Міністерство освіти та науки України
Чернівецький національний університет
імені Юрія Федьковича

Т.І. Готинчан

Основи веброзробки: HTML і CSS

Частина 1

Навчальний посібник

Чернівці
Чернівецький національний університет
2023

УДК 004.774(075.8)
Г733

Рекомендовано до друку Вченою радою факультету математики та інформатики
Чернівецького національного університету імені Юрія Федьковича
(протокол № 7 від «20» лютого 2023 року)

Рецензенти:
Перцов Андрій Сергійович, канд. фіз.-мат. наук, Team Lead, Lead Software Engineer, IT компанія «Epam»;
Мельник Василь Сергійович, канд. фіз.-мат. наук, Full Stack Software Engineer, ТОВ «Українські Інформаційні Технології».

Г733 Готинчан Т.І. Основи веброзробки: HTML і CSS. Частина 1 : навчальний посібник / Т.І. Готинчан – Чернівці : Чернівецький національний університет, 2023. – 208 с.

Видання містить навчальний матеріал з основ веброзробки, які охоплюють основні принципи вступу до веброзробки та мови розмітки HTML.

Для студентів спеціальностей «Комп'ютерні науки» та «Системний аналіз» та інших технічних спеціальностей.

УДК 004.774(075.8)

Зміст

Вступ

Виклики сьогодення та потреби людства приносять щось нове для кожної сфери його діяльності, особливо для інтернет технологій, оскільки ми живемо в епоху цифровізації.

Цифровізація – це впровадження сучасних цифрових технологій у різні сфери життя та виробництва. Застосовується цифровізація поступово у побуті, бізнесі, промисловості, на виробництві, а також у державних структурах та державному управлінні, в охороні здоров'я, у системі освіти та науки, у житті у міста тощо.

Процеси цифровізації сприяють й розвитку веброзробки. З'являються нові платформи, мови програмування, фреймворки, старі пристрої замінюються новими. Проте необов'язково одразу використовувати кожен новий тренд, оскільки не завжди нове означає "краще". Треба зважено вивчити переваги і недоліки нових інструментів та зробити вибір.

Якісна веброзробка дозволяє розширити та покращити функціонування сфери життя та виробництва. Веброзробка здійснюється фахівцями з урахуванням їхнього професійного досвіду та нових інструментів розробок. Слід прагнути того, щоб створений вебресурс був зручним для користувача та вирішував поставлені завдання та функції.

Веброзробка дозволяє реалізувати такі функції як інформаційну, комукативну, маркентингову.

Інформаційна функція полягає в тому, щоб надати користувачеві якомога повнішу інформацію за його запитом. Будь-то співробітник, тоді інформація пов'язана з його професійною діяльністю, а якщо це відвідувач сайту фірми, яка пропонує про послуги або продукцію, то інформація про них. Недолік повного спектру необхідної інформації – це серйозна помилка, яка призводить до того, що працівникам потрібно затратити більше часу на отримання потрібної інформації, а користувачі, які прийшли на сайт, не стають клієнтами.

Комунікаційна функція покликана налагодити канали спілкування співробітників між собою та з клієнтами. Для того, щоб процес спілкування проходив найбільш зручно, у вебпродукті має бути створене комунікаційне середовище. Це можуть бути такі зручні інструменти зворотного зв'язку як форми зворотного зв'язку і прямий онлайн чат, На більших проектах також бути використані масштабніші інструменти комунікації, такі як форум, корпоративний блог, присутність в соціальних мережах. Чим краще розвинена комунікаційна функція вебпродукту, тим продуктивніша робота співробітника, тим простіше налагодити контакт із клієнтом, і тим охочіше він піде на угоду.

Маркетингова функція відповідає за продаж продукції чи послуг. Це одна з головних функцій, що дозволяє його власникам отримувати постійний прибуток. Тому, по-перше, потрібно до відвідувача донести інформацію про переваги послуг та продукцію перед конкурентами, а, по-друге, важливо створити зручний інтерфейс придбання. Має бути створений зручний сервіс, який допоможе користувачу пройти поступово усі етапи цієї процедури. З цього випливає, що маркетингова функція покликана переконати відвідувача купити і зробити так, щоб процес придбання пройшов легко та комфортно. І це стосується не лише бізнес структур, а й інших установ. Будь-то сфера освіти із сервісом для абітурієнтів. Або сфера закладів охорони здоров’я із сервісом запису до лікаря на консультацію.

Розробка вебпродукту, який виконуватиме усі три функції, значно вигравatиме перед конкурентними розробками у своїй сфері задіювання.

Сучасна веброзробка ділиться на 3 напрямки:

- *Frontend* – створення інтерфейсу, видимої частини додатків та сайтів, з якою взаємодіють користувачі. Інтерфейс – посередник між програмою та клієнтом. За якість роботи цих елементів відповідає фронтенд-розробник.

- *Backend* – відповідає за внутрішню або серверну частину програмного продукту. Завдання backend-розробника полягає у тому, щоб продукт коректно обробляв запити, виводив інформацію, яку запросив користувач.

- *Fullstack-розробники* – це універсальні веброзробки, які розуміюються як у frontend-, так і в backend-розробці.

Передумовою для того, щоб стати професійним веброзробником є знання HTML (мова розмітки гіпертексту), CSS (каскадні таблиці стилів) і JavaScript. HTML, CSS і JavaScript – це мови, які відкривають світ веброзробки. Вони дуже тісно пов'язані між собою, але розроблені для дуже конкретних завдань. Розуміння того, як вони взаємодіють, допоможе стати веброзробником.

HTML призначена для додавання значення необробленому вмісту шляхом його розмітки.

CSS призначений для форматування розміченого вмісту та додавання деякої можливої інтерактивності.

JavaScript призначений для того, щоб зробити вміст і форматування інтерактивними. JavaScript може маніпулювати як HTML, так і CSS

HTML, CSS і JavaScript є лише передумовою для того, щоб стати професійним веброзробником. Веброзробники виконують значно більше різноманітних завдань, щоб розроблений вебпродукт запрацював.

Вільне володіння HTML і CSS є важливим першим кроком до того, щоб стати справжнім веб-розробником.

Навчальний посібник складається з трьох розділів. Перший розділ присвячений загальному огляду процесу веброзробки. У другому розділі наведено основні поняття мови розмітки HTML а у третьому – CSS. Кожний підрозділ завершується переліком питань на повторення, що сприяє перевірці засвоєного матеріалу.

Вступ до веброзробки

Основні поняття і етапи веброзробки

Веброзробка – це процес створення вебпродукту (онлайн-сервісів, вебсайту, вебдодатку). Виділяють такі етапи розробки вебпродукту [1]:

- проектування;
- розробка концепції;
- створення макетів сторінок;
- створення мультимедіа, об'єктів та зображень;
- верстка шаблонів та сторінок;
- програмування або інтеграція у систему керування змістом;
- оптимізація змісту;
- тестування та внесення правок;
- розгортання проєкту на хостингу;
- підтримка роботи вебсайту або вебдодатку.

Проте у залежності від сукупності та рівня складності завдань деякі етапи можуть бути відсутніми або переплітатись.

Розробка вебпродуктів починається з правильної постановки завдання. Необхідно і важливо вирішити, які завдання постають перед майбутнім ресурсом, які саме результати роботи очікуються у підсумку. Важливо вже на початкових етапах розробки досить точно визначити, які дії має зробити користувач, який відвідав ресурс та ознайомився з його вмістом.

Чітко поставлена мета допоможе створити вебпродукт максимально ефективним та незамінним маркетинговим інструментом.

Проектування вебпродукту полягає у аналізі вимог до проєкту як з боку візуальної частини, так і функціональної. Бажано чітко обумовити із замовником усі аспекти майбутнього ресурсу: задачі, функціональність, дизайн, обсяг змісту, тощо.

На підставі аналізу даних і вимог розробляється концепція майбутнього вебпродукту та складається і затверджується технічне завдання. У технічному завданні мають бути зазначені:

- призначення та аудиторія вебпродукту;
- технічні характеристики;
- вимоги до матеріалу;
- структура сайту з детальним описом усіх елементів та їх призначення на кожній його сторінці;
- наявність інтерактивних елементів, таких як зворотній зв'язок, пошук, форум тощо;
- наявність форм авторизації, підписки, опитування тощо;
- розгортання на хостингу.

У залежності від сукупності та рівня складності задач у технічному завданні додаються й інші матеріали, що стосуються вимог до вебпродукту.

Візуальна частина вебпродукту розпочинається зі створення концепції дизайну. Зазвичай розробляється декілька концепцій дизайну відповідно до технічного завдання. Після затвердження основної концепції розробляються шаблони усіх сторінок вебпродукту. Варіантів дизайну також може бути декілька. Вони вирізняються не лише гамою кольорів, але й розташуванням елементів. Продумується усе до дрібниць із врахуванням можливостей верстки. Власне цим і вирізняється вебдизайн від загального арт. Вебдизайнер має враховувати обмеження стандартів верстки і не створювати макети, які неможливо стандартними засобами реалізувати. Він не зобов’язаний знати верстку, але принципи її – так.

Вебдизайнер розробляє макет кожної сторінки під різні типи пристроїв. Зазвичай, це мобільні пристрої, планшети, та настільні комп'ютери. З врахуванням розміру пристрою розташування одних і тих самих елементів відрізняється на відповідних макетах сторінки. Створюють дизайн сторінок із врахуванням досвіду користувача.

Головним правилом є дотримання послідовності дизайну – будь-який сайт повинен створити довірчі відносини з користувачем, щоб він відчував себе комфортно при переміщенні по ньому та взаємодії з ним. Послідовний дизайн означає, що при переході з однієї сторінки сайту на іншу у користувача не повинно виникати відчуття, що він потрапив на інший сайт. А тому слід використовувати однакову колірну схему по всьому сайту, повторювати одні й ті ж елементи для різних ситуацій наприклад, один і той самий дизайн спливаючих повідомлень, модальних вікон, форм. Бажано задавати однакові шрифти, стилі і розміри заголовків, тексту, оформлення карток на різних сторінках.

Створюють макети сторінок у графічному редакторі. Деякі з них працюють лише під однією операційною системою, інші – у браузерах, а деякі можуть працювати як онлайн чи локально під будь-якою системою.

Розглянемо деякі з них [2]:

- **Adobe Photoshop** – найстаріший і найпопулярніший редактор, який використовувався протягом багатьох років. Проте на сьогодні він починає втрачати свою популярність як редактор для створення макеків вебсторінок через складність у використанні. Редактор працює під операційні системи Microsoft Windows, macOS та в мобільних версіях iOS, Windows Phone і Android. Наразі офіційної версії для систем Linux немає. Проте для версії CS і CS6 можливий запуск застосунку під Linux за допомогою альтернативи Widows API-Wine.
- **Gimp –** аналог Photoshop для систем Linux. Він має аналогічну функціональність. Проте, на жаль, не можна гарантувати правильне прочитання файлів, створених у одному редакторі, що відкриваються у іншому. Photoshop і Gimp – не є одним і тим самим редактором під різні операційні системи. Gimp також як і Photoshop не є спеціалізованим інструментом для створення вебмакетів, тому для вебдизайнерів він переобтяжений багатьма функціями.

- **Sketch** – спеціалізований редактор для розробки мобільних і вебінтерфейсів. Sketch надає необхідні інструменти для спільного процесу проектування – від ранніх ідей до піксельно-ідеальних ілюстрацій, ігрових прототипів і передачі їх веброзробникам. Редактор пропонує понад 700 розширень: плагінів, помічників, інтеграцій, які полегшують процес створення вебмакетів. Зручний він і для верстальників. Вони можуть гортати макети, переглядати розміри об'єктів, відступи між ними, шрифти та іншу інформацію, яка переноситься з макету. Редактор працює лише під операційну систему macOS і не анонсовано його перенесення на інші операційні системи. Є онлайн-робочий простір для зберігання та обміну розробкою.

- **Figma** – як і Sketch, це редактор, орієнтований на створення макетів інтерфейсів веброзробки. Працює під операційні системи Windows і macOS. Для систем Linux існують спеціальні варіанти встановлення для різних типів систем. Можна працювати як у браузері онлайн, так і встановити окремий додаток. Дані синхронізуються. Останнім часом редактор Figma набула своєї надзвичайної популярності не лише серед вебдизайнерів, але й маркетологів, менеджерів, бізнес-аналітиків тощо із-за зручності роботи з ним і можливістю створення інтерактивних елементів та демонстрації їх у дії. Наприклад, вебдизайнер розробляє прототипи сторінок вебпродукту, UX копірайтер пише тексти для інтерфейсу, проєкт-менеджер демонструє замовнику дизайн майбутнього вебпродукту, замовник залишає коментарі. Та й верстальникам у ньому працювати зручно, оскільки можна прогортати макети, продивлятись інформацію про об'єкти, які треба перенести у веброзробку.

Figma зручна як для створення багатосторінкових, так і односторінкових сайтів.

Аналізуючи макети сторінок, верстальник створює шаблони вебсторінок або її частин. Створюються HTML-розмітка та CSS-стилі елементів верстки. Слід зазначити, що на цьому етапі напов-

нення деяких елементів може бути шаблонним, а вже пізніше після розгортання вебпродукту на сервері відбуватиметься наповнення реальним вмістом – зображеннями, текстами, файлами для скачування тощо.

Працює верстальник зазвичай з редакторами коду. Краще за все використовувати спеціальні редактори HTML та CSS. Їх називають HTML-редакторами. Сучасні HTML-редактори мають безліч убудованих механізмів, які істотно спрощують розробку вебсторінок. Це – виділення синтаксичних конструкцій, вставка цілих конструкцій елементів html коду, підказка та механізми автозаповнення, перевірка помилок, зручне переміщення по коду тощо. Ці та багато інших механізмів сучасних HTML-редакторів значно полегшують роботу верстальників.

Є два основні типи редакторів код – текстові та візуальні.

Текстові редактори передбачають написання коду розмітки та каскадних таблиць стилю вручну. Це класичний варіант для тих, хто знайомий з HTML-розміткою, знає теги, CSS-правила, вміє працювати з контентом усередині сторінки та розуміє, як його оформляти.

Візуальні редактори нагадують конструктор, який оперує блоками. Переміщуючи їх, користувач може «збирати» повноцінний сайт, не написавши і рядка коду.

Кожен з HTML-редакторів має свій функціонал, свої особливості використання, свої переваги та недоліки. Розглянемо деякі текстові редактори [3]:

- **Visual Studio Code (VS Code)** – потужний безплатний інструмент, який можна використовувати не тільки для верстки, а й для програмування багатьма мовами. VS Code має частину функціоналу інтегрованого середовища розробки IDE (Integrated development environment), яка містить, окрім текстового редактора коду, ще ряд механізмів, що дозволяють проводити аналіз коду, запуск його та налагодження.

Переваги:
- відкритий вихідний код програми;
- має значну частину функціоналу IDE;
- вбудований механізм автозаповнення IntelliSense;
- автоматичний пошук помилок та їх виправлення;
- візуальне покращення коду вбудованими засобами;
- значна кількість розширень та доповнень;
- вбудований клієнт Git, що дозволяє завантажувати проект у GitHub прямо з компілятора, не використовуючи окремий термінал;
- вбудований налагоджувач для JavaScript, TypeScript, Node.js;
- плагін Live Server для перегляду вебдодатків та сайтів у реальному часі.

Недоліки:
- досить великий час запуску програми;
- відносно повільно відбувається пошук за проектами.

- **Sublime Text** – високопродуктивний редактор. Компілятор має функцію миттєвого перемикання між проєктами. Завантаження проєкту відбувається значно швидше, ніж у VS Code. Програма постачається безплатно з деякими обмеженнями. Проте щоб скористатися усіма функціями редактора доведеться купити ліцензію.

Переваги:
- працює у таких операційних системах як Windows, OS X та Linux;
- має онлайн та портативні версії;
- швидкий, легкий, не перевантажує систему;
- надає багато різноманітних доповнень з відкритим вихідним кодом, які створені великою спільнотою;
- розробники можуть використовувати кілька моніторів та редагувати різні ділянки коду одночасно.

Недоліки:

- не весь функціонал, доступний користувачеві, безплатний;
- певна незручність роботи з менеджером плагінів;
- низка плагінів сторонніх розробників може працювати некоректно.

- **Notepad++** – це легкий зручний перевірений часом та усталений безплатний текстовий редактор, розроблений для комп'ютерів під керуванням Windows. Під Linux і macOS працює через емулятори, зокрема користувачі Linux можуть використовувати його через Wine.

Переваги:

- є простим у користуванні, не вимогливим до ресурсів інструментом;
- інтерфейс редактора легко налаштовується
- має портативну та мобільну версії;
- функціонал програми можна розширити безліччю плагінів;
- підтримується робота з великою кількістю вкладок одночасно.

Недоліки:

- редактор не є IDE і не містить у собі її додатковий функціонал, доводиться використовувати якесь середовище розробки на додаток до редактора;
- мінімалістичність інтерфейсу, доводиться під'єднувати плагіни.

- **WebStorm** – зручне для веброзробки середовище розробки. Якщо створити проект у редакторі, то програма увесь вміст вважає папкою проєкту, що доволі зручно при роботі з ним.

Переваги:

- має функціонал інтегрованого середовища розробки IDE;
- автодоповнення як коду на HTML, CSS, так і на JavaScript,

- гнучкість налаштувань;
- велика кількість плагінів,
- наявність системи відстеження помилок;
- вбудована інтеграція із системами керування версіями такими як GitHub, Git, Subversion, Perforce та Mercurial.

Недоліки:
- властива всім IDE повільність у роботі та вимогливість до ресурсів;
- платна IDE, що розповсюджується за передплатою;
- складність налаштувань.

- **Brackets** – безплатний кросплатформний редактор для роботи з HTML, CSS і Javascript. Цей редактор створювався для роботи вебдизайнерів та розробників під операційну систему MacOs, проте може працювати і на інших операційних системах. Однією з найкращих функцій Brackets є наявність мініредактора, убудованого у основний код. Він з'являється лише в ті моменти, коли потрібний користувачеві. Наприклад, якщо потрібно редагувати HTML-конструкцію з певним класом, редактор пропонує відкрити вікно з кодом CSS для вибраного класу та внести туди зміни, не залишаючи головну сторінку. Таким чином можна редагувати відразу декілька файлів, де є використання певного класу, не перемикаючись між вікнами та вкладками.

Переваги:
- кросплатформність – працює на операційних системах Windows, MacOs, Linux;
- розвинута система гарячих клавіш;
- наявність режиму Live Preview, що дозволяє у реальному часі спостерігати за змінами, які вносяться до вмісту сторінки;
- вбудована підтримка препроцесорів SCSS та LESS з усіма їх особливостями та додатковими функціями;

- наявність функції, яка дає змогу отримувати інформацію безпосередньо із файлу PSD: розміри об'єктів, кольори, шрифти одразу з CSS та без контексту.

Недоліки:
- мало розширень у порівнянні з іншими HTML редакторами;
- відсутність підтримки серверних мов.

Крім редакторів кодів є і середовища для розробки. Це програми, створені для професійних розробників, що містять у собі весь спектр інструментів, необхідних для створення сайту або програми з нуля.

Написанням лише HTML-розмітки та CSS-коду для опрацювання дизайну сторінок верстка вебпродукту не обмежується. Верстальник також здійснює:
- відбір зображень, іконок та інших графічних матеріалів з макета, їх оптимізацію, компонування та сортування за папками;
- під'єднання необхідних шрифтів та налаштування їх коректного відображення;
- підключення JS-бібліотек для створення динамічних елементів;
- тестування та валідація верстки.

