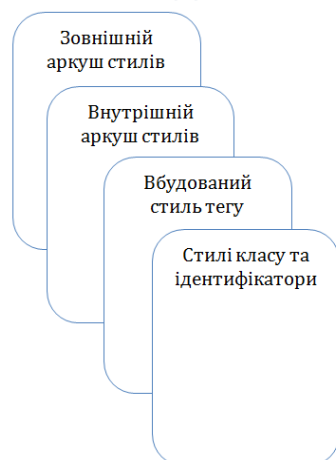
I. Теоретичні відомості

Створивши сторінку, її інформаційне та змістове наповнення, можна приступати до дизайну – тобто графічного оформлення цієї сторінки. За стандартом HTML таке оформлення задається через стилі, а точніше каскадні таблиці стилів.



Стиль – це набір правил оформлення та форматування, який можна застосувати до різних елементів сторінки. Каскадні таблиці стилів, (так звані CSS, Cascading Style Sheets) містять опис формату частини або всього тексту. В одному документі можна описати кілька аркушів, браузер використовуватиме їх каскадом, відповідно до пріоритету цих описів.

Каскад стилів



Переваги стилів:

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

Розрізняють декілька типів стилів, які можуть спільно застосовуватися до одного документа. Це стиль браузера, стиль автора і стиль користувача. Стиль браузера – це оформлення, яке за замовчуванням застосовується до елементів веб-сторінки браузером. Це оформлення можна побачити у випадку HTML-тексту, коли до документа не додається ніяких стилів. Стиль автора – це стиль, який додає до документа його розробник. Стиль користувача – це стиль, який може включити користувач

сайту через налаштування браузера. Такий стиль має більш високий пріоритет і перевизначає вихідне оформлення документа.

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

Стиль можна задавати для конкретного тега, для стандартних тегів у документі чи кількох документах, а також для ідентифікаторів та класів, створених для проекту.

Таким чином виділяється зовнішній аркуш стилів, внутрішній, вбудований стиль тегу та стилі класів та ідентифікаторів. Вони застосовуються по порядку, і найменший пріоритет мають зовнішні аркуші стилів, а найвищий – стилі конкретних тегів.

**h1 {font-size:20pt; color:green;
font-family:"Comic Sans MS"}**

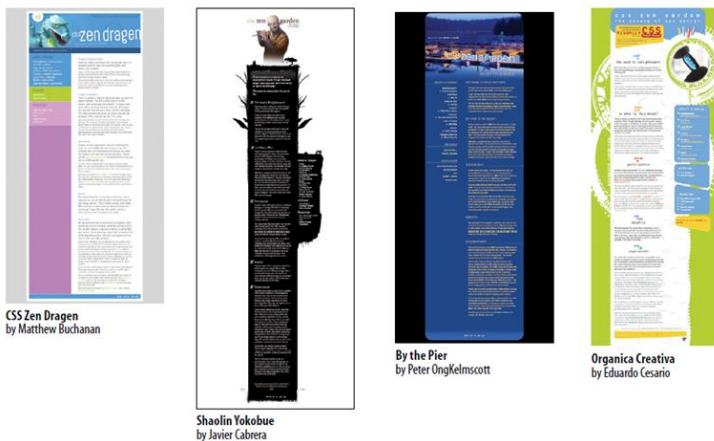
h1 {color : green}



Селектор Властивість Значення

Стилі визначають форматування, а також розташування елементів у документі. Так, можна змінити вигляд усіх заголовків чи вирівнювання усіх зображень документа. Крім того можна задати певне оформлення для горіща чи підвалу сайту, навігаційної панелі тощо.

Одна і та ж сторінка



Прикладами стильових ефектів є наступні:

- background – колір тла;
- font-family – вид шрифту;
- font-size – розмір шрифту;
- color – колір шрифту;
- text-decoration – оздоблення тексту;
- font-weight – жирність шрифту;
- text-align - вирівнювання тексту;
- border-width - ширина границі;
- border-style - стиль оформлення границі;
- margin-top – відстань від верхнього рядка;
- line-height – висота рядка.

Довідник стилів: <https://css.in.ua/>

Підключати аркуші стилів можна:

- в окремому файлі:

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

Зовнішню таблицю стилів можна створити в довільному текстовому редакторі. Файл зовнішньої таблиці стилів не може містити ніяких тегів HTML, немає заголовку і закінчення, як в HTML (див. рис. 4.1). Розширення файла таблиці стилів – .css. Значення параметрів rel і type залишаються незмінними незалежно від коду.

Наприклад.

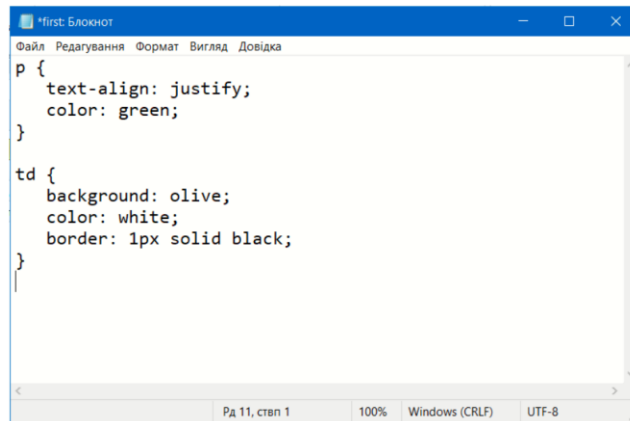


Рисунок 4.1 – Знімок екрана редактору Блокнот з фрагментом коду CSS

- на початку веб-документа:

```
<style type="text/css">
```

...

```
</style>
```

- в конкретному тезі:

```
<h1 style="font-size: 120%; font-family: Verdana, Arial, Helvetica, sans-serif; color: #336">Заголовок </h1>
```

- для спеціально виділених блоків:

```
<div style="font-size: 120%; font-family: Verdana, Arial, Helvetica, sans-serif; color: #336">Hello World!</div>
```

- для блоку з ідентифікатором:

```
#para1 {  
  text-align: center;  
  color: red;  
}  
<p id="para1">Hello World!</p>
```

для блоків з селектором класу:

```
.center {  
  text-align: center;  
  color: red;  
}  
<h1 class="center">Заголовок червоного кольору, вирівнювання по центру</h1>  
<p class="center">Текст червоного кольору, вирівнювання по центру</p>
```

Розширені можливості CSS та особливості їх використання при верстці веб-сторінок

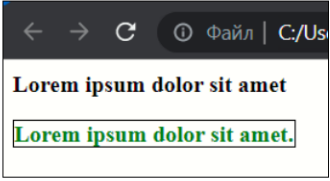
Контекстні селектори. При створенні веб-сторінок часто доводиться вкладати одні теги в інші. Для коректного використання стилів цих тегів застосовують селектори, які працюють тільки в певному контексті. Таким чином можна встановити стиль для окремого тегу, а також для тегу, який знаходиться

всередині іншого. Такі селектори називають контекстними. Вони складаються з кількох частин, розділених пробілом. Синтаксис:

селектор1 селектор2 {правила стилю}

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

Наприклад:

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

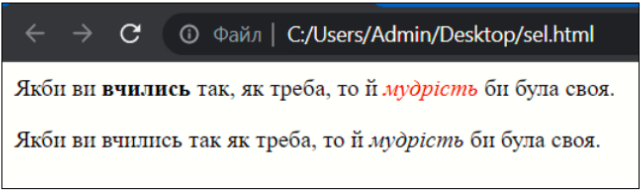
Сусідні селектори використовуються для розташованих поряд елементів.

Наприклад, теги `` в списку є сусідніми по відношенню один до одного і вкладеними в тег ``.

Сусідні селектори записуються з допомогою знаку `+`. Синтаксис:

селектор1 + селектор2 {правила стилю}

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

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

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

селектор 1 > селектор 2 {правила стилю}

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

Наприклад.

Код
<pre> <style> P > I { color: red; } </style> </head> <body> <div> <i>Lorem ipsum dolor sit amet, consectetur adipiscing.</i> <p><i>Lorem ipsum dolor sit amet</i>, consectetur adipiscing elit, <i>sed diem nonummy</i> nibh euismod tincidunt.</p> </div> </body> </pre>
Результат



Споріднені селектори схожі на сусідні селектори, але, на відміну від них, стильові правила застосовуються до всіх елементів, що розташовані поряд.

Наприклад, для селектора `h1~p` стиль буде застосовуватись до всіх елементів `<p>`, які знаходяться після заголовка `<h1>`. При цьому `<h1>` і `<p>` повинні мати спільного батьківського елемента. Тому якщо `<p>` помістити всередину `<div>`, то стилі застосовуватися вже не будуть.