Оптимізацію безпосередньо у браузері растрових зображень можна здійснювати, наприклад, за допомогою ресурсу squoosh, а svg-зображень – засобом svgomg. Це потужні інструменти оптимізації з попереднім переглядом та багатьма налаштуваннями.

Готові HTML файли верстальник передає веброзробнику. Розробка функціоналу вебпродукту може відбуватись за допомогою конструкторів, систем керування контентом CMS та власне самостійна розробка з використанням популярних інструментів, бібліотек та фреймворків.

Кожен із способів має як свої переваги, так і недоліки. Розглянемо коротко ці методи [4].

- **Конструктор** – програмний засіб, зазвичай онлайн, який дає змогу побудувати вебпродукт за модульним принципом, тобто розробник збирає вебпродукт за допомогою готових рішень, які надає конструктор. Такий підхід дозволяє створити сайт навіть користувачам без знань про веброзробку та супутніх навичок.

Переваги:

- простий у використанні, оскільки практично уся рутинна збірка здійснюється у конструкторі пропонованими засобами;
- деякі конструктори надають змогу під'єднати додаткові модулі, за допомогою яких можна створювати вебпродукти різної складності;
- розміщення на хостингу;
- базові модулі у багатьох конструкторах безплатні.

Недоліки:

- приховані витрати, пов'язанні із розширенням функціоналу, із під'єднанням додаткових модулів, розміщенням на хостингу, додавання електронної пошти з ім'ям домену тощо;
- важкість сайту, оскільки конструктор містить у собі величезну кількість додаткового програмного коду, який не відноситься до вебпродукту, але необхідний для побудови його підсумкового вигляду;
- проблеми із перенесенням вебпродукту на інший хостинг, оскільки у багатьох випадках потрібно заново будувати вебпродукт.

Популярними конструкторами створення вебпродуктів на сьогодні є Weblium, Wix, SitePro, Site123, Squarespase та ін.

- **CMS** – це комплекс програмних засобів для керування вебвмістом. CMS пропонує базовий каркас і набір додаткових інструментів та надбудов, що дозволяють не тільки створити вебсайт або вебдодаток, але й підтримувати їхню роботу, оновлювати вміст та взаємодіяти з користувачами. Усі CMS мають панель керування із зрозумілим інтерфейсом. За допомогою CMS можна створити вебпродукт довільної складності, наприклад, як інтернет-магазини, корпоративні вебсайти з ієрархією вкладених сторінок тощо

Переваги:
- багато CMS пропонують безплатні шаблони готових рішень та теми;
- зручне керування контентом за допомогою панелі керування;
- наявність модулів, доповнень, плагінів для різноманітних завдань.

Недоліки:
- вразливість сайту. Найважливіший мінус будь-якої поширеної CMS – це захист від атак;
- вимоги до знань. Розробка сайту на CMS вимагає від клієнта базових знань з верстки та програмування коли клієнт хоче додати до наявного шаблону додатковий функціонал або створити повністю з нуля власний проект;
- складнощі з перенесенням. Хоча популярні CMS на сьогодні мають автоматизовані засоби розміщення на багатьох хостингах, у разі необхідності перенесення сайту або керування його станом, можуть виникнути труднощі, оскільки доведеться виконувати всю процедуру встановлення заново;
- приховані витрати. Додаткові модулі та розширення вебпродукту для CMS можуть бути платними або вимагати платної щомісячної підписки.

Популярними CMS для створення вебпродуктів на сьогодні є WordPress, Joomla, Drupal та ін.

- **Самостійна розробка** – більш трудомісткий процес у порівнянні з використання конструкторів та CMS. Написання вебсайту або вебдодатку вимагає ґрунтовних знань не тільки з мов розмітки, стилізації та програмування, але й архітектури, а також розуміння бізнес-процесів клієнта та багато іншого. Самостійна розробка дозволяє створювати унікальні персоніфіковані вебпродукти будь-якої складності замовлення клієнта.

Переваги:
- вирішення усіх спектрів індивідуальних замовлень, оскільки технології підбираються для написання функціоналу конкретного замовлення;
- індивідуальний дизайн, оскільки при використанні CMS чи конструктора клієнт змушений використовувати готові рішення, які можуть не повністю задовольняти його потреби;
- чистий код дозволяє створити вебпродукт, який замовляє клієнт, без зайвого додаткового програмного коду, без якого не обійтись при використанні конструкторів та CMS;
- чистий код сприяє SEO-просуванню.

Недоліки:
- самостійна розробка вимагає наявність ґрунтовних знань зі створення дизайну, розмітки, стилізації, мов програмування, алгоритмів і структур даних, побудови архітектури вебпродуктів, бізнес-процесів тощо, що спонукає замовника звертатися до фахівців з веброзробки;
- використання конструкторів чи CMS дозволяє зробити певні вебпродукти швидше, ніж при розробці без них;
- тимчасові витрати. Інколи потрібно оплатити витрати на тимчасовий хостинг чи придбання серверу під час розробки;
- вартість розробки. Індивідуальна розробка завжди затратна.

Слід зазначити, думка про те, що вартість самостійної розробки вища за використання конструктора чи CMS, спірна. Розробка великого складного проєкту за допомогою CMS може бути й дорожчою за самостійну чисту розробку. Тому перед створенням вебпродукту слід ретельно продумати вибір інструментів його розробки.

Фронтенд і бекенд – це дві сторони розробки вебпродукту. Фронтенд – це візуальна частина, тобто розробка інтерфейсу (UI) з урахуванням досвіду користувача (UX). Бекенд – все, що працює на сервері, тобто створення програмних засобів, спрямованих на реалізацію логіки вебпродукту. Саме бекенд відповідає за взаємодію користувача з внутрішніми даними, які потім відображає фронтенд.

Бекенд – це не тільки написання коду, але і створення архітектури вебпродукту. Архітектура у розробці визначає структуру та порядок використання баз даних. Важливо, щоб база даних коректно взаємодіяла з кодом програми і безперервно надавала дані на сервер. Також бекенд забезпечує безпеку та налаштовує технології резервного копіювання та відновлення.

Взаємодія фронтенду та бекенду відбувається по колу:

• клієнтська частина програми (фронтенд) відправляє інформацію користувача на сервер (бекенд);

• програма на сервері обробляє інформацію, за потреби взаємодіє з базою даних;

• інформація повертається клієнтській стороні у зрозумілій для користувача формі або у формі, яку можна перетворити у зрозумілу користувачу засобами фронтенту.

Різноманіття сучасних інструментів веброзробки як фронтенду, так і бекенду, дає змогу делегувати частину бізнес-логіки на фронтенд, тим самим злегшити бекенд.

Залежно від завдань вебпродукту та задіяних інструментів можна зробити так, що певні обчислення здійснюються або в клієнті, або на сервері. Кожен з варіантів має свої переваги та недоліки. Так, сервер – більш стабільне захищене середовище, тому природньо що бізнес-логіка реалізується на ньому. Але використання вебпродукту у цьому випадку потребує постійного підключення до мережі.

Використання ж клієнтських програм, які роблять велику частину роботи, часто потребує останніх версій браузерів. Та й не усі браузери однаково працюють з одними і тими ж елементами.

Прогресивні вебдодатки, які налаштовані на сучасні браузери, можуть звертатись до серверної частини все менше. У деяких випадках додаткам потрібен бекенд тільки при першому завантаженні, а потім лише для синхронізації/захисту даних. Отримані дані можна зберігати та обробляти у браузері. Такий рівень означає, що певна частина логіки вебдодатка перебуває безпосередньо на клієнті.

На завершальному етапі веброзробки відбувається тестування усього вебпродукту. Слід зазначити, що тестування відбувається і на проміжних етапах створення компонентів, проте перед розміщенням вебпродукту на хостингу воно є обов'язковим та вирішальним.

Процес тестування стосується усіх частин веброзробки. Написаний код тестують QA-інженери. Вони перевіряють, чи відповідає фактичний результат використання веброзробки очікуваному, чи якісно працює вебпродукт, чи є дефекти, який ступінь серйозності знайдених недоліків.

Тестування програмного забезпечення – це процес випробування програмного продукту з метою перевірки відповідності між реальною та очікуваною поведінкою програми. Для досягнення зазначеної мети існує декілька видів тестування. Основними є тестування вручну та автоматичне [5].

Тестування у ручному режимі здійснює людина, яка перевіряє роботу всіх функцій програми вручну або шляхом взаємодії з програмним забезпеченням та API за допомогою відповідного інструментарію. Це дуже затратний спосіб, оскільки хтось має налаштовувати середовище та проводити тести. Крім того, необхідно враховувати людський фактор, оскільки тестувальник може допустити друкарську помилку або пропустити якийсь етап тестового скрипту.

Автоматичні тести, навпаки, виконуються машиною, яка використовує заздалегідь написаний скрипт тесту. Такі тести можуть значно відрізнятися за складністю – від перевірки одного методу в у певному класі до забезпечення умов, у яких виконання послідовності складних дій у інтерфейсі користувача призводить до однакових результатів. Такий підхід набагато стабільніший і надійніший у порівнянні з тестами, що виконуються вручну, проте якість автоматичного тестування залежить від якості тестових скриптів. Автоматичне тестування є ключовим компонентом безперервної інтеграції та безперервного постачання, а також відмінним способом масштабувати процес контролю якості у міру додавання нових можливостей у додаток. Однак проводити ручне тестування у формі так званого глибокого тестування все одно має зміст.

Знайдені дефекти оформляються у баг-репорти та відправляються розробникам для усунення. Усі етапи фіксуються спеціальними інструментами. Для відстеження багів використовується Jira та інші бак-трекінгові системи, для контролю версій програмного забезпечення, що створюється, застосовують Git, Github, GitLab та інші інструменти.

Отримані під час тестування помилки усуваються розробниками. При внесенні змін до коду або зміни оточення відбувається повторне тестування (його називають регресійним). Процес тестування має повторюватися доти, поки не усунуться усі помилки.

Після успішного тестування вебпродукт розгортають на сервері та здійснюють усі необхідні налаштування та наповнення контентом: заповнення бази даних, файли для скачування, тексти, зображення, мультимедіа тощо.

Слід розрізняти поняття хостингу та серверу .

Хостинг сайтів – це онлайн-послуга, яка дозволяє публікувати вебпродукт в інтернеті. Підписка на послугу хостингу означає оренду простору на сервері, на якому зберігатимуться усі дані та файли, необхідні для правильного функціонування вебпродукту.

Сервер – це фізичний комп'ютер, який працює безперервно, щоб створений вебпродукт був завжди доступний для тих, хто хоче його відвідати. Хостинг відповідає за підтримку роботи сервера, захист його від шкідливих атак та передачу контенту із сервера до браузера відвідувачів вебпродукту.

Більшість провайдерів пропонують декілька типів хостингу, щоб задовольнити різні потреби клієнтів. Найчастіше це [6]:

- загальний хостинг (Shared Hosting)
- VPS (Virtual Private Server – віртуальний приватний сервер) хостинг;
- хмарний хостинг (Cloud Hosting);
- WordPress хостинг;
- хостинг виділених серверів.

SEO – це комплекс заходів, зокрема, по роботі з кодом, індексацією у пошукових системах, побудові структури вебпродукту. Виділяють внутрішню та зовнішню SEO-оптимізацію [7].

Внутрішня SEO-оптимізація пов'язана з деякими змінами самого сайту і починається з визначення семантичного ядра. На цьому етапі визначаються ключові слова, які б сприяли залученню найбільш зацікавлених відвідувачів, та за якими простіше виграти конкуренцію на ринку однакових послуг. Далі ці слова вносяться у розмітку вебпродукту. Тексти, посилання, інші теги адаптуються

так, щоб пошукові системи могли їх успішно знаходити за ключовими словами.

Зовнішня SEO-оптимізація зводиться, як правило, до побудови структури вхідних посилань. Це власне і є розкрутка сайту. До розробки сайту зовнішня SEO-оптимізація не має відношення. SEO-оптимізація класифікується на «білу» та «чорну» (таку, після якої сайт за два тижні потрапляє у топ, а потім у бан по-позовників). Справжня, «біла» SEO-оптимізація, це трудомісткий та тривалий процес, вартість якого може у кілька разів перевищувати витрати на створення сайту.

Завершальним етапом веброзробки є прийняття проєкту замовником. Замовник або його довірена особа передивляється створений вебпродукт, переконується, що він відповідає вимогам, які були закладені на етапі замовлення або доповнені на етапі розробки, та підписує документ про прийняття проєкту. Якщо договором між розробником та замовником передбачена подальша підтримка проєкту чи навчання користування ним, то відбувається навчання відповідного персоналу навикам роботи з вебпродуктом. Супровід проєкту відбувається в обумовленими договором строк та завданнями.

Запитання для повторення

1. Що таке веброзробка?
2. Етапи веброзробки?
3. З чого починається веброзробка?
4. Яка мета проєктування веброзробки продукту?
5. Чому не можна опускати у веброзробці етап проєктування?
6. У чому полягає концепція вебпродукту
7. Що таке концепція дизайну вебпродукту?
8. Що таке UI та UX?
9. Чому важливо для вебдизайнера володіти знаннями з UI та UX? Назвіть їхні переваги та недоліки.
10. Що таке послідовний дизайн?

11. Які редактори для створення макетів вебсторінок ви знаєте?
12. Які завдання виконує верстальник?
13. Типи редактори кодів? Принципи роботи з ними?
14. Які редактори коду ви знаєте? Назвіть їхні переваги та недоліки.
15. Назвіть види інструментів написання функціоналу вебпродукту. Які їхні переваги та недоліки?
16. Що таке фронтенд- та бекенд-розробка?
17. Як відбувається взаємодія між фронтендом та бекендом?
18. Для чого здійснюється процес тестування?
19. Які є види тестування є? Назвіть їхні переваги та недоліки.
20. Як відбувається взаємодія тестувальника з веброзробниками?
21. Що таке хостинг?
22. За що відповідає хостинг?
23. Які є види хостингу?
24. Що таке SEO та його завдання?
25. Що є завершальним етапом веброзробки?

Верстка

Розробка вебпродукту – тривалий і нелегкий процес. Одним із етапів веброзробки є верстка вебсторінок. Поняття верстки пов'язано з такими поняттями як кросбраузерність, семантичність та валідність.

Верстка вебсайту або вебдодатку являє собою опис його візуальної частини за допомогою спеціального коду. Верстальник має перетворити прототип у код, який би візуально відображав макет, створений вебдизайнером. Причому, незалежно від того, який браузер використовує користувач, вебпродукт повинен виглядати і працювати коректно при будь-якому розширенні монітора.

Кросбраузерність – це правильна верстка вебпродукту, за допомогою якої його сторінки однаково відображаються у різних браузерах. Реалізація відбувається за допомогою мови розмітки тексту HTML і каскадних таблиць стилів CSS, а також різноманітних підходів та, в окремих випадках, скриптових мов, зокрема, JavaScript та TypeScript.

Семантична верстка або **семантичний HTML-код** – це підхід створення логічної послідовної ієрархії вебсторінок з використанням HTML-тегів згідно з їхнім призначенням. Для оформлення вебсторінок із семантичнм HTML-код використовують CSS.

Семантична верстка, по-перше, має величезне значення для SEO оптимізації сайту, оскільки грамотно побудований код позитивно впливає на поведінку пошукових роботів; по-друге, сприяє коректній роботі спеціальних програм, зокрема скрінрідерів; по-третє, полегшує завдання у подальшій підтримці роботи і розширенню вебпродукту.

Валідна верстка – це написання HTML-коду, який відповідає стандартам WHATWG і успішно проходить тест на валідаторі. WHATWG – головна міжнародна організація, що розробляє й впроваджує технологічні стандарти для HTML [8].

Валідний код – гарантія того, що верстальником не допущено логічних і синтаксичних помилок при верстанні вебсторінки.

Слід зазначити, що навіть з незначними помилками в коді вебсторінка не пройде через валідатор, але браузер може відобразити усі її елементи. Кращий валідатор – це браузер, оскільки сприйняття вебсторінки браузером – це сприйняття сайту відвідувачем. І все ж слід прагнути писати валідний семантичний HTML-код.

Існує кілька основних підходів до верстки вебсторінок [9]:

- **Фіксована верстка.** При зміні розміру вікна браузера блоки не змінюють свої розміри, а на моніторах з низьким розширенням екрану з'явиться смуга прокрутки;
- **Еластична (гнучка) верстка.** Залежно від розміру вікна браузера блоки змінюють свою ширину;
- **Адаптивна верстка.** Втілюється у життя завдяки різним технологіям під певні розширення екранів. Зміна розміру і розташування блоків відбувається після того, як певний рівень розширення екрану досягнутий;
- **Чутлива верстка**. Являє собою гнучке поєднання адаптивної і еластичної верстки. З технічної точки зору вона є найскладнішою, але в той же час найефективнішою. Філософія чутливого вебдизайну полягає у тому, щоб вебсайт був зручним для перегляду з будь-якого пристрою, незалежно від розміру екрана. Фраза чутливий дизайн була придумана Ethan Marcotte в 2011 році. «Головна особливість чутливого вебдизайну – за рахунок рухомої (fluid) сітки макет автоматично реагує на зміну розмірів екрану, роздуваючись або звужуючись, як повітряна куля» [10].
- **Версія вебсайту для мобільних пристроїв.** Це створення іншого вебсайту зі своїм дизайном, версткою, можливо трохи відмінним функціоналом та URL адресою.

Принципи чутливої верстки дають змогу створити одну розмітку під різні під пристрої. Майстерність верстальника полягає у написанні CSS до неї.

Мал.1 Відображення вмісту вебсторінки на різних пристроях

Чутливий дизайн об'єднує у собі три методики:

- гнучкий макет на основі сітки,
- гнучкі зображення,
- медіазапити.

Гнучкість макета базується на використанні відносних одиниць вимірювання замість фіксованих піксельних значень, що дозволяє регулювати ширину відповідно до доступного простору.

Гнучкість текстового вмісту досягається шляхом обчислення розмірів шрифту щодо розміру шрифту в браузерах (за замовчуванням 16px), наприклад для фіксованого розміру font-size: 32px відносний розмір дорівнює 32px / 16px = 2em. Слід також враховувати здатність обертання екрану мобільних пристроїв та планшетів, а тому використовувати view-портові величини.

Проблема гнучких зображень вирішується за допомогою правила img {width: 50%;}, img{ max-width: 100%;}, img{ min-width: 300px;} для всіх картинок на вебсторінці. Це правило гарантує, що зображення ніколи не будуть ширше, ніж їхні контейнери, і ніколи

не перевищать своїх справжніх розмірів на великих екранах. Слід зауважити, що width: 50% не є усталеним значенням, як і min-width: 300px. Орієнтуватись потрібно на дизайн сторінки та мінімальні розміри екранів пристроїв, для яких передбачена верстка.

Вирізняють такі типи верстки [9]:

- **Таблична верстка.** Суть такої верстки полягає у застосуванні сітки таблиць з невидимою межею, у яких зручно розміщувати різноманітні елементи. Для структурованого розміщення елементів можливе й вкладення таблиць.

Переваги:
- легкість застосування і вимірювання елементів;
- розміри комірок можна задавати відносними величинами. Тому ще нещодавно застосовувалась до еластичної верстки.

Недоліки:
- для того щоб вміст таблиці коректно відображався браузером, спочатку має завантажитись таблиця повністю. Якщо таблиця масивна, завантаження вебсторінки займе досить багато часу;
- занадто громіздкий код через ієрархічну структуру тегів, що підвищує складність внесення змін до окремих комірок;
- погана індексація пошуковими роботами. Контент вебсайту з табличною версткою перебуває відносно далеко один від одного, що ускладнює потрапляння сайту на вищий рівень пошукової видачі.

Хоча таблична верстка вважається застарілою, проте вона підходить для створення документів строгих розмірів, бланків, конвертів тощо.

- **Блокова верстка.** Секції, які створюються за допомогою блочних тегів, є достатньо зручними структурними елементами, оформлення та розміщення яких задається за допомогою каскадних таблиць стилів CSS.

Переваги:

- блокова верстка утворює набагато менший обсяг коду, на відміну від табличної верстки, що не тільки збільшує швидкість завантаження сторінки, а й зменшує навантаження на сервер;
- зручність зміни дизайну шляхом правки файлу стилів;
- переваги в сфері SEO. Замість коду в першу чергу розпізнається контент і семантично правильно розмічається;
- підвищена читабельність коду, що сприяє відповідності стандартам валідності;
- розташування елементів вебсторінки реалізовуються різними технологіями;
- є можливість створити адаптивний дизайн, який буде коректно відображатися як на стаціонарних, так і на мобільних пристроях.

Недоліки:

- підвищена складність освоєння. Сьогодні є багато різних можливостей, що знадобиться чимало часу для їх вивчення;
- кросбраузерність. Вирішення цієї проблеми вимагає значно більших зусиль, ніж у випадку з табличною верстkою.

- **Верстка шарами.** Шари – це такі елементи HTML-коду, які впроваджуються у сторінку сайту накладенням один на одного з піксельної точністю. Зміна параметрів шарів відбуваються за допомогою CSS або навіть JavaScript, що дозволяє використовувати різні ефекти.

Переваги:

- Висока швидкість обробки сторінок браузерами, оскільки зазвичай вміст шару має блочну структуру;
- за допомогою позиціонування різних шарів відносно один одного можна створювати живі і цікаві анімаційні ефекти;

- властивості шарів налаштовуються за допомогою CSS;
- завдяки підтримці системи декартових координат, розташування шарів на сторінці сайту можна вказати з гранично можливою точністю.

Недоліки:
- необхідність наявності досить глибоких пізнань правил стилізації CSS, мов і технологій вебпрограмування JavaScript і/або TypeScript;
- кросбраузерність. Відображення вебсторінки у різних браузерах може відрізнятися;

З технічної точки зору верстка шарами схожа на позиціонування, однак, щоб уникнути проблем з браузерами, шар слід створювати тегом .

Запитання для повторення

1. Що таке верстка вебпродукту?
2. З якими поняттями пов'язана верстка?
3. У чому полягає кросбраузерність верстки?
4. Що таке семантичний HTML-код?
5. Чи рівносильні поняття валідна, семантична та кросбраузерна верстка?
6. Як перевіряється верстка на валідність?
7. Які є основні підходи до верстки? Охарактеризуйте кожен з підходів. Назвіть їхні переваги та недоліки.
8. Які методики об'єднує у собі чутливий дизайн? У чому вони полягають?
9. Які є типи верстоk? Назвіть їхні переваги та недоліки.
10. Де використовується таблична верстка?

Робота верстальника з ресурсом Figma

Свою роботу верстальник починає з прочитання технічної документації до розробки та макетів сторінок, наданих вебдизайнером. Зазвичай прототипи сторінок подаються у певному графічному редакторі. Одним із найпопулярнішим середовищем розробки макетів сторінок є векторний графічний редактор для вебдизайну Figma**.** Запропонована зручна робота з прототипами – моделлю вебпродукту. З нею замовнику простіше оцінити, як люди користуватимуться продуктом. Щоб створити прототип вебпродукту, дизайнер малює екрани та створює зв'язки між ними. У Фігмі можна відразу показати замовнику, як дизайн виглядатиме на екрані смартфона, планшета та інших пристроїв та переходи між різними його частинами.

У Figma можна створювати [11]:

- інтерактивні прототипи сторінок вебпродуктів і мобільних додатків;
- вікна, форми зворотного зв'язку;
- векторні ілюстрації;
- векторні сітки;
- векторні графічні елементи інтерфейсу – іконки, кнопки тощо;
- налаштувати ефекти: зробити клікабельні кнопки, розкрити списки, створити анімацію для блоків та підказок.

Зручно працювати у Figma також верстальникам і веброзробникам, оскільки у ресурсі фактично створено єдине середовище для роботи цілої команди над дизайном. Крім того, є можливість організації спільної роботи над проєктом. У цьому її основна перевага над іншими графічними редакторами. У ній можна відслідкувати історію змін та версій, тому розробник завжди бачить останні зміни макету, менеджер у реальному часі бачить, що відбувається з

проєктом, а замовник може залишати коментарі до дизайну прямо у макеті.

Переваги Figma:

- *Кросплатформність*. Працювати в сервісі можна з браузера, з будь-якого пристрою та будь-якої операційної системи;
- *Командна робота над макетами.* У Figma можна працювати над проєктом одночасно, надавши доступ на перегляд чи редагування. Причому на екрані, у цьому випадку, учасники показуватимуться миготливим курсором з ім'ям. Тому завжди відомо хто одночасно перебуває всередині файлу і чим конкретно займається;
- *Хмарний сервіс*. У Figma всі документи зберігаються у хмарі. Завдяки цьому в редакторі можна колективно працювати над макетами та відкривати їх за посиланням, без завантаження;
- *Дві версії роботи.* Працювати у Figma можна онлайн через або декстопно, завантаживши програму на комп'ютер. У десктопній версії можна працювати в автономному режимі, а коли з'являється доступ до інтернету, зміни синхронізуються.
- *Підтримка історії версій*. У будь-який момент можна переглянути зміни, які внесені іншим членом команди, та за потреби відновити резервну копію;
- *Зворотній зв'язок.* У Figma до макетів учасники можуть залишати коментарі та отримувати відповідь від колег. Історія листування зберігається.
- *Відносна безплатність продукту.* Зареєстрованим користувачам більшість функцій надається безплатно. Доступ до перегляду незареєстрованим користувачам також безплатний.
- *Велика кількість плагінів.* Застосування плагінів дає змогу не лише спростити розробку дизайну вебдизайнеру, але й виокремити елементи і проаналізувати для верстальника чи веброзробника;

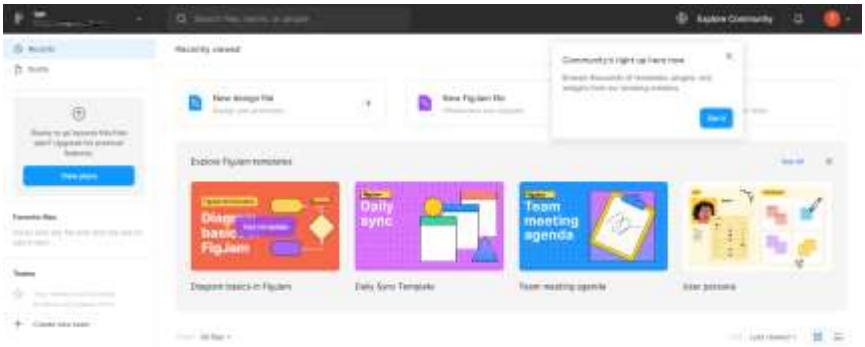
- *Створення власних бібліотек.* Надана можливість створювати бібліотеки із UI компонентами;
- *Імпорт та експорт векторних елементів.* У Figma є основні інструменти для створення та роботи з векторними об'єктами. Також вона дозволяє експортувати зображення у формат SVG, імпортувати векторні об'єкти з Adobe Illustrator або редактора Sketch.

Недоліки:
- деякі функції ресурсу платні;
- не має засобів роботи з поліграфією;
- не можливо безпосередньо імпортувати створені макети із Photoshop. Як варіант – слід спочатку перевести у формат Sketch, а потім імпортувати у Figma.

Після реєстрації на сайті розробника користувач має змогу працювати з ресурсом онлайн чи декстопно. Робота в обох режимах практично однакова з певними відмінностями. Синхронізація даних при роботі декстопно відбувається після під'єднання до інтернету. Верстальнику достатньо бути зареєстрованим і користуватись базовим безплатним функціоналом.

Користувачеві доступні дві робочі області – графічний редактор і менеджер файлів. У Figma файли зберігаються автоматично. До усіх файлів із графічного редактора можна перейти через вкладку Back to files. Останні відкриті файли можна побачити у вкладці Recent менеджера файлів. Ще є розділ Drafts. Сюди потрапляють не лише чернетки, але й файли, яким користувачу надали доступ і він виконав команду Duplicate to Your Draft.

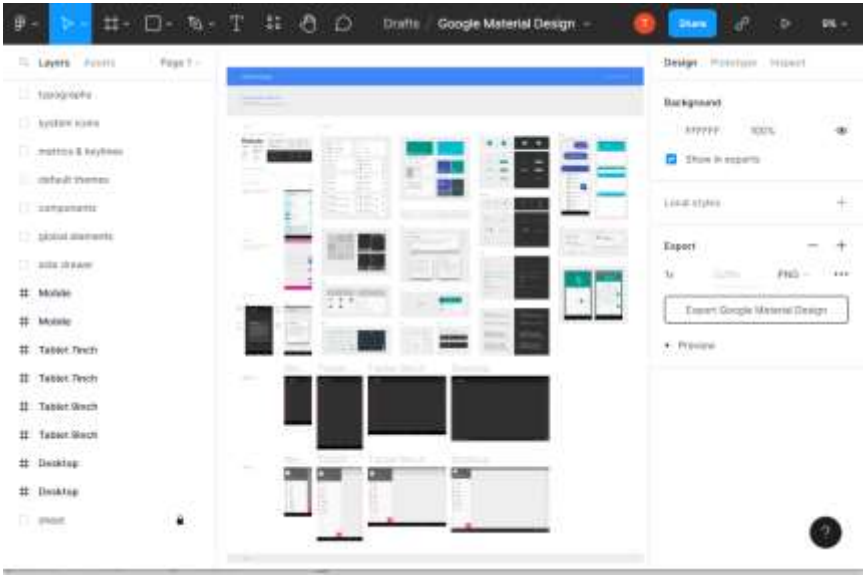
На жаль, на базовому безплатному тарифі у Figma не передбачена функція захисту від копіювання. Тому якщо надати посилання на файл макету, то через команду Duplicate to Your Draft можна файл зберегти як чернетку і далі змінювати його. У платних тарифах Figma захист від несанкціонованих копій уже забезпечується.



Мал. 2 Вікно менеджера проєктів

У вікні менеджера проєктів є зразки макетів. Їх можна використовувати для ознайомлення з інструментами програми.

Вибравши макет, переходять у робочу область – графічний редактор.



Мал. 3 Вікно графічного редактора

Як бачимо, Figma пропонує розробникам зрозумілий інтерфейс:

- зліва панель шарів проєкту,
- зверху панель інструментів,
- праворуч панель властивостей,
- посередині робоча область.

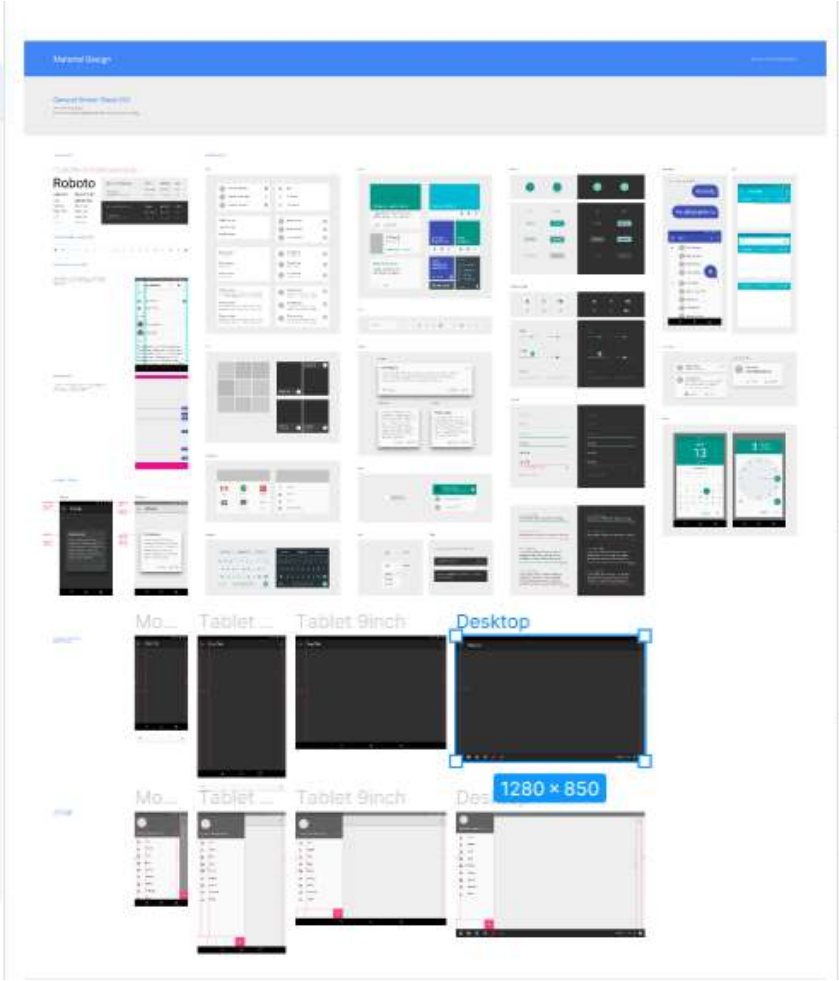
У бічній панелі зліва показана вся структура проекту. Можна простежити вкладеність елементів та знайти потрібний елемент чи групу елементів. Кожен тип елемента позначений окремою іконкою – текст, зображення, складовий блок.

Мал. 4 Панель шарів проєкту

У верхній частині сторінки розташоване головне меню – панель інструментів. Більшість інструментів у ньому призначені для дизайну, але верстальнику зазвичай знадобиться меню масштабування, розташоване праворуч.

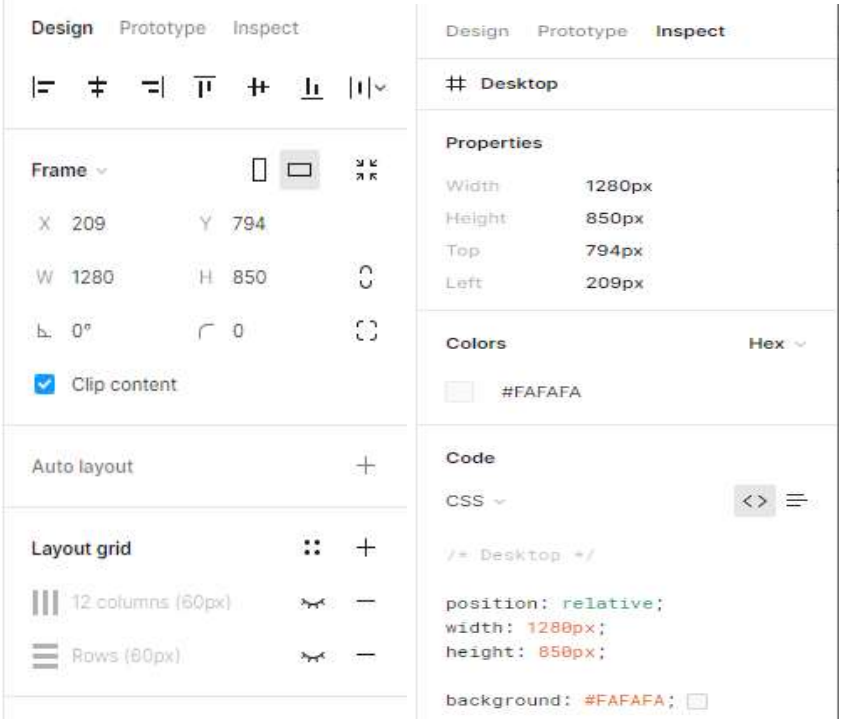
Мал. 5 Панель інструментів

Основна робоча область у центрі – для безпосередньої взаємодії з макетом.



Мал. 6 Робоча область графічного редактора

Бічна панель справа містить три вкладки Design, Prototype і Inspect, але верстальнику будуть потрібні тільки дві з них – Design і Inspect. У цих вкладках є вся доступна інформація про елементи макету.

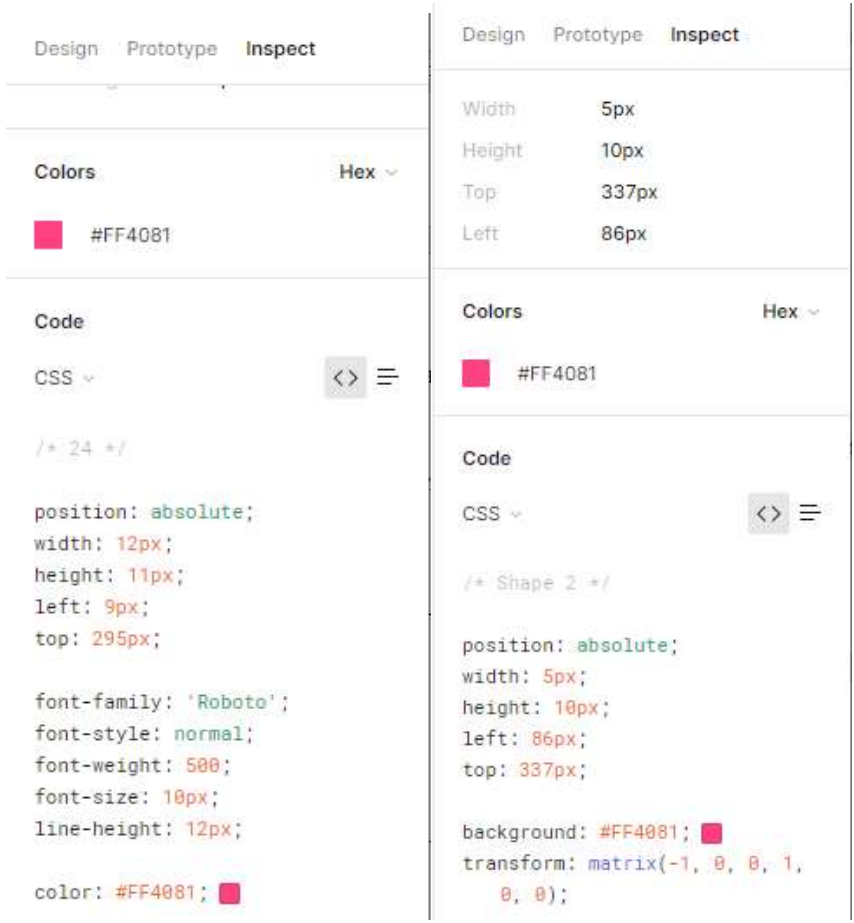


Мал. 7 Панель властивостей

У Figma безліч корисних опцій, які інтуїтивно зрозумілі. Також, під час роботи у редакторі, з'являються повідомлення з підказками, які допомагають зорієнтуватися у сервісі. При наведенні курсору на певний блок, іконку або текст показуються розміри вибраного елемента, відстані до найближчих блоків та інше.

Основне завдання верстальника під час роботи з макетом – отримати параметри елементів. Він працює в основному із секцією

Code на вкладці Inspect. Тут відображаються відповідні правила CSS.



Мал. 8 Секція Code

Розглянемо основні елементи.

- *Текст*. Щоб дізнатися усі текстові параметри, потрібно виділити елемент та відкрити вкладку Inspect у правій бічній панелі. Там у списку властивостей відобразяться усі параметри, що використовуються – назва, розмір, насиченість і колір шрифту, висота рядка та інші.