Синтаксис:

E ~ F { опис правил стилю }

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

Наприклад.

h1 ~ p { color: red; }

`<h1>Заголовок</h1>`

`<p>Абзац 1</p>`

`<p>Абзац 2</p>`

(червоний колір тексту буде встановлений для всіх абзаців)

h1 ~ p { color: red; }

`<h1>Заголовок</h1>`

`<p>Абзац 1</p>`

`<div><p>Абзац 2</p></div>`

`<p> Абзац 3</p>`

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

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

селектор::псевдоелемент {правила стилю}

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

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

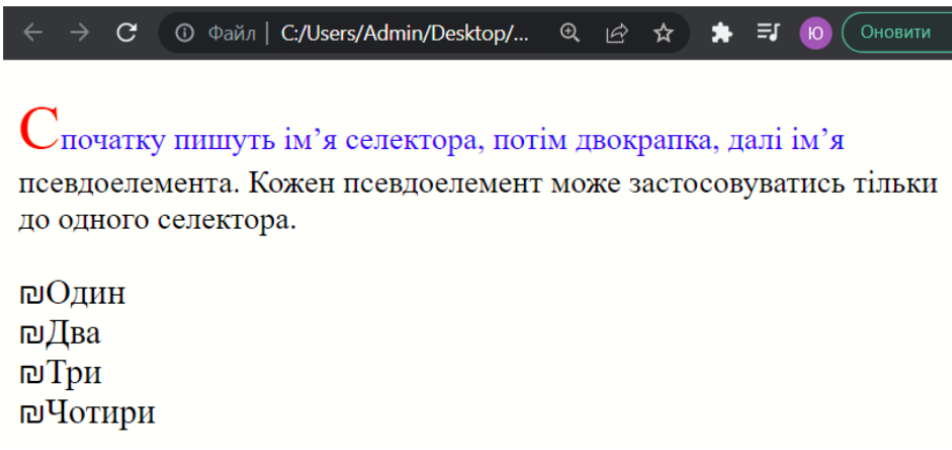
`::after` застосовується для вставки призначеного контенту після вмісту елемента. Стильова властивість `content` визначає вміст для вставки;

`::before` – вставляє контент до вмісту елемента;

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

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

Наприклад.

Код
<pre><head> <style> ul { padding-left: 0; list-style-type: none; } li::before { content: "\20aa "; } p { font-size: 90%; color: black; } p::first-letter { font-family: 'Times New Roman', Times, serif; font-size: 200%; color: red; } p::first-line{ color: blue; } </style> </head> <body> <p> Спочатку пишуть ім'я селектора, потім двокрапка, далі ім'я псевдоелемента. Кожен псевдоелемент може застосовуватись тільки до одного селектора. Якщо потрібно відразу встановити кілька псевдоелементів для одного селектора, правила стилю повинні додаватися до них окремо: </p> <p>Спочатку пишуть ім'я селектора, потім двокрапка, далі ім'я псевдоелемента. Кожен псевдоелемент може застосовуватись тільки до одного селектора. Якщо потрібно відразу встановити кілька псевдоелементів для одного селектора, правила стилю повинні додаватися до них окремо: </p> Один Два Три Чотири </body></pre>
Результат


Псевдокласи визначають динамічний стан елементів, який змінюється з допомогою дій користувача, а також положення в дереві документа. Прикладом такого стану є гіперпосилання, яке змінює свій колір при наведенні на нього курсора миші. При використанні псевдокласів браузер не перезавантажує поточний документ. Тому з їх допомогою можна отримати динамічні ефекти на сторінці.

Синтаксис:

селектор:псевдоклас {правила стилю}

Спочатку вказують селектор, до якого додається псевдоклас, а далі ім'я псевдокласу.

Допускається застосовувати псевдокласи до імен ідентифікаторів або класів, до контекстних селекторів. Якщо псевдоклас вказується без селектора спереду, то він буде застосовуватись для всіх елементів документа.

Умовно всі псевдокласи діляться на три групи:

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

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

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

:link – вибирає невідвідані гіперпосилання;

:focus – застосовується до елемента при отриманні ним фокусу;

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

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

Псевдокласи, які відносяться до дерева елементів;

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

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

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

Синтаксис:

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

- odd – всі непарні номери елементів;

- even – всі парні номери;

- <число> – порядковий номер елемента відносно свого батьківського елемента. Нумерація починається з 1;

- <вираз> – задається у вигляді $an + b$, де a і b – цілі числа, а n – автоматично приймає значення 0, 1, 2, ...

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

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

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

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

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

II. Завдання на практичну роботу

1. Використовуючи засоби CSS для форматування тексту (підключити зовнішню таблицю стилів), створити сторінки за поданим зразком. Приблизний варіант сторінки наведено на рис. 4.2-4.3. На сторінках встановити однаковий стиль для заголовків двох різних рівнів (колір і шрифт); вирівнювання, колір шрифту та відступ першого рядка абзацу; фон сторінок.

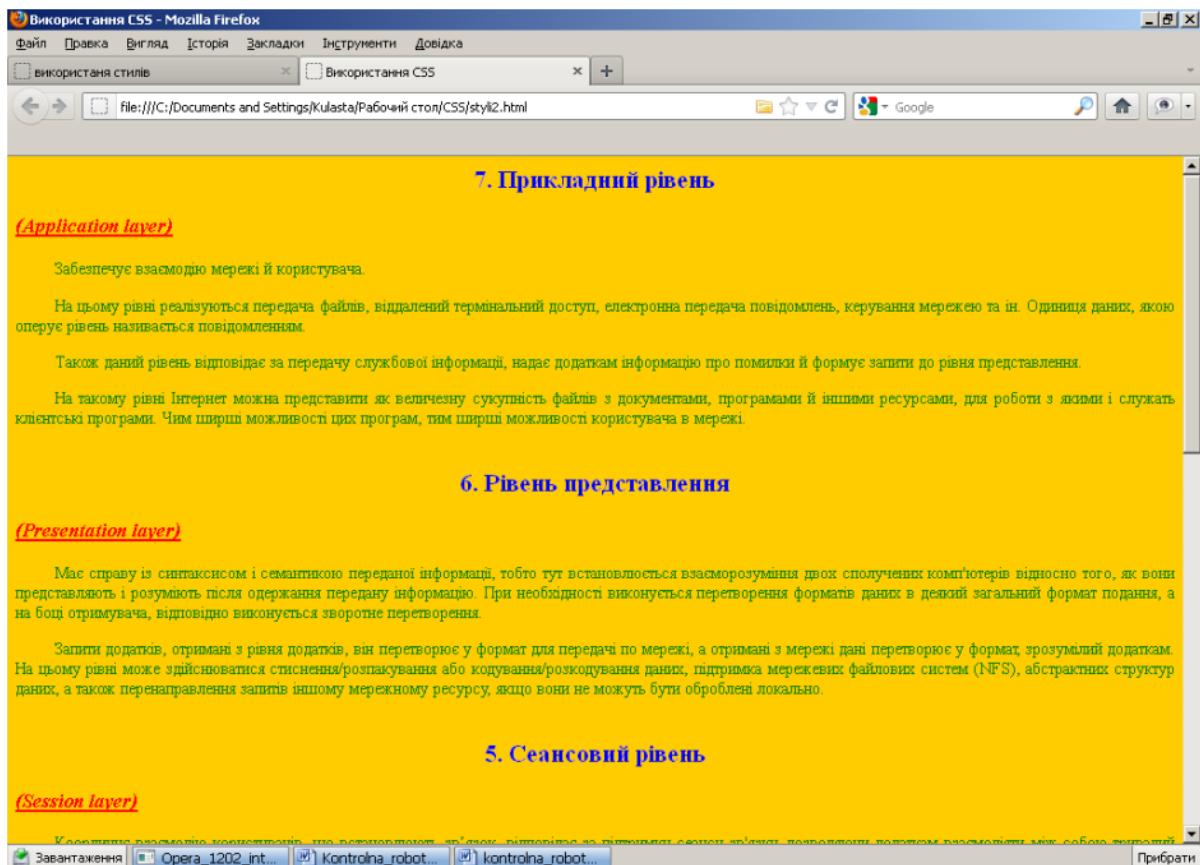


Рисунок 4.2 – Знімок екрану зразка веб-сторінки «CSS»