- *Блоки*. Властивості width та heigh у секції Code розкажуть про розмір блоку та зображення.

- *Розміри елементів та відстань між ними*. Розмір можна визначити кількома способами. Наприклад, у секції Code переглянути значення властивостей width та height. Або просто виділити потрібний блок – розмір відобразиться унизу елемента. Також можна дізнатися відстань між будь-яким елементом та сусідніми з ним. Для цього потрібно виділити його і затиснути Alt, а потім наводити курсор на інші елементи - з'являтиметься напрямна та значення.

Зауваження. У секції Code зазначені ще й розміщення вибраного елемента на макеті. Це абсолютні величини. Верстальник їх зазвичай прямо не використовує, якщо здійснює не фіксовану верстку, а чутливу. Проте ці величини допомагають йому бачення усієї картини дизайну і певних метричних розрахунків.

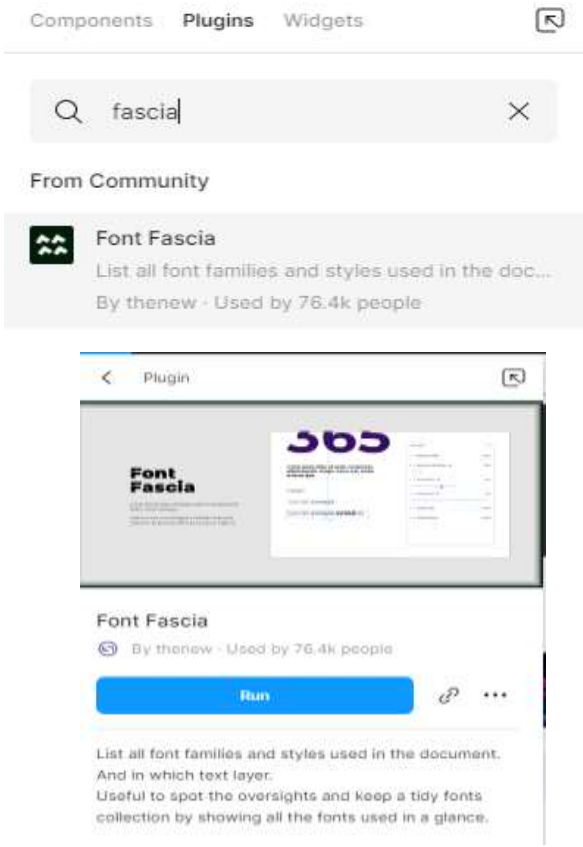
- *Робота з зображеннями*. Для цієї роботи є секція Export, розташована у вкладці. Натиснувши на плюс – секція відкриється повністю. У ній слід вибрати потрібні параметри – формат для експорту зображення та необхідність збільшення для мобільних пристроїв. Натиснувши на Export image, програма запропонує зберегти картинку.

Зауваження. Якщо потрібний прозорий бекграунд зображення – вивантажувати слід у форматі png. Інші зображення для економії місця краще вивантажувати у форматі jpg. Не треба вивантажувати зображення у svg-форматі, оскільки це просто кодування і керувати ним потім через стилі неможливо. Також зображення при експорті можуть підтягувати зайвий бекграунд. Тому усі експортовані зображення слід оптимізувати перед використанням їх у веброзробці.

- *Збереження векторного об'єкта як SVG код.* Щоб зберегти векторний об'єкт із Figma як SVG код потрібно натиснути правою кнопкою миші на елементі, вибрати «Копіювати як SVG» і вставити у код у HTML або інший веб-проект.

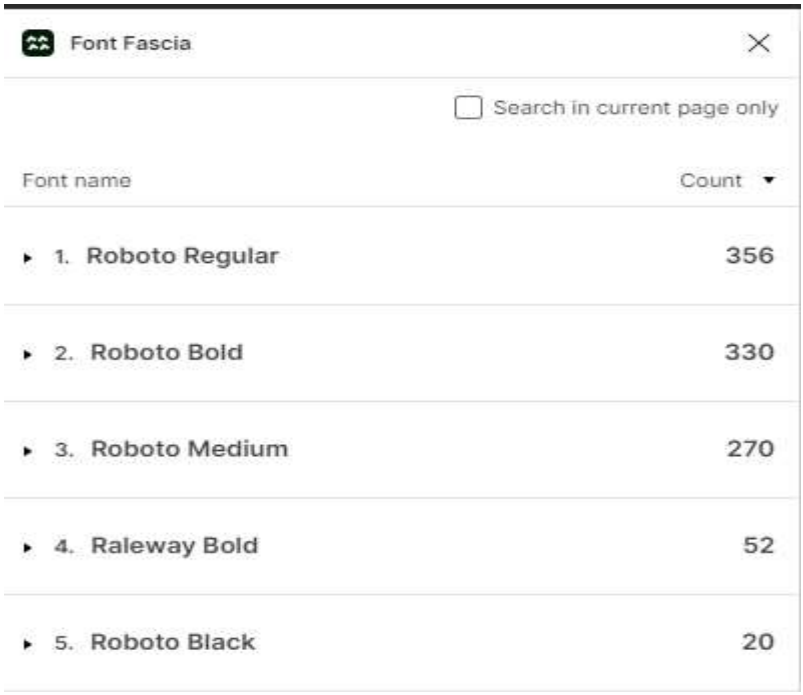
Плагіни розширюють можливості Figma, допомагають прискорити роботу й автоматизувати рутинні завдання. Можна продивлятися плагіни у меню, як у магазині додатків: на картці плагіна описані функції, кількість установок і лайків. У пункті Plugins зібрано розширення, які допоможуть прискорити роботу у Figma. Плагіни відсортовані за рекомендованими, популярними та встановленими.

Наприклад, із плагіном Font Fascia можна отримати статистику про усі шрифти, які використані у макетах проєкту. Це дає змогу верстальнику не лише побачити необхідні шрифти до верстки, але й визначити їх пріоритет.



Мал. 9 Запуск плагіна Font Fascia

Запустивши на макеті цей плагін, отримуємо



Мал. 10 Результат виконання плагіна Font Fascia

Як бачимо, перелічено усі сімейства шрифтів та їх кількість.

Ще одним корисним плагіном є Anima – Export Figma to HTML, React, Vue code, який дає можливість отримати код, що базується дизайні. Для роботи з плагіном потрібно створити обліковий запис Anima і синхронізувати проект.

Проте слід пам’ятати, що експортовані коди зазвичай місять зайву розмітку. Справа уже розробника усунути зайве.

Макети, створені вебдизайнером повинні відповідати вимозі системності.

Системність – кратність розмірів елементів і відступів, ідентична поведінка подібних компонентів і продумана кількість шрифтів.

Виділимо лише деякі аспекти, які допоможуть дотримання вимоги системності.

- *Різні відступи ідентичних елементів***.** Інколи у макетах зустрічаються різні відступи між ідентичними елементами інтерфейсу. Це очевидна помилка макета і переносити їх до верстки не слід. Тим більше, що це ускладнює верстку. Потрібно зробити відступи на сайті більш логічними – уніфікувати їх за певним вибраним значенням, що не порушує концепцію макету. На практиці рекомендується використовувати відступи, кратні певному числу. Найчастіше це число 4. Тобто потрібно буде відступи з макета перетворювати на найближче значення, кратне 4-му: 5 => 4; 7 => 8; 10 => 12 або 10 => 8.

Слід зазначити, що це загальноприйняті рекомендації. Так, для дуже дрібних відступів можна працювати з числом 2, а для великих, навпаки – 8. А деякі верстальники, наприклад, працюють з числами, кратними 5.

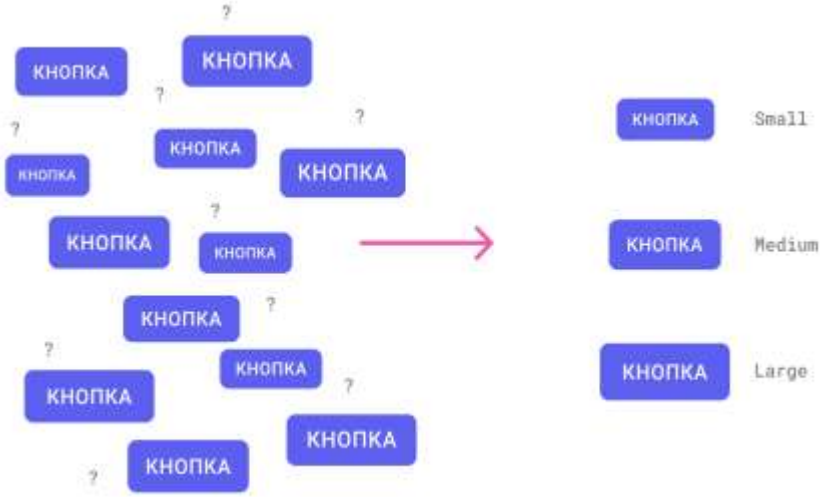
Це вдасться зробити, якщо не потрібно дотримуватись pixel perfect верстки.

Pixel perfect – це спосіб верстки строго за макетом. При цьому, дотримуються розмірів та відступів із макету з точністю до пікселя. Слід зазначити, що дотримання pixel perfect часто призводить до перевантаження коду, якщо дизайнер якомусь елементу із групи подібних елементів задав розміри чи шрифти, які відрізняються від інших. Це також може бути причиною, що завантаження весторінки відбувається повільніше. Перевірити, чи відповідає pixel perfect верстка макету можна за допомогою плагіна PerfectPixel для браузерів Chrome, Opera, Edge, та плагіна Pixel Perfect Pro для браузера Firefox.

Мал. 11 Систематизація відступів

- *Занадто багато розмірів схожих елементів.* У деяких макетах зустрічаються майже однакові елементи, але кожен чимось відрізняється від інших. Наприклад, може бути низка кнопок, при цьому, деякі з яких відрізняються помітно, а деякі на кілька пікселів. У цьому випадку рекомендується проаналізувати призначення кнопок та згрупувати їх за цією ознакою. І визначити розмір кнопки для групи. Зазвичай у макетах можна виділити до 3-4 розмірів елемента.

Це ж стосується й інших елементів, наприклад, зображень у секції з елементами карткового дизайну, груп іконок у верхньому (header) та нижньому колонтитулі (footer) сторінки тощо.



Мал. 12 Систематизація розмірів подібних елементів

При уніфікуванні розмірів елементів для тач-інтерфейсів рекомендується не робити інтерактивних елементів менше ніж 40×40 пікселів. Для десктопу можна поменше, але не занадто. Якщо це не посилання усередині тексту, то добре зберегти мінімальний розмір хоча б 32×32 пікселя.

- *Недостатня контрастність*. Дотримування контрастності кольорів є дуже важливою вимогою. Контрастність оцінюють за стандартом WCAG [12]. Він допоможе оцінити, наскільки легко зчитуватиметься текст для людей з особливостями зору, а також тих, хто використовує сайт при яскравому сонячному світлі.

Мал. 13 Дотримання контрастності кольорів

- *Занадто багато схожих кольорів однотипних елементів.* Як і у випадку з різноманітністю розмірів схожих елементів треба згрупувати їх за призначенням та уніфікувати.
- *Занадто багато розмірів шрифтів.* Велика кількість використаних різних шрифтів у макеті не тільки додає додаткової роботи верстальнику, але засмічує макет. Деякі шрифти дуже схожі між собою і звичайному користувачу важко їх розрізнити. Тому, по-перше, треба продумати групу шрифтів, а по-друге, їх розміри. Виділяють групи заголовків кожного рівня, групи текстів, та особливі унікальні елементи.

Для кожної групи підбирають свій розмір відповідно до шрифту. При цьому слід враховувати аудиторію користувачів вебпродукту. Рекомендується не зменшувати розмір шрифтів, а заокруглювати у більший бік, щоб читання тексту точно не погіршилося.
Бажано, щоб цей аспект вирішував усе ж вебдизайнер. Тут головне не порушити концепцію дизайну і слід діяти лише якщо розмірів дійсно багато або окремий текст вибивається із загального стилю.

- *Текст незручний для читання.* Щоб текст добре читався, рекомендується щоб розмір шрифту був не меншим за 16 пікселів, а міжрядковий інтервал – 1.5 для текстів і 1.3 для заголовків. Для цього у секції Code вкладки Inspect панелі властивостей висоту рядка виставити у відносних одиницях, наприклад, line-height: 1.5. За потреби можна збільшити розмір шрифту, не

нашкодивши дизайну. Тоді висота рядка буде змінюватися за розміром шрифту.

Взагалі кажучи, верстальнику достатньо уміти читати макет, а не розробляти його. Проте бувають такі ситуації, коли у макеті опущені деякі деталі. У великих компаніях звичайно макет передається на доопрацювання вебдизайнеру, а ось верстальнику-фрілансеру доводиться інколи самому вносити правки у макет. У цьому випадку, основні навики роботи з figma йому потрібно знати – це створення елементів дизайну, та стилізація їх. У цьому випадку він уже працює з вкладкою дизайн правої панелі властивостей.

Найчастіше дизайнер може опускати елементи стану у дизайні. Кожен інтерактивний елемент сайту – кнопки, посилання, елемент форми – дає зворотний зв'язок при взаємодії. Коли користувач наводить курсор на кнопку, вона реагує на це: змінює колір або відображає тінь, щоб користувач зрозумів, що з кнопкою можна взаємодіяти.

- *Немає базових станів* Базові стани – це очевидні взаємодії: наведення і натискання. У макеті вони можуть бути відсутніми повністю або тільки в окремих елементів.

Якщо стилів немає у кількох елементів, то можна повторити логіку, яка вже є у макеті. Зміна кольорів, набір властивостей, плавність – це можна запозичити у відмальованих станах.

Якщо немає орієнтирів, то доведеться вигадати стани самостійно. При цьому досить використовувати прості прийоми, наприклад зміну кольору або додавання прозорості.

Мал. 14 Додавання базових станів

- *Немає більш специфічних станів*. Найчастіше у макетах втрачені не базові стани, а більш специфічні, наприклад, фокусування чи неактивність. Ці стани відносяться до особливостей роботи вебінтерфейсу.

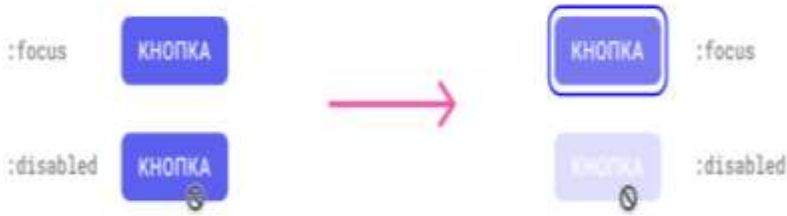
Фокус відрізняється від інших станів. Це аналог курсора, а не підсвічування елемента. Тому він повинен бути пізнаваний на всіх елементах і виглядати однаково на кнопці, посиланні, елементах форми та інших.

Якщо немає фокусу для конкретного елемента, але є для інших, то досить повторити його.

Якщо у макеті фокуси не продумані, слід скористатися стандартним фокусом браузера. Там, де він не продуманий, намалювати його наближеним до браузерного.

Неактивні елементи зазвичай блідіше і не реагують на наведення та натискання. Тому достатньо до відповідного кольору додатку відсоток прозорості. Рекомендується уніфікувати відсоток прозорості для усіх неактивних елементів.

Є й більш рідкісні варіанти, наприклад, поле в режимі читання або відвідане посилання. Таких специфічних станів багато. Слід намагатися зрозуміти, чим відрізняється логіка елемента і як це візуально підкреслити: кольором, яскравістю або навіть додатковою рамкою.



Мал. 15 Додавання специфічних станів

- *Недостатньо станів валідації елементів форми.* Якщо для валідації розписано мало станів, то можна обмежитися доповненням станів для неправильного заповнення. Для інших станів можна максимально залишати браузерну валідацію: браузери перевіряють форми і навіть вміють писати, яким параметрам не відповідає введене значення. Обійтися можна стилізацією помилок.

Обов'язково слід додати текст пояснення: користувач повинен знати, що саме спрацювало не так.

- *Відсутність технічних сторінок.* Буває й таке, що у макеті відсутні деякі технічні сторінки. Зазвичай, це сторінки: 404, відновлення пароля, порожніх відповідей на пошукові запити або застосованих фільтрів, коли нічого не знайшлося.

Для сторінок на кшталт 404 досить взяти базову сторінку, призначену для простого текстового контенту, як блок про компанію, новину чи статтю і написати туди потрібний текст.

З пошуком та фільтром простіше: дизайн оточення у же є. Достатньо подумати про те, які дії будуть корисні користувачеві у такій ситуації, і додати їх у вигляді посилань або кнопок поруч із текстом.

Вище наведені лише окремі ситуації, які виникають при опрацюванні макета верстальником. Постійне навчання, досвід і практика завжди будуть гарним порадником йому.

Запитання для повторення

1. Для чого призначений редактор Figma?
2. Хто може використовувати редактор у своїй діяльності?
3. Назвіть їхні переваги та недоліки.
4. Які області роботи надає редактор?
5. Для чого призначений менеджер проєктів?
6. Опишіть інтерфейс графічного редактора.
7. Які вкладки містить панель властивостей. Які їх призначення?
8. Де відображені правила CSS, елементів макету?
9. Чи можна побачити усі правила макету сторінки зараз?
10. До є можливість експортувати зображення з макету? У яких форматах?
11. У яких випадах рекомендується використовувати кожен з пропонованих форматів?
12. Які дії треба виконати, щоб зберегти векторний об'єкт як SVG код для вставки у HTML або в інший проєкт?
13. Які плагіни дають змогу вибрати усі шрифти, застосовані у макетах проєкту?
14. У чому полягає вимога системності макету?
15. Назвіть найпоширеніші недоліки, які порушують вимогу системності макету.
16. Які є рекомендації роботи з відступами із їх систематизацією?
17. Наведіть приклади рекомендацій для систематизації у роботі з текстом.
18. У чому полягає суть принципу pixel perfect верстки?
19. За якими стандартом оцінюють контрастність кольорів?
20. Чи потрібно верстальнику вміти працювати з базовими інструментами у векторному редакторі? Відповідь обґрунтуйте.

Інструменти розробника

Для фронтенд-розробника існують інструменти у всіх сучасних браузерах, які допомагають йому у розробці. Крім того, деякі текстові редактору коду пропонують свої вбудовані інструменти для зручності, щоб постійно не перемикатися між редактором та браузером. З їхньою допомогою можна дізнатися, як побудувалося DOM-дерево, які теги та атрибути є на сторінці, чому не підвантажилися шрифти, які не спрацювали правила CSS та багато іншого. Розглянемо на прикладі браузера Google Chrome.

Нижче наведена інформація взята з ресурсу розробника [13].

Chrome DevTools – це набір інструментів веброзробника, вбудованих безпосередньо у браузер Google Chrome . DevTools може допомогти редагувати сторінки на льоту та швидко діагностувати проблеми, що зрештою допоможе швидше створювати кращі вебпродукти.