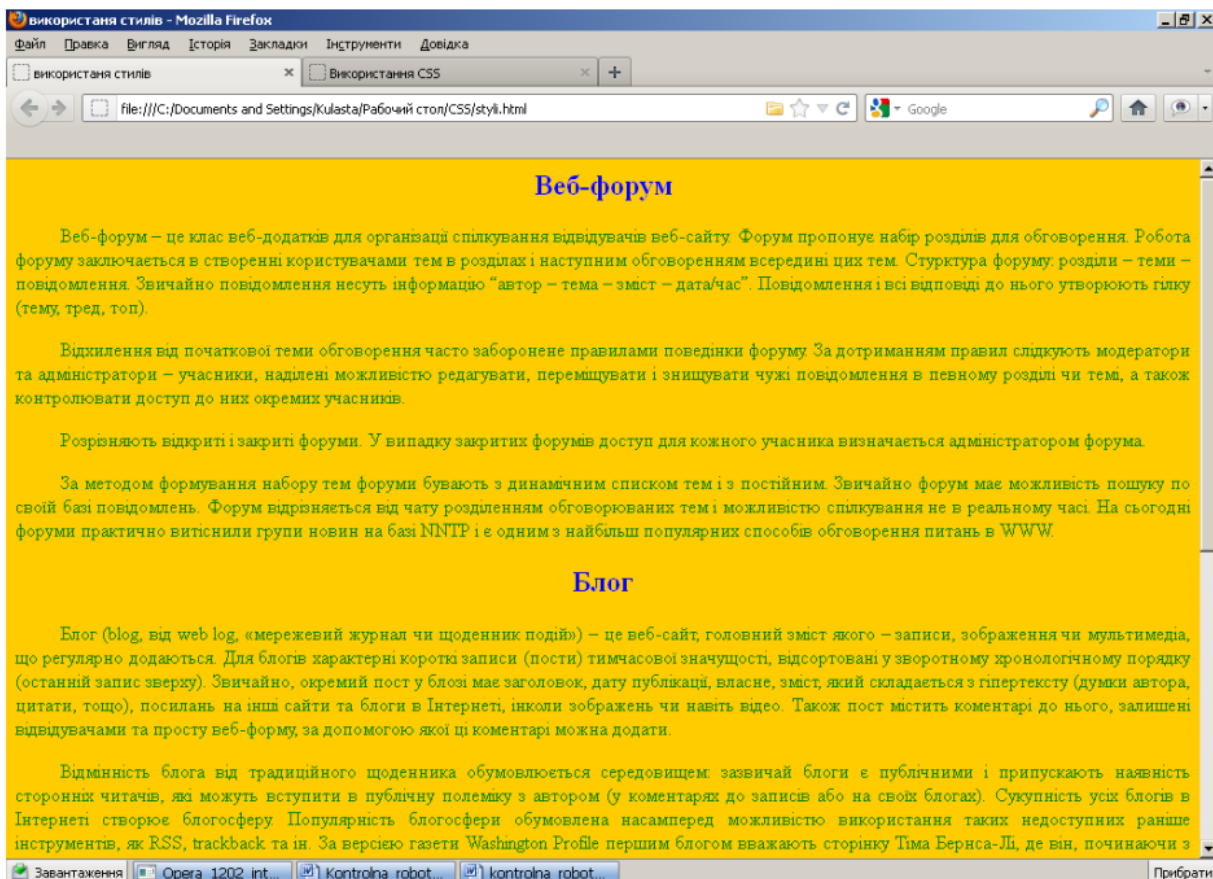
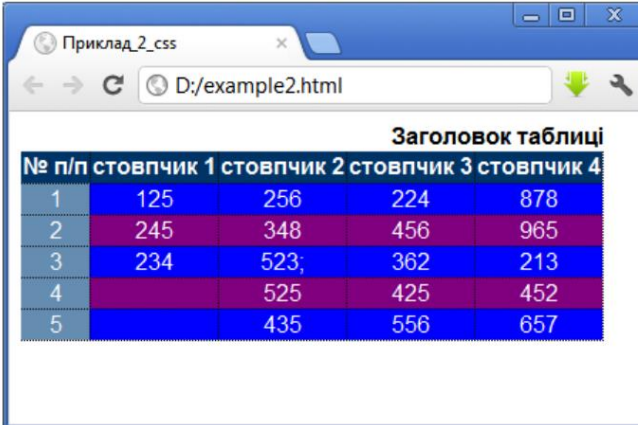


Рисунок 4.3 – Знімок екрану зразка веб-сторінки «CSS»

2. Використовуючи засоби CSS для форматування тексту та класи (підключити зовнішню таблицю стилів), створити сторінку за поданим зразком (рис. 4.4).



Заголовок таблиці				
№ п/п	стовпчик 1	стовпчик 2	стовпчик 3	стовпчик 4
1	125	256	224	878
2	245	348	456	965
3	234	523	362	213
4		525	425	452
5		435	556	657

Рисунок 4.4 – Знімок екрану зразка веб-сторінки «Класи»

III. Контрольні питання

1. Що таке CSS?
2. Перерахувати переваги використання CSS.
3. Що таке селектор?
4. Навести синтаксис написання селекторів.
5. Перерахувати типи каскадних таблиць стилів.
6. Як можна підключити каскадну таблицю стилів?
7. Що таке клас?
8. Навести синтаксис написання класів.
9. Що таке ідентифікатор?
10. Навести синтаксис написання ідентифікаторів.
11. Що таке контекстні селектори, їх синтаксис.
12. Що таке сусідні селектори, їх синтаксис.
13. Що таке дочірні селектори, їх синтаксис.
14. Що таке псевдоелемент?
15. Що таке псевдоклас?
16. Навести синтаксис написання псевдоелементів.
17. Навести синтаксис написання псевдокласів.
18. Перерахувати псевдоелементи, які ви знаєте.
19. Як поділяються псевдокласи?
20. Перерахувати псевдокласи, які визначають стан елементів.
21. Перерахувати псевдокласи, які відносяться до дерева елементів.
22. Перерахувати псевдокласи, які вказують на мову тексту.

Практична робота №5

Підключення бази даних до сайту

I. Теоретичні відомості

Існує декілька типів підключення бази даних до сайту. Основні з них, це використання PHP скриптів и технологій ASP.NET.

1.1 Використання PHP скриптів

Отримання інформації через БД відбувається в кілька етапів.

Відвідувач запитує веб-сторінку, вказуючи в браузері її адресу (URL).

Веб-сервер (Apache в нашому випадку) визначає, що запитується PHP-файл і запускає його інтерпретатор.

Скрипт PHP звертається до MySQL і запрошувати необхідну інформацію.

База даних MySQL повертає результат запиту назад у програму PHP.

Скрипт аналізує отриману інформацію і зберігає її в одній або декількох змінних. Потім текст виводиться за допомогою функції echo.

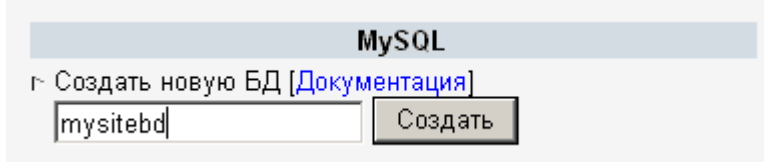
Остаточо сформований програмою код HTML відправляється на веб-сервер, де він пересилається відвідувачу.

Створення бази даних

При виборі хостингу обов'язково слід поцікавитися, закладена чи в обраному тарифному плані база даних MySQL. Після відкриття доступу до сайту вам повідомляється чотири параметра для управління базою: ім'я хоста БД, ім'я користувача, ім'я бази і пароль до неї. Щоб перевірити загальні принципи роботи з даними, створимо свою локальну БД і буде надалі маніпулювати з нею.

Якщо ви встановили комплект Денвер, запустіть веб-сервер Apache і наберіть в браузері адресу <http://localhost/phpMyAdmin/index.php>.

Після чого відкриється панель phpMyAdmin через яку можна створити БД, вказавши її ім'я.



Введіть відповідне ім'я латинськими символами, і база буде створена.

Підключення до MySQL

Для підключення до існуючої БД використовується функція `mysql_connect`. Її синтаксис такий.

```
mysql_connect (адреса, ім'я користувача, пароль)
```

Адреса - це IP-адреса або ім'я хоста комп'ютера, де запущена MySQL (наприклад, localhost для локальної БД). Ім'я користувача і пароль необхідні для підключення до бази (приклад 1).

Приклад 1. Підключення до сервера БД

```
<? php
$ dbhost = "localhost"; // Ім'я хоста БД
$ dbusername = "root"; // Користувач БД
$ dbpass = ""; // Пароль до бази
$ dbconnect =mysql_connect ($ dbhost, $ dbusername, $ dbpass);
if (! $ dbconnect) {echo ("Не можу підключитися до сервера бази даних!"); }
?>
```

Пароль до локальної бази можна не вказувати, тому він опущений. Зверніть увагу на значок @ перед іменем mysql_connect. Якщо його не вказати, буде виведена наступна рядок:

```
Warning: mysql_connect () [function.mysql-connect]: Unknown MySQL Server Host 'localhost' (11001) in z: \ home \ mysite.ru \ www \ connectbd.php on line
```

Використовуючи символ @ можна показувати свої власні повідомлення про помилки. У прикладі 1, якщо змінна dbconnect не визначена, що може бути тільки в разі невдачі підключення до БД, про це буде виведено попередження.

Вибір бази даних

Наступний крок для отримання даних полягає у виборі потрібної бази, що зберігається на сервері. Для цього використовується функція mysql_select_db. Її синтаксис наступний.

```
mysql_select_db (ім'я БД, $dbconnect)
```

Змінна \$dbconnect - це ідентифікатор підключення до сервера БД, як було показано в прикладі 1, використовувати її в даному випадку не обов'язково (приклад 2).

Приклад 2. Підключення до бази даних

```
<? php
$dbhost = "localhost"; // Ім'я хоста БД
$dbusername = "root"; // Користувач БД
$dbpass = ""; // Пароль до бази
$dbname = "mysitedb"; // Ім'я бази

$dbconnect = mysql_connect ($dbhost, $dbusername, $dbpass);
if (! $dbconnect) {echo ("Не можу підключитися до сервера бази даних!"); }
if (mysql_select_db ($dbname)) {echo "Підключення до бази $dbname встановлено!";
}
else die ("Не можу підключитися до бази даних $dbname!");
?>
```

Відслідковувати виникнення помилки можна різними способами, але скрізь використовується функція die для переривання роботи програми і виведення попередження (приклад 3).