Існує багато способів відкрити DevTools, оскільки різні користувачі хочуть отримати швидкий доступ до різних частин інтерфейсу DevTools:

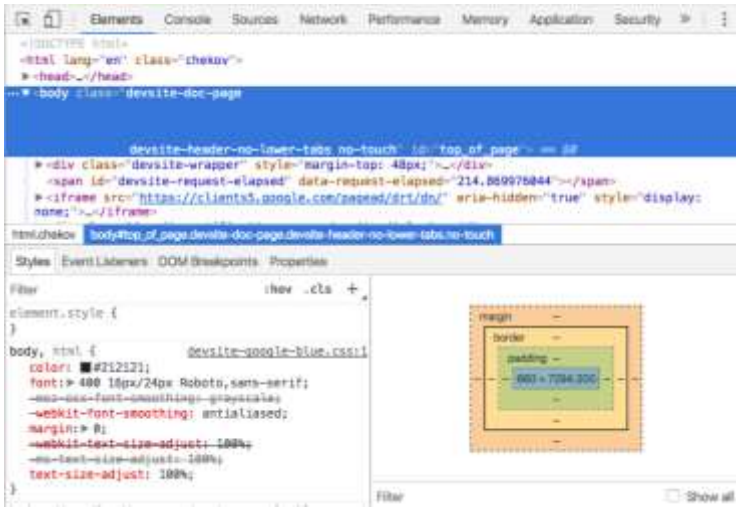
- Якщо потрібно відкрити інструменти розробника, а потім переходити по вкладках, то слід натиснути правою кнопкою миші на сторінці та вибрати команду Перевірити (Inspect). Також можна натиснути Ctrl+ Cmd+I (Mac), F12 (Windows), Ctrl+ Shift+I (Linux);
- Якщо потрібно працювати з DOM або CSS, то слід спочатку відкрити інструменти розробника та перейти на панель Елементи. Також можна натиснути Cmd+Option+C (Mac) або Ctrl+Shift+C (Windows, Linux, ChromeOS).
- Якщо потрібно переглянути зареєстровані повідомлення або запустити JavaScript, то слід спочатку відкрити інструменти

розробника та перейти на панель Консоль. Також можна натиснути Cmd+Option+J (Mac) або Ctrl+Shift+J (Windows, Linux, ChromeOS.

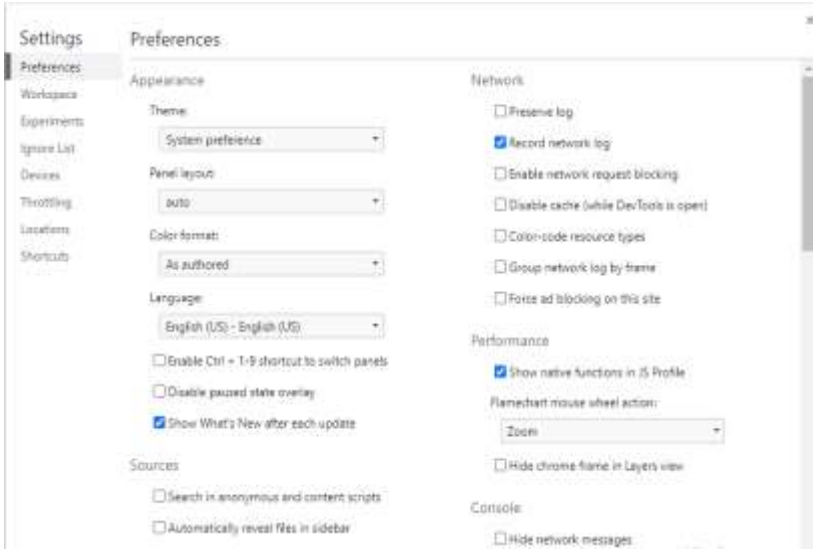
Мал. 16 Відкриття інструментів розробника

У відкритих DevTools можна налаштувати інструменти розробника та змінити їх зовнішній вигляд, наприклад, вибрати темне оформлення інтерфейсу або вибрати мову перегляду тощо. У налаштуваннях є список гарячих клавіш, знання цих комбінацій значно прискорить роботу.

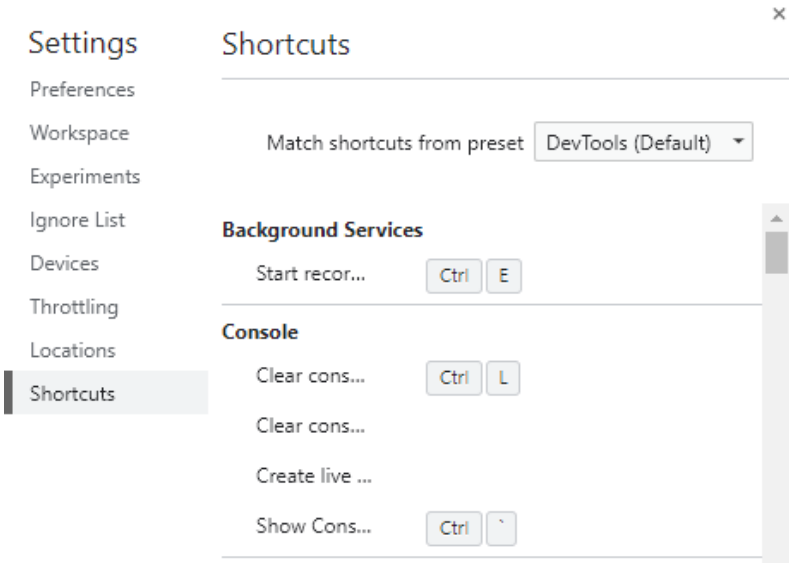
Натиснувши на значок, зазначений на мал. 20, можна змінити положення інструментів розробника у вікні браузера, зручному для користувача, здійснити пошук, перейти до потрібної панелі, передивитись більше можливостей інструментів тощо.



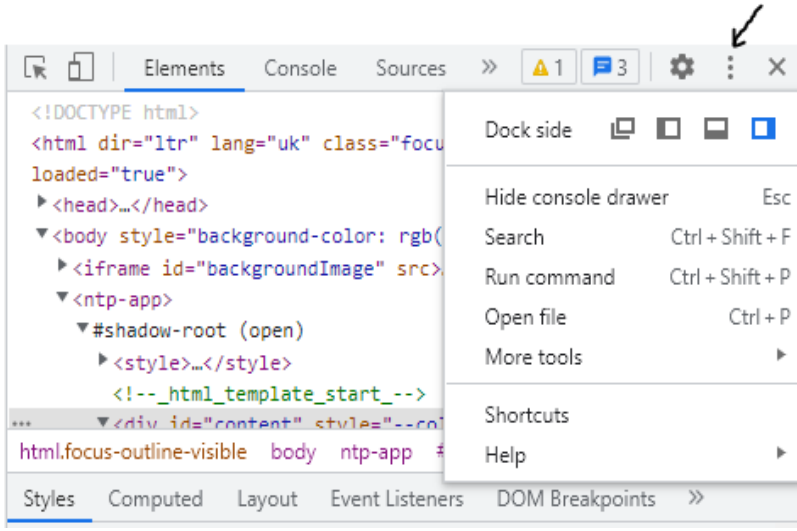
Мал. 17 Вікно інструментів розробника



Мал. 18 Налаштування інструментів розробника



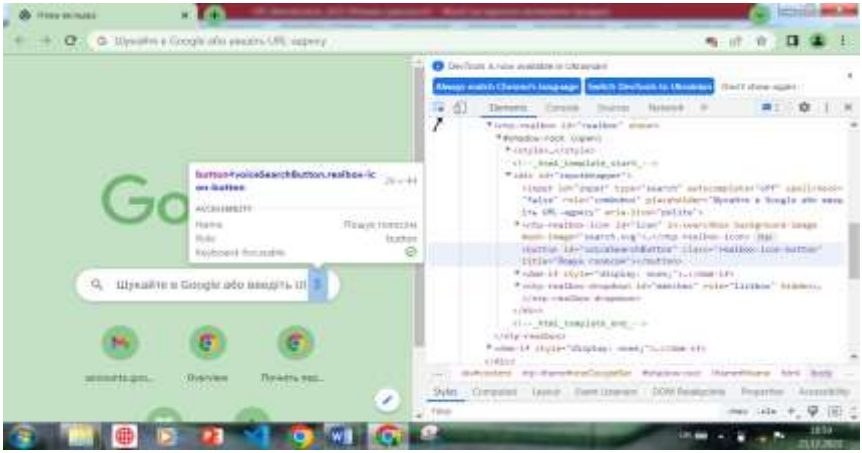
Мал. 19 Гарячі клавіші інструментів розробника



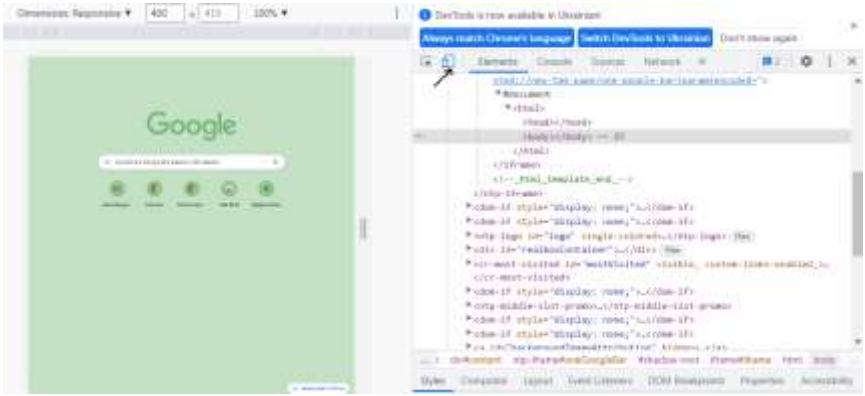
Мал. 20 Вибір розташування вікна інструментів розробника

DevTools може розміщуватися знизу, ліворуч, праворуч, а ще можна відкріпити налагоджувач і працювати з ним в окремому вікні.

У вікні налагоджувача крім вкладок є ще дві команди, які дають змогу візуально вибрати вузол DOM та передивитись вигляд вебсторінки на різних пристроях . Причому розміри можна задати вручну або вибрати пристрої із випадного меню Responsive.

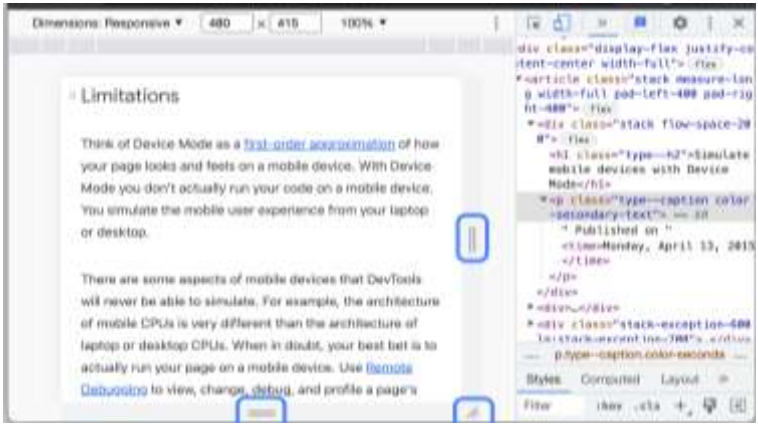


Мал. 21 Інспектор вибраного вузла DOM



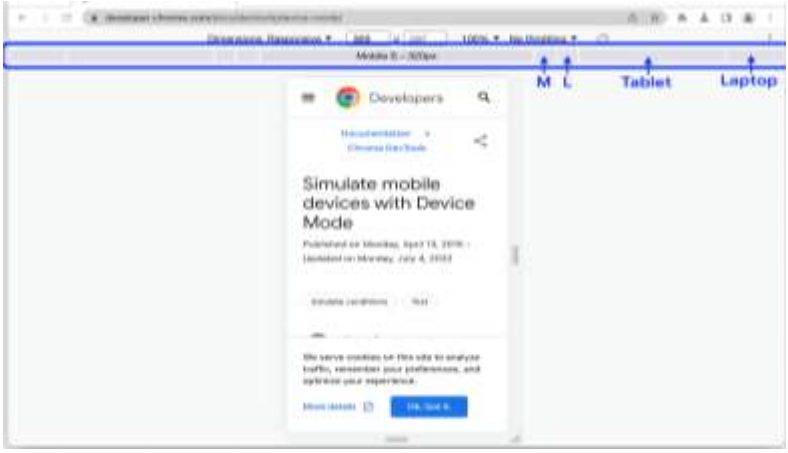
Мал. 22 Вигляд сторінки на пристрої зазначених розмірів

За замовчуванням панель інструментів пристрою відкривається у вікні перегляду з параметрами Dimensions , встановленими на Responsive. Змінювати розміри можна або вручну, задавши розміри, бо перетягуванням ручок



Мал. 23 Підбір розмірів сторінки

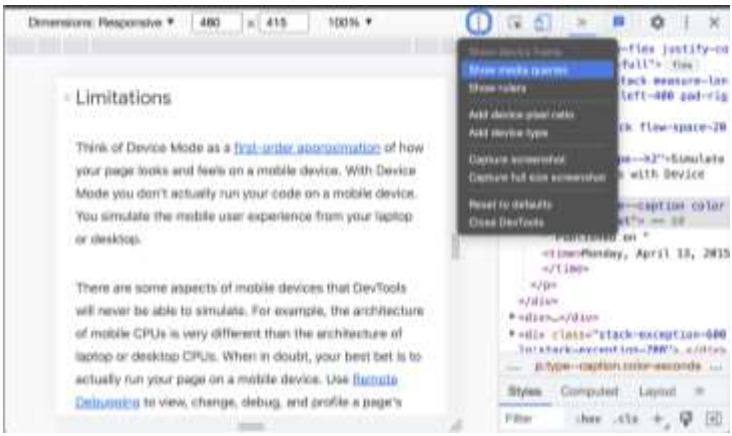
Крім того, можна скористатися панеллю попередніх налаштувань ширини, щоб установити ширину одним із наведених нижче клацанням миші:



Мал. 24 Підбір розмірів сторінки

Мобільний С	Мобільний М	Мобільний Л	Планшет	Ноутбук	Ноутбук Л	4K
320 px	375 px	425 px	768 px	1024 px	1440 px	2560 px

Щоб відобразити контрольні точки медіа-запитів над вікном перегляду, слід натиснути Додаткові параметри > Показати медіа-запити .

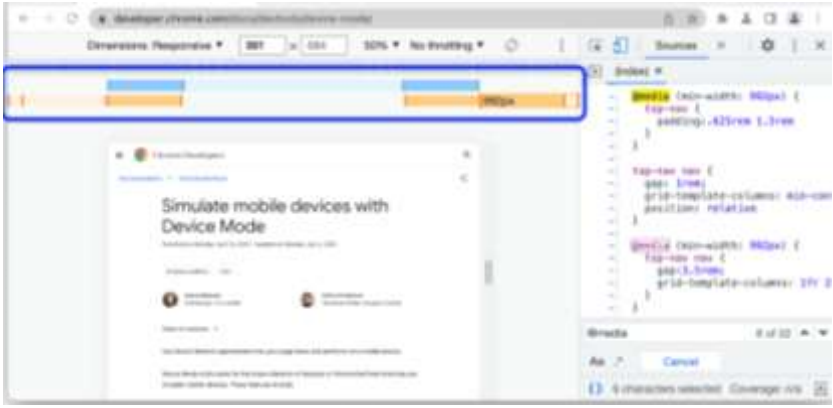


Мал. 25 Робота з медіа-запитами

DevTools тепер відображає дві додаткові панелі над вікном перегляду:

- Синя смуга з max-width точками розриву.
- Помаранчева смуга з min-width точками зупинки.

Натискуйте між точками зупинки, щоб змінити ширину вікна перегляду, щоб точка зупинки запускалася.



Мал. 26 Робота з медіа-запитами

Щоб знайти відповідну @media декларацію, клацніть правою кнопкою миші між точками зупину та виберіть Показати у вихідному коді. DevTools відкриває панель Sources у відповідному рядку редактора .

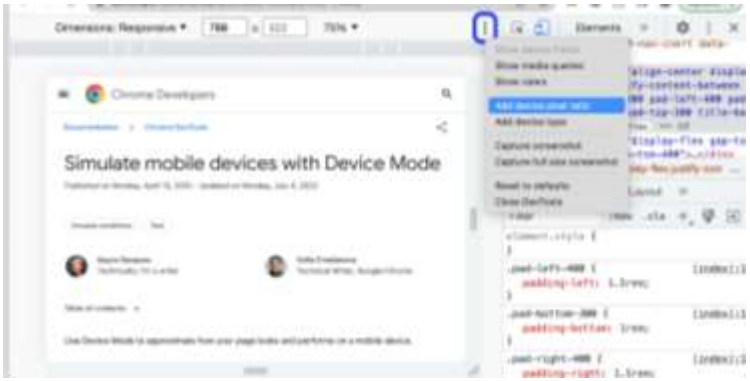
Мал. 27 Робота з медіа-запитами

Коефіцієнт пікселів пристрою (DPR) – це співвідношення між фізичними пікселями на апаратному екрані та логічними (CSS) пік-

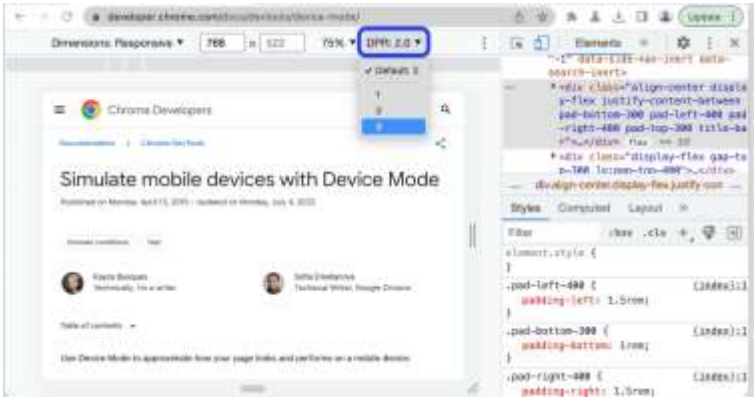
селями. Іншими словами, DPR повідомляє Chrome, скільки екранних пікселів використовувати для малювання пікселя CSS. Chrome використовує значення DPR під час малювання на дисплеях HiDPI (висока кількість точок на дюйм).

Щоб встановити значення DPR потрібно:

1) натиснути Додаткові параметри > Додати співвідношення пікселів пристрою;
2) на панелі дій у верхній частині вікна перегляду вибрати значення DPR із нового спадного меню DPR

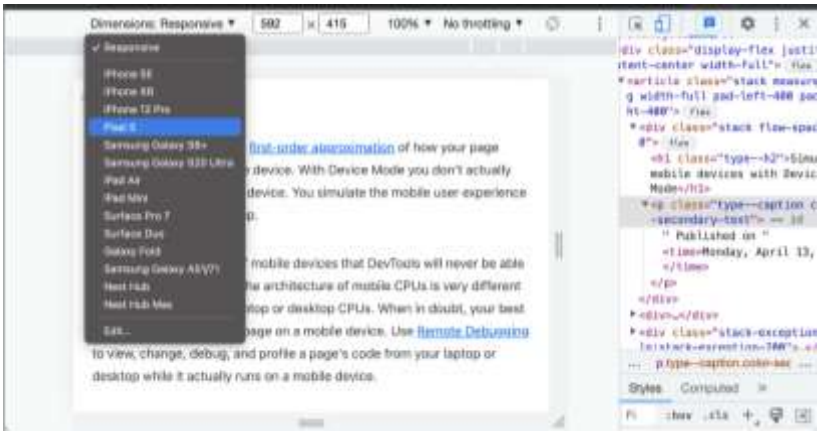


Мал. 28 Вибір DPR



Мал. 29 Вибір DPR

Також можна використовувати список Тип пристрою, щоб імітувати мобільний пристрій або настільний пристрій. Щоб імітувати розміри певного мобільного пристрою, виберіть пристрій зі списку Розміри .