Приклад 3. Контроль над помилками при підключенні до БД

```
if (!mysql_select_db ($dbname)) die ("Не можу підключитися до бази даних $dbname!");
```

Або

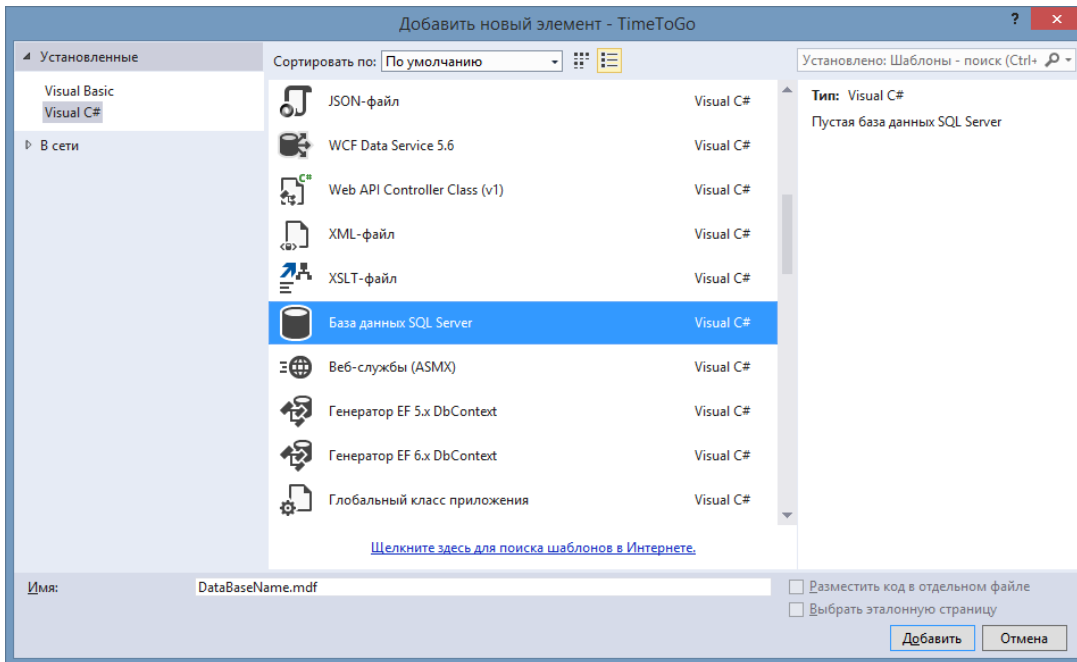
```
mysql_select_db ($dbname) or die ("Не можу підключитися до бази даних $dbname!");
```

Функція die, як і функція echo призначена для виведення тексту, але після її застосування робота програми переривається.

1.2 Використання ASP.NET

Для використання БД на технологіях ASP.NET необхідно, щоб сайт був написаний, як проект сайту ASP.NET. Також необхідні SQL Server Express та IIS. Якщо використовуєте Visual Studio 2013, то все це вже встановлено за замовчуванням. Якщо ваш сайт відповідає цій вимозі, то можна використовувати всі можливості.

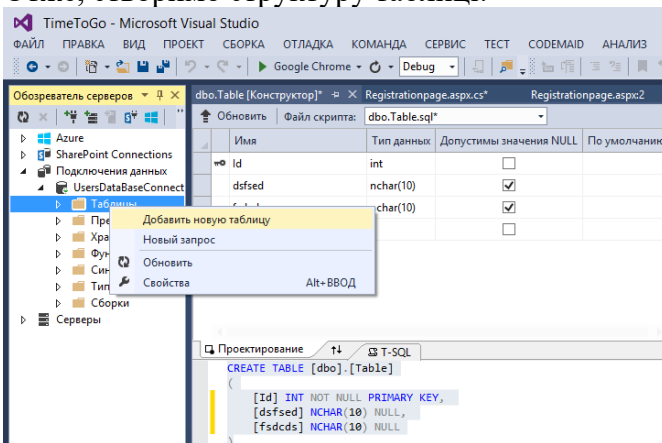
Ми можемо створити базу даних прямо в проекті, або ж створити її на сервері MS SQL. Для зберігання баз даних в проекті призначена папка App_Data. Для цього натиснемо правою кнопкою миші на проект і в контекстному меню виберемо Add-> New Item У вікні додавання нового елемента виберемо SQL Server Database і назвемо нову базу даних DataBaseName.mdf:



Ми можемо створити базу даних і на сервері. Після цього база даних додається в проект, і ми можемо побачити її в папці App_Data. Тепер в оглядачі баз даних (вікно Server Explorer) ми можемо підключитися до неї і створити таблиці, які будуть зберігати дані.

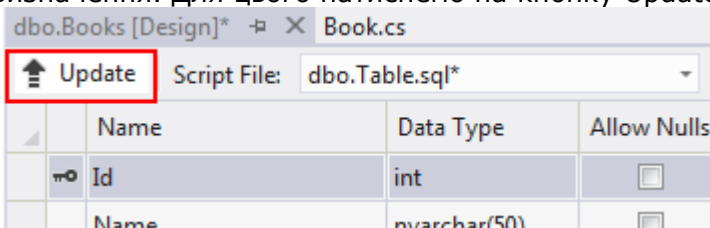
Розкриємо вузол DataBaseName.mdf і знайдемо вузол Tables. Натиснемо на цей вузол правою кнопкою миші і в меню виберемо пункт Add New Table. І перед нами з'явиться вікно, в якому нам треба визначити назви і типи стовпців нової таблиці. За угодами про найменування таблиці при роботі з Entity Framework повинні відповідати імені моделі. Тобто, так як наша модель називається Data, то таблиця буде називатися Datas. А Entity Framework автоматично розпізнає, що таблиця Datas відповідає класу Data.

Отже, створимо структуру таблиці:



Після цього, якщо ми працюємо з Visual Studio 2010, нам буде запропоновано просто ввести ім'я таблиці - введемо ім'я Books, і потім таблиця додається в БД.

А в Visual Studio Express 2012 for Web нам треба згенерувати таблицю на основі заданого вище визначення. Для цього натиснемо на кнопку Update:



Тепер, по-перше, щоб взаємодіяти з БД, нам потрібен клас контексту даних, нехай це буде наступний клас `BookContext`:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data.Entity;

namespace BookStore.Models
{
    public class BookContext : DbContext
    {
        public DbSet<Book> Books { get; set; }
        public DbSet<Purchase> Purchases { get; set; }
    }
}
```

По-друге, визначимо рядок підключення до БД. Для цього відкриємо файл `Web.config` і додамо в кінець секції `configuration` визначення рядка підключення. Однак тут треба відразу зауважити, що для Visual Studio 2010 рядок підключення відрізнятиметься від рядка підключення, яка використовується в Visual Studio 2012-2013.

Для Visual Studio 2012-2013 виглядатиме визначення рядка підключення буде виглядати наступним чином:

```
<configuration>
.....
    <connectionStrings>
        <add name="BookContext" connectionString="Data
Source=(LocalDB)\v11.0;AttachDbFilename='|DataDirectory|\Bookstore.mdf';Integrated
Security=True"
        providerName="System.Data.SqlClient" />
    </connectionStrings>
</configuration>
```

Після цих дій ми можемо звертатись до бази даних та виконувати з нею дії використовуючи мови програмування `C#`.

Щоб напевно бути впевненим, що підключення до бази даних закрито, слід викликати метод `Dispose` у контексту даних:

```
protected override void Dispose(bool disposing)
{
    db.Dispose();
    base.Dispose(disposing);
}
```

Це перевизначення версія методу `Dispose` контролера, яка викликається при знищенні об'єкта контролера. В неї поміщається виклик `db.Dispose()`, який знищує все пов'язані з контекстом даних ресурси і підключення

II. Завдання на практичну роботу

1. Додайте до свого сайту БД.
2. Створіть підключення БД до свого сайту за допомогою `ConnectionString`.
3. Створіть сторінку реєстрації користувачів, згідно вашій об'єктній області. Запис даних про користувача повинен проходити після перевірки по ключових полях.
4. Створіть форму авторизації на сайті. Авторизація повинна проходити після перевірки, чи даний користувач зареєстрований в БД.
5. Створіть форму для коментарів користувачів вашого ресурсу.

6. Дані, з однаковими параметрами повинні міститися в БД, і вивантажитися на сторінку з БД.
7. Корзина повинна оброблятися в БД(для випадків коли користувач працює з товарами).
8. Оформлення замовлення повинно зберігатися в БД.

III. Контрольні питання

1. Що таке СУБД? Для чого вона потрібна?
2. Які існують способи підключення БД до ресурсу?
3. В яких цілях підключають БД до ресурсу?
4. Що таке ConnectionString? Для чого вона потрібна?
5. Як виконуються команди над БД?
6. Що необхідно для підключення БД до сайту?
7. Яка мова використовується для виконання дій над БД?

Практична робота №6

Мова програмування JavaScript

I. Теоретичні відомості

У JavaScript можна оголошувати змінні для зберігання даних за допомогою var. Технічно можна просто записати значення і без оголошення змінної, проте через деякі причини це не рекомендується.

Разом з оголошенням можна відразу присвоїти значення: var x = 10.

Змінні, які названі ВЕЛИКИМИ_ЛІТЕРАМИ, є константами, тобто ніколи не змінюються. Зазвичай їх використовують для зручності, щоб було менше помилок.

У JavaScript є логічні значення true (істина) і false (брехня). Оператори порівняння повертають їх.

Рядки порівнюються побуквенно. Значення різних типів наводяться до числа за порівняння, за винятком суворого рівності === (! ==).

Значення null і undefined рівні == один одному та не дорівнюють нічому іншому.

В інших порівняннях (за участю ",") їх краще не використовувати, оскільки вони поведуться не як 0.

У js є 5 «примітивних» типів: number, string, boolean, null, undefined і 6-й тип – об'єкти object. Оператор typeof x дозволяє з'ясувати, який тип знаходиться в x, повертаючи його у вигляді рядка.

Для операцій над логічними значеннями в JavaScript є || (АБО), && (І) і ! (НЕ). Хоча вони і називаються «логічними», але в JavaScript можуть застосовуватися до значень будь-якого типу і повертають також значення будь-якого типу.

JavaScript підтримує три види циклів:

- while – перевірка умови перед кожним виконанням.
- do..while – перевірка умови після кожного виконання.
- for – перевірка умови перед кожним виконанням, а також додаткові налаштування.

Щоб організувати нескінченний цикл, використовують конструкцію while (true). При цьому він, як і будь-який інший цикл, може бути перерваний директивою break. Якщо на цій ітерації циклу більше нічого не потрібно робити, але повністю припиняти цикл не слід – використовують директиву continue.

Обидві директиви підтримують «мітки», які ставляться перед циклом. Мітки – єдиний спосіб для break/continue вплинути на виконання зовнішнього циклу. Мітки не дозволяють перейти в довільне місце коду, у JavaScript немає такої можливості.

Особливості вбудованих функцій

Конкретне місце, де виводиться модальне вікно з питанням – це зазвичай центр браузера, і зовнішній вигляд вікна вибирає браузер. Розробник не може на це впливати. З одного боку – це недолік, оскільки не можна вивести вікно у своєму, особливо красивому дизайні. З іншого боку, перевага цих функцій порівняно з іншими складнішими методами взаємодії саме в тому, що вони дуже прості. Це найпростіший спосіб вивести повідомлення або отримати інформацію від відвідувача. Тому їх використовують в тих випадках, коли простота важлива, а всякі «красивості» особливої ролі не грають.

JavaScript DOM

Структура документа в DOM

DOM представляє документ як ієрархію об'єктів нового типу - вузлів (node). На вершині ієрархії знаходиться вузол (об'єкт) document, який представляє весь документ. Як вузли в DOM представлено весь зміст документа: HTML елементи, атрибути цих елементів та текст HTML елементів-контейнерів. Візьмемо для прикладу невеликий фрагмент HTML документа:

```
<HTML>
<HEAD>
<TITLE>
Overview of the DOM
</TITLE>
</HEAD>
<BODY>
```

```
<H1>
Ієрархія вузлів
</H1>
<p>
На вершині ієрархії знаходиться вузол <TT>
document </TT>, який представляє DOM сам документ.
</P>
</BODY>
</HTML>
```

Розділ HEAD цього документа містить елемент TITLE із текстом. Розділ BODY містить заголовок першого рівня та параграф, у якому одне слово набрано телетайпним шрифтом.

Ієрархія вузлів визначається вкладеністю тегів один в одного і тексту всередину елементів. Текст елемента P представлений двома текстовими вузлами, оскільки він розбитий на частини елементом TT . Треба відзначити, що Internet Explorer 5 не вважає кореневий вузол document частиною цього дерева.

Очевидно, що елементи, що мають теги, що відкриває і закриває (елементи-контейнери), можуть мати діток. Текст, атрибути та елементи типу IMG, які не мають закриває тега, мати дітей не можуть. Очевидно також, що вузли типу атрибут можуть бути дітками лише вузлів типу елемент. HTML різні елементи, наприклад, P або IMG , мають різний набір допустимих для них атрибутів. У DOM вузли-елементи різного виду мають різний набір допустимих для них вузлів-атрибутів. За рідкісними винятками вони відповідають HTML 4. Списки рекомендованих атрибутів для різних елементів можна знайти у документації W3C. Проте реальні списки залежать від реалізації браузерів.

Вузли різного типу (document, елементи, атрибути та текст) мають свій набір властивостей та методів, які дозволяють через сценарії маніпулювати ними. Ряд вузлів-елементів (об'єктів) мають власні властивості та методи. Так, наприклад, об'єкт елемента TABLE має метод createTHead(). Повний набір рекомендованих властивостей та методів можна знайти у документації W3C. Треба відзначити, що Internet Explorer 5 підтримує набагато ширший набір, ніж міститься у рекомендації W3C.

При обговоренні властивостей та методів вузлів ми використовуємо наступний фрагмент документа:

```
<DIV>
<UL ID="components">
<LI>HTML</LI>
<LI>CSS</LI>
<LI>Javascript</LI>
</UL>
</DIV>
```

У DOM цьому фрагменту відповідає гілка дерева, що росте з вузла <DIV> до вузла . Тут вона розгалужується на три гілочки за кількістю вузлів (вузли-атрибути не прийнято включати до складу дерева). Кожен із цих вузлів має по одній гілці, що закінчується текстовим вузлом.

Навігація по дереву документа

Навігацію по дереву документа можна починати з будь-якого вузла-елемента, для якого ми знаємо ідентифікатор, який надається йому як значення атрибута ID . Посилання на такий вузол можна отримати за допомогою getElementById() об'єкта document . Параметр методу є ідентифікатор. Наступний рядок коду надає змінній oList посилання на наш список:

```
var oList = document.getElementById("components")
```

Стартуючи з деякого вузла, ми можемо блукати деревом у будь-якому напрямку, використовуючи ряд властивостей вузлів. Так, вузли-елементи та текстові вузли мають властивість parentNode , яка повертає посилання на батьківський вузол. Візьмемо приклад вузол (об'єкт) oList . Посилання на батьківський елемент DIV цього вузла можна отримати так:

```
var oParent = oList.parentNode
```

Вузли-елементи та текстові вузли, що є дітками деякого вузла, входять до складу колекції `childNodes` цього вузла. (Вузли-атрибути складають окрему колекцію `attributes`) До кожного з них можна звертатися за індексом масиву. Наприклад, рядок коду

```
var oItem1 = oList.childNodes[1]
```

надає змінній `oItem1` посилання на елемент `CSS` нашого списку. Саме цей елемент представлений в DOM як вузол `childNodes[1]` вузла `oList`. Перший (`childNodes[0]`) і останній елементи колекції мають спеціальні імена: `firstChild` і `lastChild`. Ці імена є властивостями батьківського вузла. Кожен з елементів колекції має властивості `previousSibling` і `nextSibling`. Ці властивості зберігають посилання на найближчих братиків елемента - попередній і наступний елементи колекції (повертають `null`, коли братиків немає). Так, елемент `childNodes[1]` є властивістю `nextSibling` елемента `childNodes[0]` і властивістю `previousSibling` елемента `childNodes[2]`. Використовуючи ці властивості, ми можемо отримати посилання на вузол `childNodes[1]` будь-яким із наступних способів:

```
oList.firstChild.nextSibling  
oList.childNodes[2].previousSibling
```

Посилання більш віддалені вузли як у горизонталі, і по вертикалі дерева формується шляхом злиття посилань на найближчих родичів за стандартними правилами об'єктно-орієнтованого програмування. Так,

```
oList.childNodes[1].firstChild
```

є посиланням на текст "CSS" елемента `CSS` нашого списку.

Зауважимо, що всі описані вище властивості вузлів (`parentNode`, `firstChild`, `lastChild`, `nextSibling` і `previousSibling`), необхідні для навігації по дереву документа, є властивостями тільки для читання. Крім них, вузли мають ще низку властивостей.

Властивості-характеристики вузлів

Властивість вузла `nodeType` (тільки для читання) повертає 1, 2 або 3 для вузлів, що відповідають тегу, атрибуту або тексту відповідно. Властивість `nodeName` (тільки для читання) повертає ім'я HTML тега, якому відповідає даний вузол, наприклад `P` для параграфу або `UL` для нумерованого списку. Для вузлів-атрибутів `nodeName` повертає назву атрибута, а тестових вузлів повертає `#text`.

Текстові вузли мають ще одну дуже важливу властивість: `nodeValue`. Ця властивість для читання та запису зберігає зміст текстового вузла. Для елементів воно повертає `null`, а атрибутів -- значення атрибута.

Створення нових вузлів

- Метод `createElement()`
- Метод `createTextNode()`

Техніку створення нових елементів обговоримо на конкретному прикладі. Ми хочемо додати до нашого списку елемент