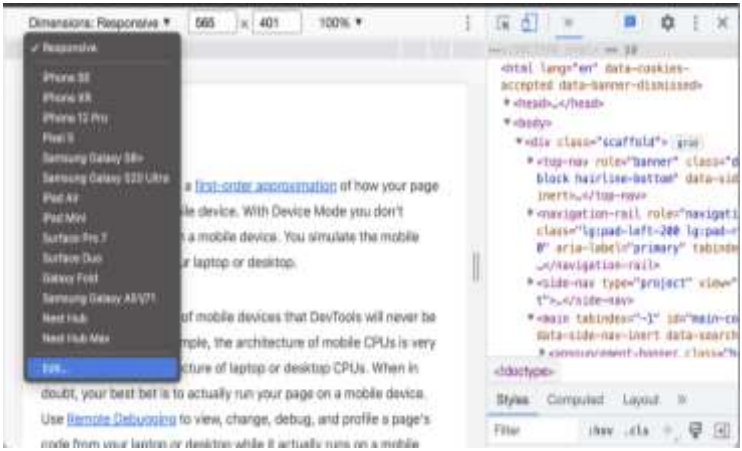
Мал. 29 Вибір мобільного пристрою

Щоб повернути вікно перегляду в альбомну орієнтацію слід натисніть піктограму Повернути. Таким чином ми імітуємо різні пристрої та їх орієнтації відображення.

Щоб додати спеціальний пристрій потрібно:

1) у списку пристроїв вибрати Редагувати;
2) на вкладці Налаштування > Пристрої вибрати пристрій зі списку підтримуваних або виконати команду Додати спеціальний пристрій, щоб додати свій власний;
3) якщо ви додаєте власний, введіть назву, ширину та висоту пристрою, а потім натисніть Додати. Поля співвідношення пікселів пристрою, рядок агента користувача та тип пристрою необов'язкові. Поле типу пристрою – це список, для якого за замовчуванням встановлено значення Mobile;

4) повернувшись у вікно перегляду, виберіть щойно доданий пристрій зі списку Розміри.



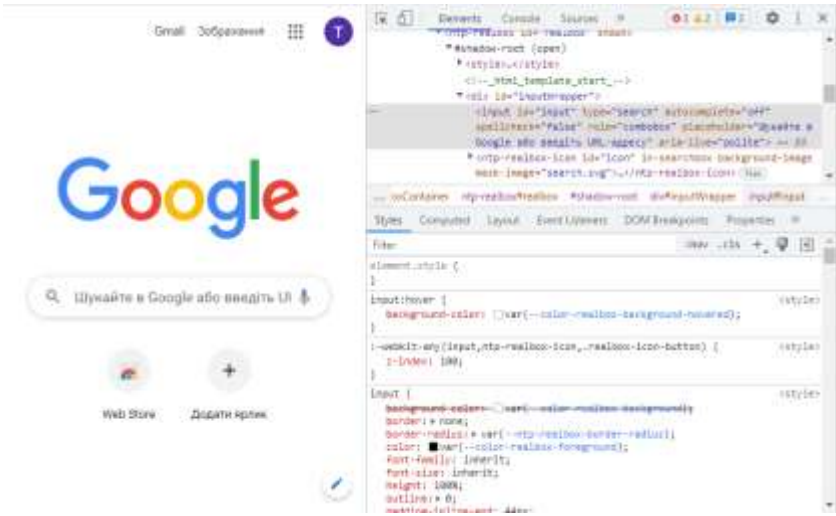
Мал. 29 Редагування даних мобільного пристрою

Мал. 30 Додавання мобільного пристрою

З допомогою панелі Елементи можна відстежувати вузли DOM та їхні властивості на сторінці, можна редагувати стилі та перевіряти верстку на переповнення.

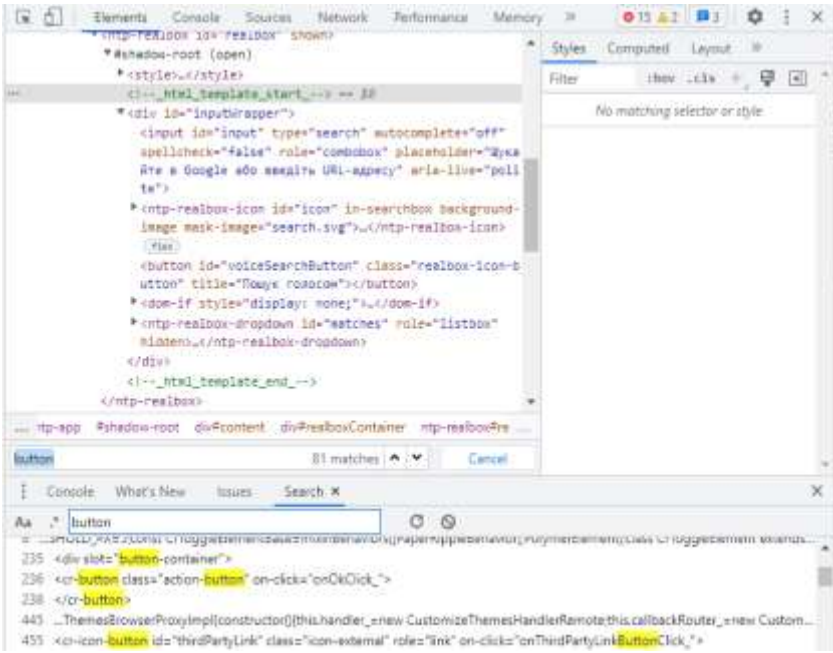
Є три способи отримати інформацію про будь-який вузол DOM на сторінці.

- *Через інспектор елемента*. Спосіб зручний, якщо вибрати вузол DOM і натиснути по ньому правою кнопкою миші команду Перевірити. DevTools відкриває панель Елементи та вибирає елемент у дереві DOM. На панелі Стилі внизу можна побачити правила CSS, застосовані до вибраного елемента.



Мал. 31 Інспектор елемента сторінки

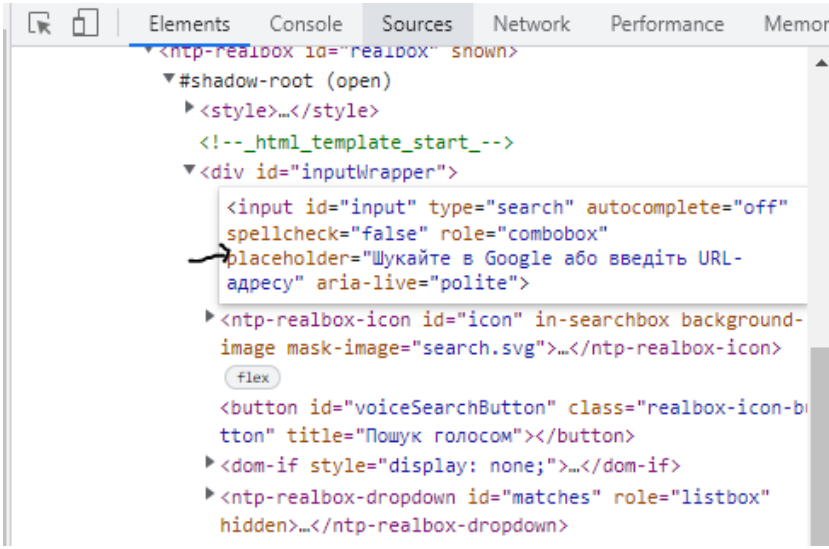
- *Пошук за елементами*. При відкритому налагоджувачі натиснути Ctrl+Shift+F, і внизу з'явиться вікно пошуку за тегом, атрибутом, класом або текстовим вмістом вузла DOM. Ввести необхідну назву, наприклад button і виділяться усі відповідні вузли.



Мал. 32 Пошук за елементом

- *Візуальний пошук*. При відкритому налагоджувачі відкривають режим візуального пошуку, знаходять потрібний елемент та натискують по ньому (див. мал. 21). Якщо потрібно відредагувати атрибут, клас або текст вузла, потрібно двічі натиснути в потрібній точці та внести зміни. Щоб переміститися вперед, натискують Tab, назад – Shift+Tab, а щоб приховати елемент натискують H.

У вікні браузера можна змінити вузли розмітки і подивитись як це буде виглядати. Щоб редагувати вміст вузла як HTML із підсвічуванням синтаксису та автозаповненням, потрібно двічі натиснути на вміст у дереві DOM і можна вносити зміни. Для цього правою кнопкою миші натискують на елемент і вибирають команду Edit as HTML або натискають на клавішу F2. Результат одразу відображається на екрані.



Мал. 33 Вікно редагування елемента розмітки

Щоб приховати вибраний вузол слід натиснути клавішу H . Якщо натиснути клавішу H ще раз, то вузол буде показано знову.

Щоб видалити вибраний вузол слід натиснути клавішу Delete . Якщо натиснути комбінацію клавіш Ctrl+ Z або Cmd+ Z (Mac), то остання дія скасовується і вузол з'являється знову.

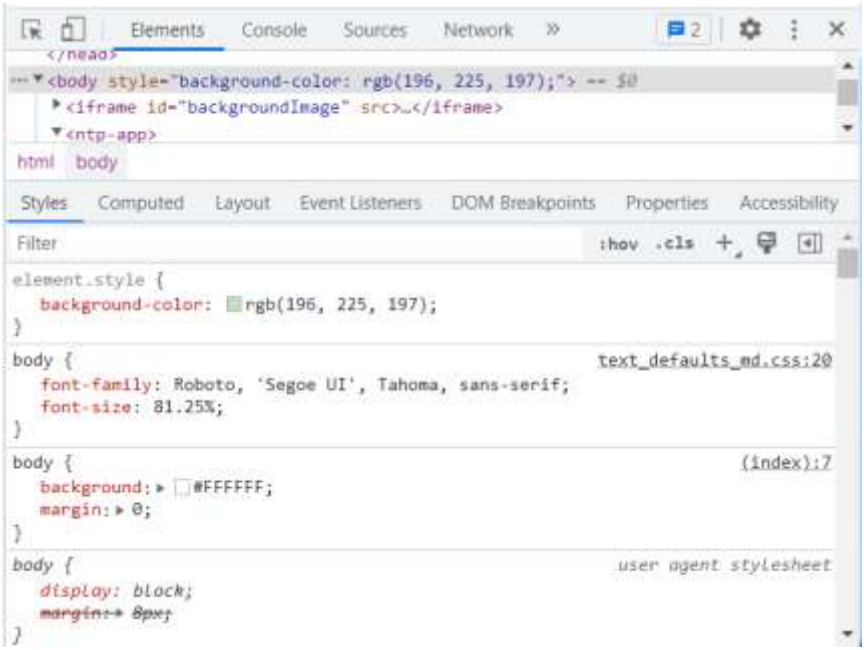
Звичайно, виправлений текст залишиться до перезавантаження сторінки і ніяк не вплине на вебпродукт.

У браузері можна не лише працювати з розміткою, але переглядати та робити тестове редагування стилів. Інформація про стиль знаходиться на вкладці Styles. Праворуч виводяться стилі, вказані розробником, а праворуч від кожного стилю файл і рядок, де вони прописані, а також метрика.

Слід зазначити, що браузери прописують стилі за замовчуванням вузлам DOM. Зазвичай вони відображаються курсивом та позначені як user agent stylesheet. У правилах стилізації, створених

розробником або з популярних бібліотек, зазначені файли з номером рядка, де вони розміщені.

Помилкове правило позначено піктограмою уваги, а яке не діє згідно з принципом каскадування, позначено як перекреслене.

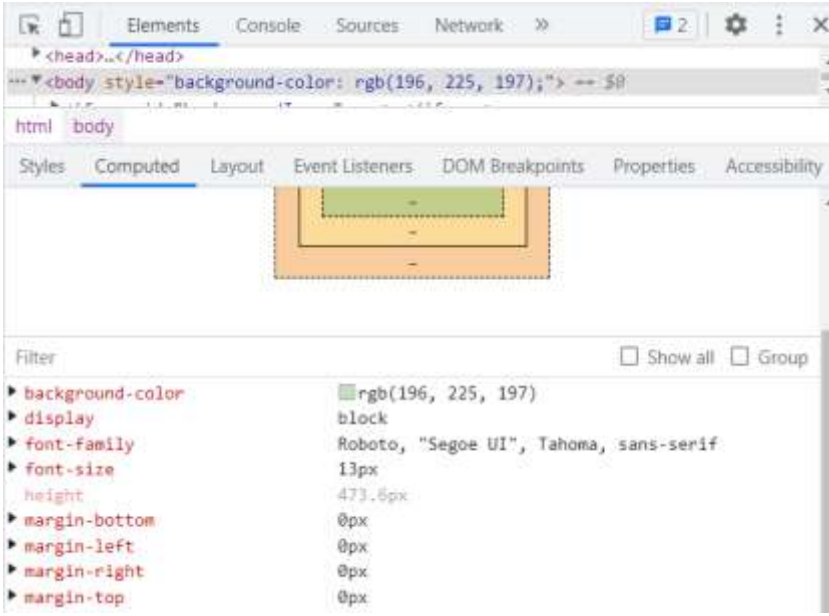


Мал. 34 Відображення стилів та метрики вибраного елемента розмітки

На вкладці Styles продемонстровані правила стилізації з різних джерел, що діють на вузол DOM. Тут зручно переглядати усі правила, фільтрувати за типами, змінювати їх за потреби та навіть додавати нові властивості. Причому результат одразу відображається у браузері, що доволі зручно.

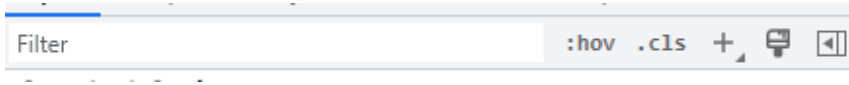
А ось щоб побачити яке правило у підсумку подіяло, краще перейти на вкладку Computed. Слід зазначити, що усі значення розмірів представлені у px, навіть якщо у правилах вони подавались у

відносних величинах. Розкривши кожну із властивостей можна побачити які саме значення були із різних правил та остаточне з них.



Мал. 35 Вкладка Computed

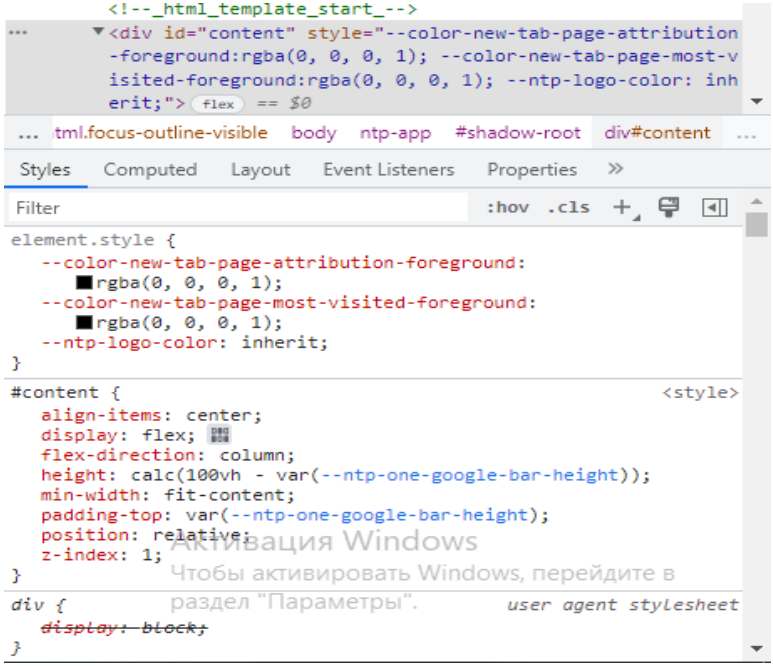
Розглянемо можливості вкладки Styles.



Мал. 36 Панель фільтрів вкладки Styles

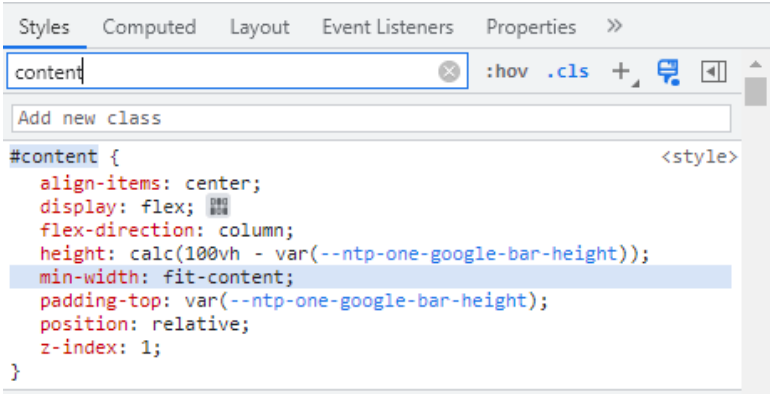
- Filter дає змогу відібрати з усіх правил зазначене;
- :hov дозволяє заначити необхідні стани;
- .cls дає змогу додати новий клас;
- + дозволяє додати нове правило.

Якщо не заданий жодний фільтр, то можна побачити усі властивості, які діють на вибраний вузол розмітки.



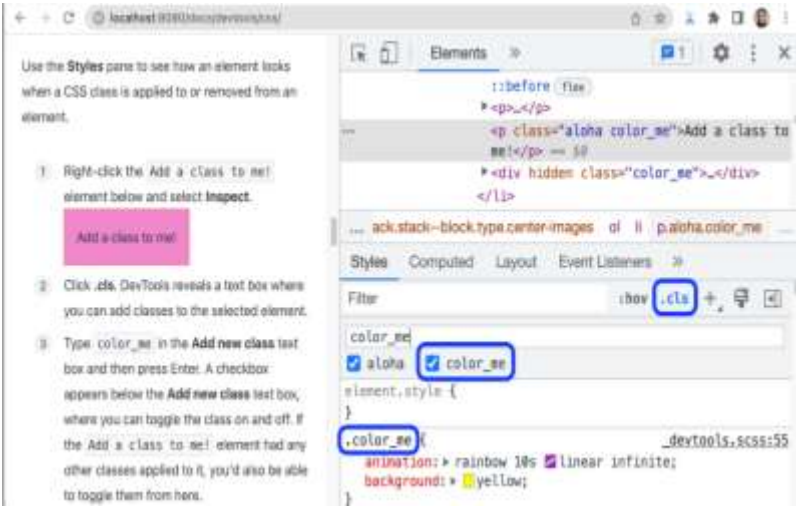
Мал. 37 Відображення усіх правил

Якщо задати, наприклад, значення ідентифікатора виділеного елемента, то бачимо саме це правило



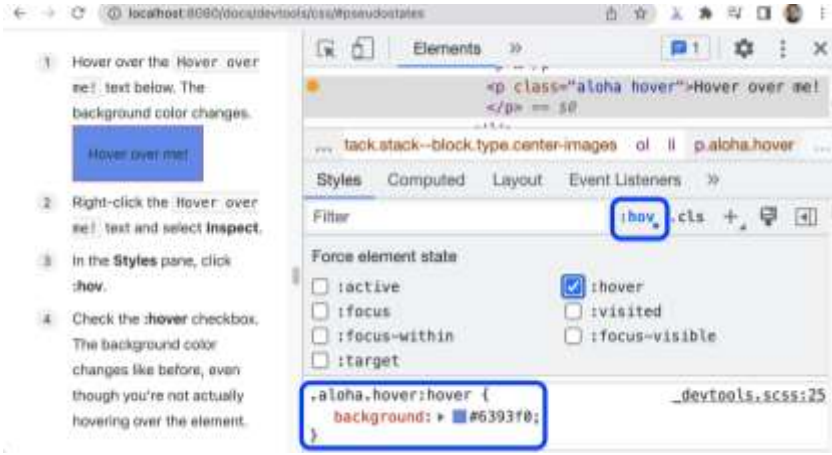
Мал. 38 Дія фільтру

Вкладку Styles використовують також щоб побачити, як виглядає елемент, коли клас CSS застосовано до елемента або видалено з нього. Для цього натисніть .cls . DevTools відкриває текстове поле, де можна додати класи до вибраного елемента. З'являється прапорець, у якому можна вмикати та вимикати клас. Якщо вибраний елемент має будь-які інші класи, застосовані до нього, то також можна перемикати їх звідси.