```
<LI>XML</LI>
```

Цьому HTML елементу в DOM відповідають два вузли: вузол-елемент `` та текстовий вузол "XML". Отже, ми повинні створити два нових вузли. Нові вузли створюються за допомогою методів `createElement()` та `createTextNode()` об'єкта `document`.

Метод `createElement()`

Метод `createElement()` приймає як параметр рядок з назвою елемента тегу і створює новий HTML елемент зазначеного типу. Наприклад, рядок коду

```
var oItem = document.createElement("LI")
```

створює новий елемент списку ``, який не має змісту. Метод повертає посилання на створений ним об'єкт. Створений вище елемент `` знаходиться у пам'яті, але не входить до складу поточного документа. Для того, щоб він став частиною документа, його треба додати до існуючих вузлів документа за допомогою методів `insertBefore()` або `appendChild()`, які докладно описані нижче.

Зауважимо, що в Internet Explorer 5 можна створювати через сценарій всі елементи, крім FRAME, IFRAME та SELECT. Властивості створюваних елементів є властивостями для читання та запису.

Метод createTextNode()

Спосіб createTextNode() використовується для створення текстового вузла. Він приймає в якості параметра рядок тексту, який визначає значення властивості nodeValue текстового вузла. Наприклад, рядок коду

```
var oText = document.createTextNode("XML")
```

створює новий текстовий вузол "XML". Метод повертає посилання на створений ним об'єкт. Створений текст не входить до складу поточного документа. Для того, щоб він став частиною документа, його треба приєднати до існуючих вузлів документа за допомогою методів appendChild(), replaceNode() або insertBefore().

Якщо у Вас випадково завалалося посилання на непотрібний текстовий вузол (наприклад, oText), можна не створювати новий вузол, а скористатися вже існуючим. Для цього треба просто привласнити новий текст як значення властивості nodeValue існуючого вузла:

```
oText.nodeValue="XML"
```

Отже, ми створили два нових вузли: вузол-елемент та текстовий вузол "XML". Тепер займемося вбудовуванням цих вузлів у документ.

Редагування дерева документа

- *вставка: методи appendChild() та insertBefore()*
- *копіювання: метод cloneNode()*
- *заміщення: методи replaceChild() і replaceNode()**
- *видалення: методи removeChild() і removeNode()**
- *перестановка: метод swapNode()**
- *Додаткові методи: applyElement()* і ChildNodes()*

Примітка. Методи, позначені зірочкою*, не входять до списку рекомендованих W3C

DOM, як редактор, дозволяє копіювати, вставляти, замінювати та видаляти як окремі вузли, так і цілі гілки дерева документа. Ми почнемо із приєднання одного вузла до іншого.

Нагадаю, що у нас є два вузли: вузол-елемент та текстовий вузол "XML". Обидва вузли знаходяться в пам'яті, і ми хочемо вбудувати їх у поточний документ. Перш за все, потрібно задати текст "XML" як зміст елемента . Зробити це можна за допомогою методу appendChild().

Вставка: метод appendChild()

Цей метод додає елемент до кінця колекції childNodes вузла, який його активізував. Сам цей вузол стає батьківським вузлом для вузла, посилання на який метод приймає як параметр. Наприклад, рядок коду

```
oItem.appendChild(oText)
```

додає вузол oText до вузла oItem. Зазначимо, що метод повертає посилання на об'єкт, який він додає. У нашому випадку це об'єкт oItem.firstChild. Тепер ми маємо в пам'яті елемент (гілочку дерева з двох вузлів)

```
<LI>XML</LI>
```

Настав час вставляти цю гілочку в поточний документ. Якщо ми хочемо вставити її в кінець нашого списку, то треба, як і вище, використовувати метод appendChild() :

```
oList.appendChild(oItem)
```

Оскільки вузол oList, до якого ми приєднали вузол oItem, є частиною поточного документа, створений елемент списку також стає частиною документа. Тепер наш список має такий вигляд:

```
<UL ID="components">
<LI>HTML</LI>
<LI>CSS</LI>
<LI>Javascript</LI>
<LI>XML</LI>
</UL>
```

Обговоримо тепер як можна вставити створений нами елемент списку не в кінці, а, скажімо, після елемента списку `HTML`. Зробити це можна за допомогою методу `insertBefore()`.

Вставка: метод insertBefore()

На відміну від методу `appendChild()`, метод `insertBefore()` дозволяє вказати, в яке місце колекції `childNodes` майбутнього батьківського вузла буде вставлено новий вузол. Як випливає з назви, метод вимагає посилання на вузол, перед яким буде вставлений новий брат. Ми створимо це посилання в окремому рядку, хоча це необов'язково. Отже, код

```
var oBrother = oList.firstChild.nextSiblingoList.insertBefore(oItem,
oBrother)
```

додає в колекцію `childNodes` вузла `oList` вузол `oItem` відразу після вузла `childNodes[0]`. Як перший параметр метод `insertBefore()` приймає посилання на вузол, який ми хочемо додати, а як другий параметр - посилання на вузол, перед яким буде вставлений новий брат. Другий параметр методу є необов'язковим. Якщо батьківський вузол не має дітей, то задавати його не треба. Якщо батьківський вузол має діток, а другий параметр не заданий, то вузол, що додається, стає останнім серед діток батьківського об'єкта. Метод `insertBefore()` повертає посилання на вставлений в документ об'єкт.

Тепер наш початковий список має такий вигляд:

```
<UL ID="components">
<LI>HTML</LI>
<LI>XML</LI>
<LI>CSS</LI>
<LI>Javascript</LI>
</UL>
```

Продовжимо обговорити методи редагування дерева документа.

Копіювання: метод cloneNode()

Якщо ми хочемо скопіювати деякий вузол разом із усіма його атрибутами, то треба скористатися методом `cloneNode()`. Як параметр метод приймає вираз типу `Boolean`. Якщо воно дорівнює `false`, то копіюється лише той вузол, який активізує метод. Якщо параметр методу дорівнює `true`, то копіюється вузол разом із його нащадками. Наприклад, рядок коду

```
var oClone = oList.cloneNode(true)
```

копіює в пам'ять всю гілку дерева, що починається на вузлі `oList`, тобто весь наш список повністю. Метод повертає посилання копію вузла. Використовуючи це посилання, ми можемо надалі працювати з цією копією, наприклад, відредагувати її та вставити в документ.

Заміщення: методи replaceChild() та replaceNode()

Метод `replaceChild()` дозволяє у вузла, який його активізує, замінити одного з його дітей на нового. Посилання на новий і замінений вузли метод приймає як перший і другий параметрів, відповідно. Так, наступний фрагмент сценарію

```
var oItem = document.createElement("LI")
oItem.appendChild(document.createTextNode("JScript"))
```

```
oList.replaceChild(oItem, oList.lastChild)
```

спочатку створює елемент списку з текстом "JScript", а потім замінює їм останній елемент нашого списку. Зазначимо, що метод повертає посилання на вставлений в документ вузол.

Тепер наш список має такий вигляд:

```
<UL ID="components">
<LI>HTML</LI>
<LI>CSS</LI>
<LI>JScript</LI>
</UL>
```

Звичайно, описаний вище приклад треба розглядати лише як ілюстративний, оскільки той самий результат можна отримати набагато простіше:

```
oList.lastChild.firstChild.nodeValue="JScript"
```

Якщо хочемо замінити деякий вузол у документі іншим вузлом, треба скористатися методом `replaceNode()`. Підкреслимо, що цей метод не входить до списку рекомендованих W3C, але підтримується Internet Explorer 5. Метод `replaceNode()` видаляє з документа вузол, який його активізує, і вставляє в документ замість нього новий вузол, посилання на який приймає як параметр. Зауважимо, що вузол видаляється разом із усіма своїми атрибутами та нащадками. Так, наступний фрагмент сценарію

```
var oParagraph = document.createElement("P")
var oText = document.createTextNode("Складові частини DHTML")
oParagraph.appendChild(oText)
oList.replaceNode(oParagraph)
```

спочатку створює параграф з текстом "Складові частини DHTML", а потім замінює наш список `oList` на цей параграф. Зазначимо, що метод повертає посилання на вставлений документ вузол.

Видалення: методу `removeChild()` та `removeNode()`

Метод `removeChild()` дозволяє у вузла, який активізує, видалити одного з його діток. Посилання на видалений вузол метод приймає як параметр. Наприклад, рядок коду

```
var oRemovedItem = oList.removeChild(oList.lastChild)
```

видаляє з нашого списку останній елемент. Метод повертає посилання на видалений ним вузол. Оскільки віддалений з документа вузол залишається в пам'яті, ми можемо працювати з ним, використовуючи це посилання. Наприклад, додати до якогось іншого списку.

Якщо ми хочемо видалити певний вузол із документа, то треба скористатися методом `removeNode()`. Підкреслимо, що цей метод не входить до списку рекомендованих W3C, але підтримується Internet Explorer 5. Як параметр метод приймає вираз типу `Boolean`. Якщо він дорівнює `false`, то видаляється лише той вузол, який активізував метод. При цьому гілка дерева, що йде від нього, приєднується до його батьківського вузла. Якщо параметр методу дорівнює `true`, то вузол видаляється разом зі своїми нащадками. Наприклад, рядок коду

```
var oRemovedList = oList.removeNode(true)
```

видаляє весь наш список з документа повністю. Метод `removeNode()` повертає посилання на видалений ним вузол. Оскільки віддалений з документа вузол залишається в пам'яті, ми можемо працювати з ним, використовуючи це посилання.

Використовувати `false` як параметр треба осмислено. Так, якщо ми застосуємо метод `removeNode()` з параметром `false` до нашого списку, його дітки повинні будуть перейти до елемента `DIV`. Але, згідно з вимогами HTML 4, елементи списку `LI` не можуть розміщуватись у цьому контейнері!

Перестановка: метод `swapNode()`

Якщо ми хочемо переставити два вузли в документі (загалом дві гілки дерева), то треба скористатися методом `swapNode()`. Підкреслимо, що цей метод не входить до списку рекомендованих W3C, але підтримується Internet Explorer 5. Цей метод змінює місцями вузол, який його активізував, і вузол, посилання на який приймає як параметр. Наприклад, рядок коду

```
var oSwappedNode = oFirst.swapNode(oSecond)
```

переставляє вузли `oFirst` та `oSecond`. Повертає посилання вузол, який активізує метод.

Невеликий відступ для шанувальників Internet Explorer: W3C DOM не підтримує відомі нам ще з Internet Explorer 4 дуже зручні для копіювання та вставки різних фрагментів документа такі властивості об'єктів, як `innerHTML`, `innerText`, `outerHTML` та `outerText`. Багато дій за допомогою цих властивостей програмуються набагато простіше, ніж у рамках W3C DOM. В якості ілюстрації нижче наведено простий приклад використання властивості `outerHTML`. Коли користувач клацає на кнопці передачі форми на сервер, замість кнопки з'являється повідомлення з подякою:

```
<INPUT TYPE="submit"
      VALUE="Надіслати"
      onClick="this.outerHTML='Дякую Вам за участь у нашому
опитуванні.' ">
```

А ось як цю ж дію можна реалізувати, використовуючи лише засоби W3C DOM:

```
<INPUT TYPE="submit" VALUE="Надіслати"
      onClick="replaceButton(this) ">
<SCRIPT TYPE="text/javascript">
    function replaceButton(oButton) {
        var oText = document.createTextNode("Дякую Вам за
участь у нашому опитуванні.")
        var oParent = oButton.parentNode
        oParent.replaceChild(oParagraph,oButton)
    }
</SCRIPT>
```

Крім розглянутих нами методів редагування дерева документа є ще два методи, про які необхідно сказати.

Метод `applyElement()`

Описані вище методи `appendChild()` і `insertBefore()` дозволяють вузлу заводити дітей. Internet Explorer 5 підтримує також метод, який дозволяє вузлу-елементу мати батька. Таким методом є метод `applyElement()`, який приймає як параметр посилання на майбутнього батька. Наприклад, рядок коду

```
var oParent = oChildNode.applyElement(oParentNode)
```

задає вузол `oParentNode` як батьківський для вузла `oChildNode`. При цьому, вузол `oParentNode` позбавляється (якщо вони у нього були) як свого батька, так і своїх дітей перед тим, як "придбати" нового спадкоємця. Метод повертає посилання нового батька. Підкреслимо, що метод `applyElement()` не застосовується до текстових вузлів.

Метод `hasChildNodes()`

Дізнатися, чи є у вузла дітки, можна за допомогою методу `hasChildNodes()`. Наприклад, рядок коду

```
oTestedNode.hasChildNodes()
```

повертає `true`, коли вузол `oTestedNode` має дітей, і `false` в іншому випадку.

Робота з атрибутами елементів

- Метод `createAttribute()`
- Метод `setAttribute()`
- Метод `removeAttribute()`
- Метод `getAttribute()`
- Додаткові методи в *Internet Explorer 5*

Всі атрибути вузла-елемента (за винятком атрибуту `STYLE`) складають колекцію `attributes`. W3C визначає цю колекцію як масив із доступом за іменами елементів, наприклад, `oNode.attributes.align` або `oNode.attributes["align"]` повертає значення атрибуту `ALIGN` вузла `oNode`. (У документації Microsoft по DHTML сказано, що до елементів колекції треба звертатися за індексом масиву. Але на практиці працює описане вище звернення за іменами. Ймовірно, документація оновлюється рідше, ніж з'являються нові версії браузера.) Ім'я атрибута треба набирати великими літерами незалежно від того, в якому вигляді. Властивість `nodeName` для вузлів-атрибутів повертає назву атрибута, а `nodeValue` - значення атрибута. Властивість атрибутів `specified` дозволяє дізнатися, чи визначено цей атрибут. Якщо, наприклад, у вузла `oNode` атрибут `ALIGN` визначено, то

```
oNode.attributes["align"].specified
```

повертає `true`, а якщо не визначено, то `false`. (Така поведінка якості `specified` реалізована і в *Internet Explorer 5* всупереч документації Microsoft по DHTML)

Метод `createAttribute()`

Метод `createAttribute()` об'єкта `document` дозволяє створити вузол-атрибут. Як параметр метод приймає ім'я атрибута. *Internet Explorer 5* не підтримує цей метод.

Метод `setAttribute()`

Атрибути як нових елементів, і тих, що задані через HTML, можна задати або традиційним способом - присвоюючи значення властивості (атрибуту) вузла, або з допомогою методу `setAttribute()`. Обидва способи демонструються нижче для атрибуту `ID`. Так, будь-який з рядків коду

```
oNode.id= "newItem"  
oNode.setAttribute("id", "newItem")
```

задає для елемента `oNode` як ідентифікатор рядок `"newItem"`. Метод `setAttribute()` потребує два параметри. Перший параметр - рядок, який визначає назву атрибута. Другий параметр - рядок, число або булевий вираз відповідає значенню атрибута. Згідно з документацією в *Internet Explorer 5*, за замовчанням назви атрибутів чутливі до регістру, в якому вони набрані. Якщо набір малих і великих літер не співпадає з використаним раніше в назві атрибута, буде створено новий атрибут. Залежність від регістру можна скасувати, задаючи як третій параметр методу число `0`.

Метод `removeAttribute()`

Видалити атрибут у елемента можна, активізуючи метод `removeAttribute()`, який вимагає як параметр назву цього атрибута. Якщо віддалений атрибут має значення за замовчанням, воно відновлюється. Відповідно до документації в *Internet Explorer 5*, за замовчуванням у параметрі набір малих і великих літер повинен збігатися з використаним раніше у назві атрибута. Залежність від регістру можна скасувати, задаючи як другий параметр методу число `0`. Метод повертає `true` при успішному виконанні дії та `false` в іншому випадку.

Метод `getAttribute()`

Дізнатися поточне значення атрибута елемента можна, активізуючи метод `getAttribute()`, який вимагає як параметр назва цього атрибута.

Додаткові методи в *Internet Explorer 5*

Метод `clearAttributes()` видаляє всі атрибути об'єкта, який активізує цей метод. Не потребує параметрів. Повертає посилання на цей об'єкт.

Метод `mergeAttributes()` дозволяє скопіювати всі атрибути об'єкта, посилання на який він приймає як параметр, і додати ці атрибути до об'єкта, який активізував цей метод. Наприклад, рядок коду

`oTargetNode.mergeAttributes(oSourceNode)`

задає ті ж атрибути для вузла `oTargetNode`, що й у вузла `oSourceNode`. Метод повертає посилання на вузол, що його активізував. Обидва вузли (`oTargetNode` та `oSourceNode`) повинні відповідати однаковим HTML-елементам, наприклад, елементам `H1` або `P` .

II. Завдання на практичну роботу

1. Провести роботу зі змінними:

- оголосити дві змінні: `admin` і `name`;
- записати в `name` рядок "Васьок";
- скопіювати значення з `name` в `admin`;
- вивести `admin` (має вивести «Васьок»).

2. Додати сторінку, яка запитує ім'я і виводить його.

3. Написати функцію `min(a,b)`, яка повертає найменше з чисел `a,b`.

4. Створити код JavaScript для перетворення одиниць, який за величиною температури за шкалою Фаренгейта обчислює величину температури в градусах Цельсія і навпаки. Веб-сторінка має два текстових поля:

- температура за Фаренгейтом
- температура за Цельсієм

які містять відповідні числа.

Зовнішній вигляд:

Температура за Фаренгейтом

Температура за Цельсієм

Сигналом для початку обчислення служить зміна числа в текстовому полі. Температура за Фаренгейтом (F) і температура за Цельсієм (C) зв'язані формулою $C = 5/9 * (F - 32)$

5. Створити код JavaScript для перегляду зображень, який в параметрах веб-сторінки отримує імена файлів, як масив у форматі JSON(наприклад,