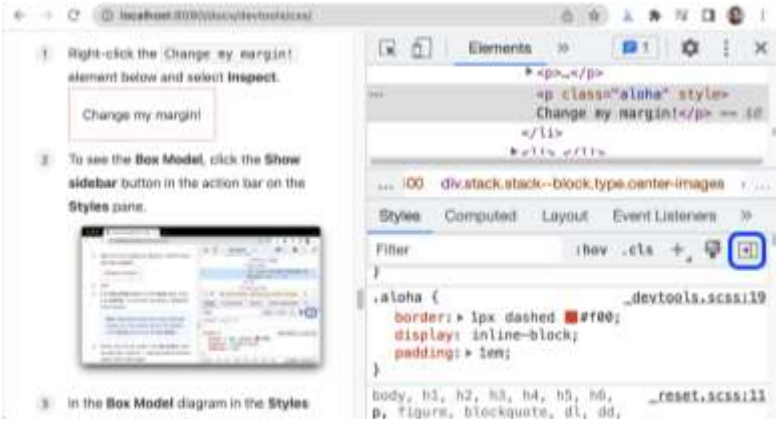
Мал. 39 Робота з класами

Можна також застосувати псевдостан CSS до елемента. DevTools підтримує :active, :focus, :hover, :visited та інші. Слід вибрати вузол DOM, для якого треба перевірити дію стану. На панелі « Стилі » натиснути :hov . Поставити, наприклад, прапорець :hover . Колір фону змінюватиметься, як і раніше, навіть якщо фактично не наводити курсор на елемент. Можна редагувати й задавати інші значення.



Мал. 40 Робота з псевдостанами

На вкладці Styles використовують також інтерактивну діаграму Box Model, щоб змінити ширину, висоту, відступ, поле або довжину рамки вибраного елемента. Щоб побачити модель коробки, слід натиснути кнопку « Показати бічну панель » на панелі дій вкладки Styles.

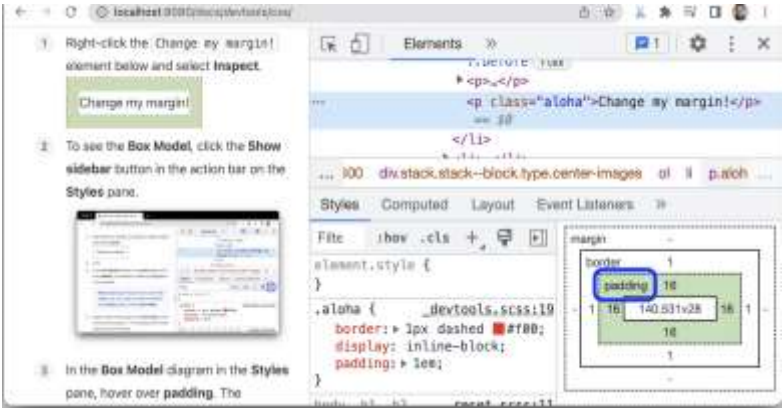


Мал. 41 Перехід до додаткових інструментів

Блокова модель за замовчуванням має пікселі, але вона також набуває інші значення, такі як 1em, 25% або 10vw. На зображені відносні величини переводяться та відображаються в абсолютних – пікселях

Розглянемо деякі дії для роботи з розмірами блокової моделі.

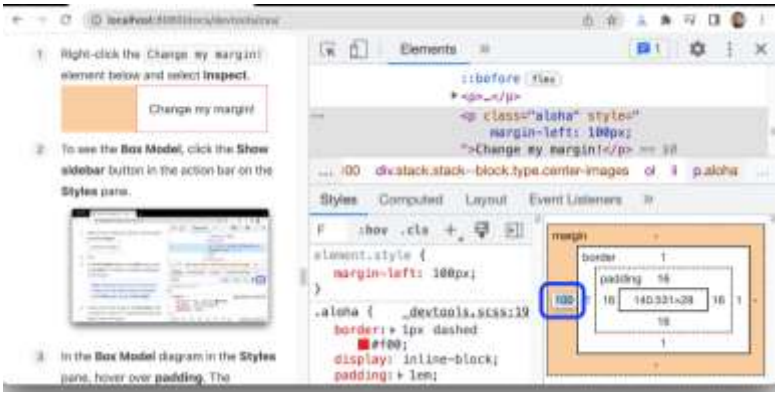
На діаграмі Box Model на панелі Styles наведіть курсор, наприклад, на padding. Відступ елемента виділено у вікні перегляду.



Мал. 42 Робота з Box Model

Бачимо, що відносне значення 1em відобразилось в абсолютній величині 16px.

Можна додавати й відсутні властивості. Двічі натисніть ліве поле в Box Model. Елемент наразі не має полів, тому left-margin має значення -. Введіть 100 і натисніть Enter. Бачимо, що розміри зовнішнього відступу зліва вже додані та вузол змістився на 100зч вправо на сторінці.



Мал. 43 Робота з Box Model

Повну інформацію роботи з усіма інструментами Chrome DevTools можна прочитати на ресурсі розробника [13].

Запитання для повторення

1. Що таке Chrome DevTools та їх призначення?
2. Як можна відкрити Chrome DevTools?
3. Чи можна налаштувати Chrome DevTools під потреби користувача?
4. Як можна налаштувати перегляд вебсторінки під різні пристрої?
5. Як можна додати свій пристрій до перегляду?
6. Як працювати з правилом @media?

7. Що таке DPR і як їх налаштовувати?
8. Як можна вибрати вузол DOM для перегляду?
9. Які дії над вузлом можна робити?
10. Як працювати з правилами стилізації вузлів DOM?
11. Як розрізнити властивості за замовчуванням від браузера та від користувача?
12. Що відображається на вкладці Computed?
13. Як працювати з псевдостанами вузла?
14. Як додати чи вимкнути клас вузла?
15. Як працювати з Box Model?

HTML

Основні поняття

Мова розмітки – це спеціальна комп'ютерна мова, яка дозволяє вставляти в текст спеціальні символи або їх послідовність для передачі інформації про будову текстового документа. Текстовий документ, написаний з використанням мови розмітки, містить не тільки сам текст, але й додаткову інформацію про різні його ділянки: вказівку на заголовки, виділення, списки, посилання тощо. У більш складних випадках мова розмітки дозволяє вставляти в документ зображення, аудіо- та відеовміст, інтерактивні елементи та фрейми зі вмістом інших документів [14].

Розрізняють логічну та візуальну розмітки. У логічній розмітці головне яку роль відіграє певна ділянка документа в його загальній структурі. У візуальній розмітці вже визначається, як саме відображатиметься ця ділянка документа. Ідея мов розмітки полягає у тому, що візуальне відображення документа має автоматично виходити з логічної розмітки та не залежати від його безпосереднього змісту. Це спрощує автоматичну обробку документа та його відображення в різних умовах та на різних пристроях.

Мови розмітки використовуються скрізь, де потрібно отримання форматованого тексту: у видавництві, документообігу, веброзробці тощо.

Серед мов розмітки вирізняють мови, призначені для простого та швидкого додавання форматування у текст. Це так *звані полегшені мови розмітки* (Lightweight markup language). Особливості таких мов:

- мінімум функцій;
- невеликий набір підтримуваних тегів;
- легкість у освоєнні;
- читабельність.

Застосовуються вони там, де людині доводиться готувати текст у звичайному текстовому редакторі (блоги, форуми, вікі), або там, де важливо, щоб користувач із звичайним текстовим редактором також міг прочитати текст.

Серед поширених полегшених мов розмітки є BBCode, Markdown, reStructuredText, Вікі-розмітка, а також різні системи автодокументування. Так, Markdown використовують багато програмістів для написання документації, описів своїх проектів, написання блогів (Jekyll) і так далі. BBCode часто використовується для розмітки повідомлень на форумах. Метою створення було надати можливість користувачам форматувати повідомлення більш простим і безпечним способом, без використання HTML.

Основною мовою розмітки візуального представлення вебсторінок є мова гіпертекстової розмітки **HTML** (HyperText Markup Language).

Гіпертекст – це невпорядкована система текстових документів, між якими є перехресні посилання, які дозволяють переміщатися від одного документа до іншого.

Родоначальником технічної реалізації гіпертексту є **Тім Бернерс-Лі**. Він розробив всі основні концепції глобальної мережі і реалізував їх на початку 90-х років.

Спочатку HTML створювався як платформно незалежна мова для розмітки технічної документації з використанням гіперпосилань, при цьому вся мультимедійна складова була принесена туди набагато пізніше – у п'ятій версії. Тепер HTML забезпечує підтримку мультимедійної інформації виключно за допомогою ресурсів самого браузера.

Розробка HTML у різні часи відбувалась під егідою:

- IETF до 1996 року;
- World Wide Web Consortium (W3C) до 2019 року;
- робочою групою Web Hypertext Application Technology Working Group (WHATWG) до тепер.

За час свого розвитку HTML значно змінилась. Офіційної версії HTML1 не було опубліковано. Наступні специфікації були опубліковані:

- HTML 2 24 листопада 1995 року;
- HTML 3 14 січня 1997 року;
- HTML 4 18 грудня 1997 року;
- HTML 5 28 жовтня 2014 року;

У середині кожної специфікації вносились свої зміни, тому виділяли версії. Так останньою версією специфікації HTML 5, була HTML 5.2 опублікована 14 грудня 2017 року як рекомендація W3C.

Між розробниками були певний час непорозуміння та конкуренція. 28 травня 2019 року W3C оголосила, що WHATWG буде єдиним видавцем стандартів HTML і DOM. W3C і WHATWG публікували конкуруючі стандарти з 2012 року. Хоча стандарт W3C був ідентичним WHATWG у 2007 році, стандарти з тих пір поступово розходилися через різні дизайнерські рішення. WHATWG "Living Standard" деякий час був де-факто вебстандартом.

Зараз діє вебстандарт Living Standard від WHATWG [8].

HTML-документ – це текстовий файл з розширенням .html. Документ, розмічений за допомогою HTML, інтерпретується браузером, у результаті чого користувачі мають змогу бачити не вихідний код з елементами розмітки, а остаточний результат його обробки – вебсторінку. Будь-який HTML-документ являє собою ієрархічну структуру, яка складається зі спеціального набору елементів – тегів. **Тег** – це синтаксична одиниця мови HTML, яка виділяє або створює елемент. Це набір символів, за допомогою якого браузер розуміє, де елемент створюється, починається і закінчується. Кожен тег позначає якусь сутність: заголовок, абзац тексту, список, зображення, мультимедіа, секцію тощо.

Є 2 види тегів:

- **Подвійні (парні) теги** показують початок і кінець елемента. Початок елемента позначається відкриває тегом <...>, а кінець – закриває </ ...>, усередині зазначається його вміст.

<ім'я_тегу> вміст тегу </ім'я_тегу>

- ***Одинарні теги*** не мають пари і складаються лише з початкового тегу. Вмісту тегу немає. Сутність полягає у призначенні тегу, а зміст зазначається через внутрішні елементи – атрибути. Вони слугують для зміни властивостей документа, підключення файлів тощо.

<ім'я_тегу>

Для більшості існуючих тегів можна задати додаткові властивості з використанням атрибутів. **Атрибут** використовується для визначення характеристик html-елемента.

Є два типи атрибутів: атрибут зі значенням та логічний атрибут. Атрибути пишуться всередині тегу, що відкриває, кілька атрибутів перераховуються через пробіл, їхній порядок значення не має.

Імена атрибутів повинні складатися з одного або декількох символів, крім керуючих, таких як пропуск, ", ', >, / і =.

Атрибути зі значеннями складаються з двох частин – ім'я властивості та її значення. Зазвичай значення атрибута поміщають у лапки. Допускаються, як одинарні, так і подвійні лапки.

<ім'я_тегу атрибут= “значення”>

Наприклад, подання подвійних та одинарних тегів з атрибутами

```
<button class="btn" disabled> Виконати <button>
<img src="theater.png" alt="будівля театру" width="500"
height="500">
```

Логічні атрибути можна записувати в трьох видах: без значення, з порожнім значенням і значенням, що збігається з ім'ям атрибута. Усі варіанти рівнозначні, але бажано дотримуватись однієї обраної форми запису.

controls
controls=""
controls="controls"

```
<audio src="audio/music.mp3" controls loop
autoplay></audio>
```

Порядок атрибутів тегу не має значення і на результат відображення елемента не впливає. Також можна переносити атрибути на інший рядок. Наступні записи рівносильні.

```
<img src="photo.png" width="500" height="500" alt="">

<img alt="" height="500" width="500"
 src="title.png">

<img src="title.png"
width="500"
height="500"
alt="">
```

Деякі теги мають свій набір характерних атрибутів, але крім цього існують атрибути, які можна додавати до будь-якого елемента. Тому вони називаються **універсальними** або **глобальними** атрибутами. Наведемо основні універсальні атрибути та їх призначення [8]:

- `accesskey` – дозволяє отримати доступ до елемента за допомогою поєднання клавіш;
- `class` – визначає ім'я класу, що дозволяє пов'язати елемент із стильовим оформленням;

- `contenteditable` – повідомляє, що елемент доступний для редагування користувачем;
- `contextmenu` – встановлює контекстне меню елемента;
- `data-*` – дозволяє створювати свої атрибути для збереження довільної інформації;
- `dir` – задає напрямок та відображення тексту – зліва направо або праворуч наліво;
- `draggable` – вказує, чи можна перетягувати елемент, використовуючи Drag and Drop API;
- `dropzone` – визначає, що робити з даними, що перетягуються користувачем: скопіювати їх, перетягнути або зв'язати;
- `hidden` – приховує вміст елемента від перегляду;
- `id` – указує ім'я стильового ідентифікатора;
- `itemid` – визначає унікальний глобальний ідентифікатор елемента мікроданих;
- `itemprop` – використовується для додавання властивостей мікроданних до елемента;
- `itemref` – зв'язує властивості елемента з атрибутом елемента копіювання;
- `itemscope` – задає область дії словника у структурі даних;
- `itemtype` – вказує адресу словника, який буде застосовуватись для визначення властивостей елемента структури даних;
- `lang` – браузер використовує атрибут для правильного відображення деяких національних символів;
- `spellcheck` – указує браузеру перевіряти чи ні правопис та граматику у тексті;
- `style` – використовується для визначення стилю елемента за допомогою правил CSS;
- `tabindex` – встановлює порядок отримання фокуса під час переходу між елементами за допомогою кнопки Tab;
- `title` – описує вміст елемента у вигляді спливаючої підказки.

Зауваження. 1. Імена тегів та їх атрибутів не чутливі до регістру, це означає, що їх можна писати маленькими або великими літерами. Але у спільноті верстальників та веброзробників прийнято

писати теги та атрибути в нижньому регістрі – name, а не Name чи NAME.

2. Починаючи зі специфікації HTML5 і продовжуючи у Living Standart деякі правила розмітки були ослаблені. Зокрема, використання елементів <html>, <head> і <body> не є обов'язковим для розмітки HTML. Проте браузер все одно вважає, що вони існують. Також дозволяється не використовувати закриваючу зворотну косу рису в деяких елементах без вмісту.

3. Допускається значення атрибутів не поміщати у подвійні чи одинарні лапки. Наприклад, наступні записи рівносильні.

```
<p autocapitalize=sentences> Текст</p>
<p autocapitalize="sentences"> Текст</p>
```

Проте, якщо значення атрибуту складається з декількох окремих слів, його треба брати у лапки, щоб уникнути помилок.

```
<p class="text-center warning"> Текст</p>
```

Тому гарною практикою вважається дотримання строгого написання подвійних і одинарних тегів, а також атрибутів.

Крім тегів у розмітці використовують **коментарі** для того, щоб залишити у вихідному коді пояснення, замітку, тимчасово закоментувати ділянку коду тощо. Вміст коментаря не відображається на вебсторінці. Також коментар може бути багаторядковим. Це зручно для об'ємних описів.

```
<!-- коментар -->
<h1> Заголовок вебсторінки </h1> <!--заголовок-->
<!-- багаторядковий
  коментар.
-->
```

У середині подвійних тегів можуть бути вкладені інші теги. Дотримуючись рекомендацій стандарту HTML, теги можна вклада-

ти один в одного. У разі вкладеності, потрібно дотримуватися порядку їх закриття та вирівнювання. Закриття – для валідності HTML-коду, а вирівнювання – для читабельності.

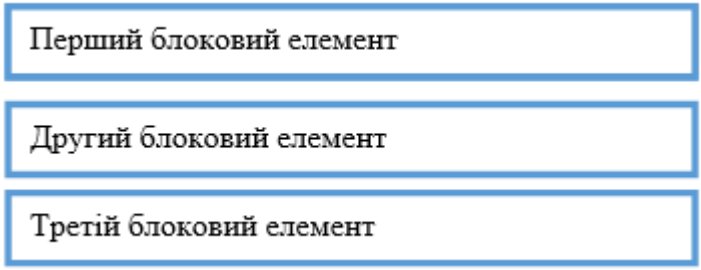
```
<тег1>
   <тег2>
      <тег4>...</тег4>
      <тег5>...</тег5>
       <тег6>...</тег6>
   </тег2>
   <тег3>
       <тег7>...</тег7>
       <тег7>...</тег7>
   </тег3>
</тег1>
```

Потік документа – це вертикальний і горизонтальний порядок відображення елементів на сторінці, що задаються відповідними тегами. Вертикально потік йде зверху вниз, а горизонтальний потік – зліва направо або справа наліво для деяких країн.

За замовчуванням елементи на сторінці відображаються у тому порядку, в якому вони зазначені в HTML-документі. Це пов'язано з тим, що більшість тегів за замовчуванням мають поведінку, за якою їх можна розділити на дві групи – блокові та рядкові (вбудовані).

До основних властивостей блокових тегів належать:

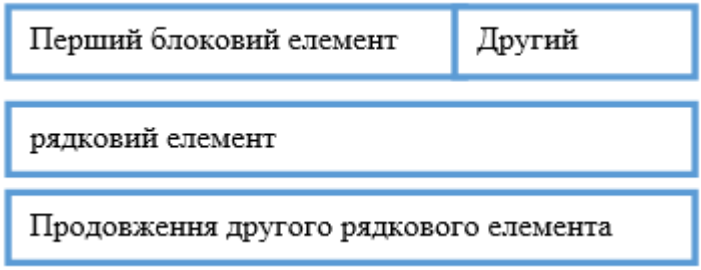
- браузер їх виводить на екран у прямокутні області, що йдуть один за одним зверху вниз;
- починаються з нового рядка;
- ширина визначається шириною батьківського тегу;
- займають весь доступний простір по горизонталі;
- можна задавати ширину, висоту, внутрішні і зовнішні відступи.



Мал. 1 Поведінка блокових тегів за замовчуванням

До основних властивостей рядкових тегів належать:

- містяться усередині блокових і займають місце в рядку, переходячи на наступні рядки, якщо в поточному не вистачило місця; перенесення на новий рядок відбувається тільки після заповнення ширини батьківського елемента;
- до і після рядкового елемента відсутні переноси рядків;
- ширина і висота рядкового елемента залежить тільки від його змісту;
- в основному вони використовуються для зміни вигляду тексту або його смислового виділення.