```
var a=['file1.jpg', 'file2.gif', 'file34.gif']
```

) та показує одне вибране зображення. Вибір зображень відбувається за допомогою натискання на іконки – зменшені копії зображень.

Зовнішній вигляд веб-сторінки:



6. Створити JavaScript, який перевіряє знання таблиці множення. Веб-сторінка містить текстовий напис для показу загального рахунку, кнопку «наступне завдання», текстовий напис для показу завдання, текстове поле для вводу відповіді, кнопку «перевірити» та текстовий напис для виводу результатів перевірки.

Зовнішній вигляд веб-сторінки:

Загальний рахунок 90% (9 правильних відповідей з 10)

наступне завдання

3 × 4 =

Помилка, правильна відповідь «12»

7. Реалізувати за допомогою JavaScript "перевірку на людяність": код javascript випадковим чином вибирає число з 2 цифр та відображає його у вигляді набору "пікселів" а потім вимагає від користувача ввести це число і перевіряє правильність вводу. Для кожної цифри заздалегідь задано свою множину "пікселів" (кожен має свої координати). Після вибору числа відповідні цифрам набори "пікселів" показуються у випадковій послідовності.

Наприклад:

Введіть число

Помилка

Поради

Посилання на об'єкт DOM найпростіше отримати за допомогою функції `document.getElementById();`

Наприклад, якщо в коді HTML ви маєте фрагмент

```
<a href="http://www.znu.edu.ua" id="mylink">visible text</a>
```

то відповідний об'єкт DOM можна отримати як

```
var mylink=document.getElementById('mylink');
```

Отримавши посилання на об'єкт DOM можна отримати значення атрибута:

```
var mylinkHref=mylink.href;
```

або встановити значення атрибута:

```
mylink.href="http://www.znu.edu.ua/contacts";
```

Видимий текст всередині мітки встановлюється і читається з допомогою атрибута `innerHTML`:

```
var oldText=mylink.innerHTML;
```

```
mylink.innerHTML='new text';
```

Особливим атрибутом є `style`, який керує властивостями CSS. Наприклад, продовжуючи початий вище приклад, можна встановити колір шрифту і тла:

```
mylink.style.color='red';
```

```
mylink.style.backgroundColor='yellow';
```

III. Контрольні питання

1. У чому полягає суть тегу SCRIPT?
2. Описати сучасну розмітку для тегу SCRIPT.
3. Що таке зовнішні скрипти?
4. Види циклів в JavaScript.
5. Як організувати нескінченний цикл?
6. Що таке DOM ?
7. Як отримати посилання на вузол документа?
8. Властивості вузлів.
9. Які методи редагування дерева документа Ви знаєте?
10. Які методи роботи з атрибутами елементів Ви знаєте?

Додатки

Додаток А Набір тем для сайту

Для виконання практичної роботи студент створює сайт відповідної тематики, що містить 3-4 змістовних сторінок та сторінку автора сайту. Нижче наведено перелік тем, які можна використати або реалізувати власну ідею.

1. Портфоліо робіт певного фахівця (дизайнер, фотограф, відео-оператор, кулінар тощо).
2. Послуги умовної компанії (бізнес, медицина, перевезення, розваги, готелі, кафе тощо).
3. Візитка відомої особи: письменника, спортсмена, актора, співака, музичного гурту, політичного діяча.
4. Галерея творчих робіт: фотографії, картини, книги, пісні чи інше.
5. Новинки технічних досягнень: комп'ютери, телефони, автомобілі, побутова техніка, роботи.
6. Міста та країни світу. Улюблені краєвиди, історичні екскурси, цікаві відомості.
7. Хобі студента: спорт, творчість, навчання, домашні тварини та інше.
8. Паблік для популярного фільму, серіалу, мюзіклу чи творчої спільноти.
9. Електронна бібліотека із збіркою певних книжок, статей, коміксів тощо.
10. Домашня сторінка студента, де викладено його захоплення, мандрівки, фотографії друзів, домашніх тварин та інша відкрита інформація.

Додаток Б

Таблиця спеціальних символів HTML

Ім'я	Код	Вид	Опис
 sp	 		нерозривний пробіл
£	£	£	фунт стерлінгів
&euro	€	€	знак євро
¶	¶	¶	символ параграфа
§	§	§	параграф
©	©	©	знак соруіght
®	®	®	знак зареєстрованої торгової марки
&trade	™	™	знак торгової марки
°	°	°	градус
±	±	±	плюс-мінус
¼	¼	¼	дріб – одна четверта
½	½	½	дріб – одна друга
¾	¾	¾	дріб – три четвертих
×	×	×	знак множення
÷	÷	÷	знак ділення
&fnof	ƒ	ƒ	знак функції
Грецькі букви			
&Alpha	Α	Α	грецька велика буква альфа
&Beta	Β	Β	грецька велика буква бета
&Gamma	Γ	Γ	грецька велика буква гамма
&Delta	Δ	Δ	грецька велика буква дельта
&Epsilon	Ε	Ε	грецька велика буква епсилон
&Zeta	Ζ	Ζ	грецька велика буква дзета
&Eta	Η	Η	грецька велика буква ета
&Theta	Θ	Θ	грецька велика буква тета
&Iota	Ι	Ι	грецька велика буква йота
&Kappa	Κ	Κ	грецька велика буква каппа
&Lambda	Λ	Λ	грецька велика буква лямбда
&Mu	Μ	Μ	грецька велика буква мю
&Nu	Ν	Ν	грецька велика буква ню
&Xi	Ξ	Ξ	грецька велика буква ксі
&Omicron	Ο	Ο	грецька велика буква омікрон
&Pi	Π	Π	грецька велика буква пі
&Rho	Ρ	Ρ	грецька велика буква ро
&Sigma	Σ	Σ	грецька велика буква сигма
&Tau	Τ	Τ	грецька велика буква тау
&Upsilon	Υ	Υ	грецька велика буква іпсилон
&Phi	Φ	Φ	грецька велика буква фі
&Chi	Χ	Χ	грецька велика буква хі
&Psi	Ψ	Ψ	грецька велика буква псі
&Omega	Ω	Ω	грецька велика буква омега
&alpha	α	α	грецька мала буква альфа
&beta	β	β	грецька мала буква бета
&gamma	γ	γ	грецька мала буква гамма
&delta	δ	δ	грецька мала буква дельта
&epsilon	ε	ε	грецька мала буква епсилон
&zeta	ζ	ζ	грецька мала буква дзета

&eta	η	η	грецька мала буква ета
&theta	θ	θ	грецька мала буква тета
&iota	ι	ι	грецька мала буква йота
&kappa	κ	κ	грецька мала буква каппа
&lambda	λ	λ	грецька мала буква лямбда
&mu	μ	μ	грецька мала буква мю
&nu	ν	ν	грецька мала буква ню
&xi	ξ	ξ	грецька мала буква ксі
&omicron	ο	ο	грецька мала буква омікрон
&pi	π	π	грецька мала буква пі
&rho	ρ	ρ	грецька мала буква ро
&sigmaf	ς	ς	грецька мала буква сигма
&sigma	σ	σ	грецька мала буква сигма
&tau	τ	τ	грецька мала буква тау
&upsilon	υ	υ	грецька мала буква іпсилон
&phi	φ	φ	грецька мала буква фі
&chi	χ	χ	грецька мала буква хі
&psi	ψ	ψ	грецька мала буква псі
&omega	ω	ω	грецька мала буква омега
Стрілки			
&larr	←	←	стрілка вліво
&uarr	↑	↑	стрілка вверх
&rarr	→	→	стрілка вправо
&darr	↓	↓	стрілка вниз
&harr	↔	↔	стрілка вліво-вправо
Інші символи			
&spades	♠	♠	
&clubs	♣	♣	
&hearts	♥	♥	
&diamonds	♦	♦	
"	"	"	подвійна лапка
&	&	&	амперсанд
<	<	<	знак "менше"
>	>	>	знак "більше"
Знаки пунктуації			
&hellip	…
&prime	′	'	мінути і фути
&Prime	″	"	секунди і дюйми
Загальна пунктуація			
&ndash	–	–	тире
&mdash	—	—	довге тире
&lsquo	‘	‘	
&rsquo	’	’	
&sbquo	‚	‚	
&ldquo	“	“	
&rdquo	”	”	
&bdquo	„	„	
«	«	«	
»	»	»	

Перелік літератури

1. HTML конструювання і сайтобудівництво для початківців і новачків. [Електронний ресурс]. Режим доступу: <http://htmlbook.in.ua/>.
2. Сучасний підручник з JavaScript. [Електронний ресурс]. Режим доступу: <https://uk.javascript.info/>.
3. HTML in Easy Steps / Mike McGrath // книга – 2020. – 192 с. – ISBN - 9781840788761.
4. JavaScript in Easy Steps / Mike McGrath // книга – 2013. – 192 с. – ISBN - 9781840785708.
5. Лекційні матеріали з ВЕБ-технологій та дизайну. [Електронний ресурс]. Режим доступу: <https://www.victoria.lviv.ua/library/students/wd/index.html>.
6. Лекційні матеріали «Сучасні методи веб-програмування». [Електронний ресурс]. Режим доступу: <http://sites.znu.edu.ua/webprog/lect/1169.ukr.html>.