Мал. 2 Поведінка рядкових тегів за замовчуванням

Зауваження. 1. Поділ елементів на блокові та вбудовані використовувався у специфікації HTML до версії 4.01. Починаючи з HTML5 це протиставлення замінено складнішим набором категорій контенту. Прямої відповідності між поведінкою розміщення елеме-

нтів за замовчуванням та категорією контенту немає. Серед елементів категорії контенту можуть бути як блокові, так і рядкові елементи. **2.** Деякі елементи за своєю поведінкою рядкові, проте їм можна задавати розміри через атрибути width та height. Наприклад, таким є тег img. Це так звані заміщувані елементи з точки зору CSS, тобто зовнішні об'єкти, чиї представлення незалежні від моделі форматування CSS. Типовими елементами, що заміщуються, є img, iframe, video, embed. Деякі елементи розглядаються як такі, що заміщуються тільки в певних випадках: audio, canvas, object. Специфікація HTML також указує, що елемент input може бути заміщуваним, оскільки тег <input> з типом image є елементом, що заміщується на кшталт <img> [15] .

Щоб визначити чи є за поведінкою тег блоковим чи рядковим, та які атрибути, крім універсальних, до нього можна застосовувати, слід користуватись довідниками HTML елементів. Наприклад, довідником [16 – 18].

Серед HTML-елементів слід виділити два універсальні:

- універсальний блоковий контейнер div.
- універсальний рядковий контейнер span.

Наприклад, у [17] зазначено, що тег:

- ** ** – визначає загальний блок вмісту, який не має жодного семантичного значення. Використовується для елементів макета, таких як контейнери. Це не впливає на вміст або макет, доки не буде певним чином стилізовано за допомогою CSS.

```
<div class="warning">
    <img src="/media/examples/photo.jpg"
         alt=" підказка суті зображення.">
    <p>Текст</p>
</div>
```

- **** – визначає загальний вбудований контейнер вмісту, який не має жодного семантичного значення.

```
<h1> Фірма <span lang="en"> WebStudio </span> </h1>
```

Слід зауважити, що у Living Standart визначаються HTML-елементи та описані їх призначення для використання.

HTML-елемент – це термін, що описує якусь семантичну сутність у стандарті HTML. Наприклад, це абзац, заголовок, список, посилання тощо. У HTML-документі HTML-елемент представляється відповідно подвійним або одинарним тегом – синтаксичною конструкцією мови HTML.

Наприклад, у [16] зазначено, що HTML-елементи:

- **div** – HTML-елемент є загальним контейнером для вмісту потоку. Це не впливає на вміст або макет, доки не буде певним чином стилізовано за допомогою CSS;
- **span** – визначає загальний вбудований контейнер вмісту фразування, який не має жодного семантичного значення. Його можна використовувати для групування елементів з метою стилізації. Його слід використовувати лише тоді, коли жоден інший семантичний елемент не підходить.

Отже, за специфікацією визначаємо призначення і вкладеність HTML-елементів, а у довідниках поведінку за замовчуванням відповідних HTML-тегів, а також специфічні речі.

Проте слід зазначити, що поведінку тегу за замовчуванням завжди можна змінити за допомогою CSS.

Короткий опис усіх чинних та експериментальних HTML-елементів з переходом до повного опису можна знайти у [8] .

Не всі символи, які необхідно відобразити у тексті, можна набрати на клавіатурі. Для введення таких знаків використовують символьні підстановки.

Спецсимволи HTML або **символи-мнемоніки**, є конструкцією SGML (Standard Generalized Markup Language – стандартна уза-

гальнена мова розмітки), що посилається на певні символи із символьного набору документа. В основному вони використовуються для вказівки символів, яких немає у стандартній комп'ютерній клавіатурі, або які не підтримують кодування HTML-сторінки (Windows-1251, UTF-8 тощо).

У стандарті кодування Unicode зберігаються знаки майже всіх мов світу, а також спеціальні та службові символи. На підставі них браузер знає, який Unicode-символ потрібно відобразити.

Підстановку у HTML-код можна здійснити декількома способами:

- &мнемокод; – вставлення символу за його «мнемокодом», тобто іменем;
- &#КОД10; – вставлення символу за його десятковим кодом;
- &#xКОД16; – вставлення символу за його шістнадцятковим кодом.

Наприклад, символ авторського права © у текст можна додати одним із способів чином:

```
<p>Усі права захищені &copy; Автор, 2022</p>
<p>Усі права захищені &#169; Автор, 2022</p>
<p>Усі права захищені &#xa9; Автор, 2022</p>
```

Відображення у всіх випадках буде однаковим.

Спецсимволи чутливі до регістру, тому їх потрібно прописувати точно так, як зазначено в таблиці. Спецсимволи, які не мають мнемоніки, можуть не відображатися зовсім або некоректно відображатися у тих чи інших браузерах.

Є багато ресурсів, де можна знайти мнемоніку для більшості символів, наприклад, [8, 19] .

Зазначимо, що спецсимволи можна вставляти не лише у HTML-код, але й як значення властивостей CSS.

Запитання для повторення

1. Що таке мова розмітки і де вона використовується?
2. Що може містити текстовий документ, написаний мовою розмітки?
3. Які види розмітки є? Яка їхня роль?
4. Що таке полегшана мова розмітки? Де зазвичай їх використовують? Назвіть декілька полегшених мов розмітки та місце їх застосування.
5. Що таке гіпертекст?
6. Як розшифровується абревіатура HTML? Який стандарт HTML є чинним?
7. Що таке HTML-документ? З чого він складається?
8. Що таке HTML-елемент? Чим він представляється у розмітці?
9. Види тегів та атрибутів. Призначення тегів і атрибутів та їхні записи.
10. Чи відіграє роль порядок подання атрибутів у тезі?
11. Як позначається коментар у HTM-розмітці? Де його можна зазначати?
12. Яке правило вкладання тегів?
13. Що таке потік документа та яка можлива поведінка розташування тегів за замовчуванням? Основні властивості.
14. Назвіть універсальні блоковий та рядковий теги. Яка їхня семантика?
15. Яке призначення HTML-розмітки?
16. Чи можна змінити поведінку тегу за замовчуванням?
17. Чому у HTML-розмітці використовують символи-мнемоніки?
18. Якими способами можна вставити символ-мнемоніки у HTML-код ?
19. Чи чутливі теги, атрибути тегів та символи-мнемоніки до регістру?
20. Які вимоги до написання HTML-коду?

Специфікація HTML

Специфікація HTML Living Standard – головний документ, який описує стандарти, можливості та майбутній розвиток мови розмітки HTML.

Нижче наведена інформація взята зі специфікації HTML Living Standard [8].

Для вивчення верстки, насамперед важливі розділи, які описують, які є типи елементів, що позначає кожен з них, і розуміння того, як їх можна вкладати один в одного.

HTML-елемент – це термін, що описує якусь семантичну сутність у стандарті HTML. Наприклад, це абзац, заголовок, список, посилання тощо. У HTML-документі HTML-елемент представляється відповідно подвійним або одинарним тегом – синтаксичною конструкцією мови HTML.

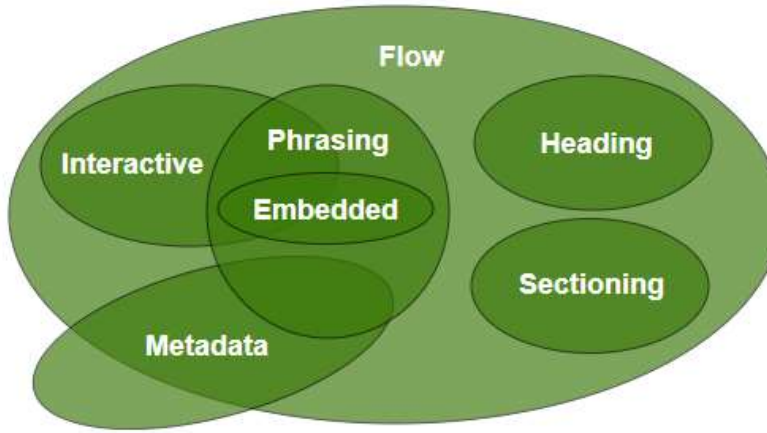
Контентна модель (content model) або модель вмісту визначає, який тип вмісту слід очікувати всередині елемента і які елементи можуть бути вкладені в інші елементи.

Кожен HTML-елемент може не потрапити у жодну або потрапити в одну чи більше категорій, які групують разом елементи зі схожими характеристиками. У цій специфікації використовуються такі широкі категорії:

- Вміст метаданих;
- Вміст потоку;
- Вміст розділів;
- Вміст заголовка;
- Вміст фраз (тексту);
- Вбудований вміст;
- Інтерактивний вміст.

Ці категорії пов’язані таким чином:

- Вміст розділів, вміст заголовків, вміст фразування, вбудований вміст та інтерактивний вміст – це все типи вмісту потоку.
- Метадані іноді є вмістом потоку.
- Метадані та інтерактивний вміст іноді є вмістом фразування.
- Вбудований вміст також є типом вмісту фраз, а іноді є інтерактивним вмістом.
- Інші категорії також використовуються для певних цілей, наприклад, елементи керування формою указуються за допомогою кількох категорій для визначення загальних вимог.
- Деякі елементи мають унікальні вимоги і не вписуються у жодну категорію.



Мал. 3 Категорії специфікації

Наведемо характеристику категорій та елементи, які належать їй.

- **Вміст метаданих** – це вміст, який налаштовує подання чи поведінку решти вмісту, або який встановлює зв'язок документа з іншими документами, або який передає іншу «поза смугою» інформацію. До цієї категорії належать HTML-елементи:
base, link, meta, noscript, script, style, template, title.

- **Вміст потоку.** Більшість елементів, які використовуються у тілі документів і програм, класифікуються як вміст потоку. До цієї категорії належать HTML-елементи:
a, abbr, address, area(якщо він є нащадком map елемента), article, aside, audio, b, bdi, bdo, blockquote, br, button, canvas, cite, code, data, datalist, del, details, dfn, dialog, div, dl, em, embed, fieldset, figure, footer, form, h1,h2, h3, h4, h5, h6, header, hgroup, hr, i, iframe, img, input, ins, kbd, label, link(якщо це дозволено в тілі документа), main(якщо це ієрархічно правильний main елемент), map, mark, MathML, math, menu, meta (якщо присутній атрибут itemprop), meter, nav, noscript, object, ol, output, p, picture, pre, progress, q, ruby, s, samp, script, section, select, slot, small, span, strong, sub, sup, SVG, svg, table, template, textarea, time, u, ul, var, video, wbr, автономні спеціальні елементи, текст.

- **Вміст розділів** – це вміст, що визначається областю заголовків та колонтитулів: header і footer. Кожен елемент секційного вмісту потенційно має заголовок та схему (outline). До цієї категорії належать HTML-елементи:
article, aside, nav, section

- **Вміст заголовка** визначає заголовок розділу (незалежно від того, явно позначений за допомогою елементів вмісту розділу, чи передбачається самим вмістом заголовка). До цієї категорії належать HTML-елементи:
h1, h2, h3, h4, h5, h6, hgroup (якщо він має нащадком елементів h1 – h6).

- **Вміст фраз (тексту)** – це текст документа, а також елементи, які позначають цей текст на рівні внутрішнього абзацу. Більшість елементів, що належать до категорії, можуть містити лише елементи зі своєї категорії, а не будь-які елементи з категорії потокового вмісту.

Текст у контексті моделей вмісту означає або нічого (пробіли), або Text-вузли. Text-вузли та значення атрибутів повинні складатися зі скалярних значень, за винятком не символів, й елементів керування, крім пробілів ASCII.

До цієї категорії належать HTML-елементи:
a, abbr, area(якщо він є нащадком елемента map), audio, b, bdi, bdo, br, button, canvas, cite, code, data, datalist, del, dfn, em, embed, i, iframe, img, input, ins, kbd, label, link (якщо це дозволено в тілі документа), map, mark, MathML, math, meta (якщо присутній атрибут itemprop), meter, noscript, object, output, picture, progress, q, ruby, s, samp, script, select, slot, small, span, strong, sub, sup, SVG, svg, template, textarea, time, u, var, video, wbr, автономні спеціальні елементи, текст.

- **Вбудований вміст** – це вміст, який імпортує інший ресурс у документ, або вміст з іншого словника, який вставляється у документ.

До цієї категорії належать HTML-елементи:
audio, canvas, embed, iframe, img, MathML, math, object, picture, SVG, svg, video.

- **Інтерактивний вміст** – це вміст, який спеціально призначений для взаємодії з користувачем. До цієї категорії належать HTML-елементи:
a(якщо присутній атрибут href), audio(якщо присутній атрибут controls), button, details, embed, iframe, img (якщо присутній атрибут usemap), input(якщо присутній атрибут type не в стані hidden), label, select, textarea, video (якщо присутній атрибут controls).

Зазначимо, що глобальний атрибут tabindex також може зробити будь-який елемент інтерактивним вмістом.

Додаткові категорії вмісту

- **Відчутний вміст.** Як правило, елементи, модель вмісту яких допускає будь-який вміст потоку або вміст фразування, повинні мати принаймні один вузол у своєму вмісті, який є відчутним вмістом і не має зазначеного атрибута hidden.

Відчутний вміст робить елемент непорожнім, надаючи або деякий нащадковий непорожній текст, або щось, що користувачі можуть почути (audio елементи) або переглянути (video, img, або canvas елементи) або іншим чином взаємодіяти (наприклад, інтерактивні елементи керування формою). Однак ця вимога не є жорсткою, оскільки існує багато випадків, коли елемент може бути порожнім, наприклад, коли він використовується як заповнювач, який пізніше буде заповнено сценарієм, або коли елемент є частиною шаблону і на більшості сторінок буде заповнено, але на деяких сторінках це нерелевантно.

До цієї категорії належать HTML-елементи: a, abbr, address, article, aside, audio (якщо присутній атрибут controls), b, bdi, bdo, blockquote, button, canvas, cite, code, data, del, details, dfn, div, dl(якщо дочірні елементи елемента включають принаймні одну групу імені-значення), em, embed, fieldset, figure, footer form, h1, h2, h3, h4, h5, h6, header, hgroup, i, iframe, img, input (якщо атрибут type не в стані hidden), ins, kbd, label, main, map, mark, MathMLmath, menu(якщо нащадки елемента включають принаймні один елемент li), meter, nav, object, ol (якщо нащадки елемента включають принаймні один елемент li), output, p, picture, pre, progress, q, ruby, s, samp, section, select, small, span, strong, sub, sup, SVGsvg, table, textarea, time, u, ul(якщо нащадки елемента включають принаймні один елемент li), var, video, автономні спеціальні елементи, текст (який не є пробілом між елементами).

- **Елементи підтримки сценаріїв** – це елементи, які самі по собі нічого не представляють (тобто вони не відображаються), але використовуються для підтримки сценаріїв, наприклад, для забезпечення функціональності для користувача. До цієї категорії належать HTML-елементи: script, template.

- **Прозора модель вмісту.** Деякі елементи мають прозору модель вмісту. Це означає, що тип очікуваного вмісту успадковується від батьківського елемента. Додатково до власного дозволеного контенту вони можуть містити будь-який контент, який допустимий для батьківського елемента. Якщо такий елемент не має батьківського елемента, то його прозора модель контенту повинна

розглядатися як така, що очікує будь-який вміст із категорії потокового вмісту.

До цієї категорії належать HTML-елементи:
`a, audio, canvas, del, ins, map, object, video`

- **Кореневий секційний вміст.** Крім секційного вмісту, є кілька елементів, що є кореневими секційними. Вони виділені з секційного вмісту, але можуть мати власну структуру заголовків, а розділи і заголовки всередині цих елементів не взаємодіють зі структурою заголовків їхніх батьків.

До цієї категорії належать HTML-елементи:
`blockquote, body, details, dialog, fieldset, figure, td.`

У специфікації описані всі HTML-елементи, їх призначення та рекомендації до використання.

Часто виникають питання, пов'язані з вкладеністю тегів. Для того щоб правильно застосувати вкладення тегів, потрібно слідувати алгоритму:

1) відкрити специфікацію **Living Standart** і відшукати потрібні елементи.
2) перевірити контентну модель елемента, в який вкладаємо.
3) перевірити категорії елемента, який вкладаємо.

Якщо категорія підходить і обмеження не забороняють, то вкладати можна, інакше – ні.

Запитання для повторення

1. Що визначає контентна модель згідно з і специфікацією HTML Living Standard?
2. Назвіть широкі категорії HTML-елементів у специфікацією HTML Living Standard. Чи перетинаються ці категорії?
3. Наведіть характеристику широких категорій.
4. Назвіть додаткові категорії вмісту.
5. Наведіть алгоритм визначення правильності вкладення тегів.

Семантична верстка

Семантика – це розділ лінгвістики, що вивчає смислове значення одиниць мови. Семантика у мові розмітки HTML – це підхід створення сторінок з допомогою певного набору тегів, яким чітко заданий зміст та його конкретне застосування. Іншими словами, семантична верстка – це підхід до розмітки сторінки, у якому наголос зроблено не так на зміст, але на смислове призначення кожного блоку та на логічну структуру всього документа.

До появи стандарту HTML5 уся розмітка сторінок здійснювалася переважно за допомогою тегів , яким надавали класи class або ідентифікатори id для наочності розмітки. Наприклад, верхній колонтитул вебсторінки будувався за допомогою подвійного тегу … . З їх допомогою у HTML-документі розміщували усі основні контейнери – верхні та нижні колонтитули, бічні панелі, навігацію та багато іншого.

Стандарт HTML5, а тепер й Living Standart надав нові елементи для структурування, угруповання контенту та розмітки текстового вмісту. Були введені нові семантичні теги. Ці елементи дозволили поліпшити структуру вебсторінки, додавши змістове значення вмісту в них. Тепер замість … пишуть <header>…</header>, а використовують у випадку необхідної стилізації.

Іншим спонукальним чинником застосування семантики у HTML розмітці стало застосування приладів зчитування тексту. Тому, наприклад, тег фізичного форматування <i> (від англ. italic, що робить текст курсивним) тепер замінюють тегом логічного форматування (від англ. emphasis, акцентування), а тег фізичного форматування <b> (англ. bold, що робить текст напівжирним) тепер замінюють тегом логічного форматування (англ. strong , сильний) у випадках, коли потрібно передати певні емоції при зчитуванні тексту приладами. За допомогою CSS акцентування можна візуально подати курсивом, напівжи-

рним зображенням, підкресленням тощо. Також веброзробники можуть використовувати вбудовані атрибути вказівки для приладів зчитування, зробивши розмітку семантичною. Використання різної розмітки для акцентів, цитат та іноземних слів дозволяє машинним вебагентам більш точно визначати значимість як окремих елементів вебсторінки, так і всього тексту загалом.

Сьогодні поширеною практикою є анотації ARIA, які дозволяють створювати доступні анотації всередині веб-документів. Типові випадки використання включають пропозиції редагування (тобто додавання та/або видалення у редагованому документі) та коментарі (наприклад, редакційний коментар, пов'язаний з частиною документа, що переглядається). ARIA визначає семантику, яку можна застосувати до елементів, розділених на ролі (що визначають тип елемента інтерфейсу користувача) і стани та властивості, які підтримуються роллю. Автори повинні призначити роль ARIA та відповідні стани і властивості елемента протягом його життєвого циклу, якщо елемент уже не має відповідної семантики ARIA (через використання відповідного елемента HTML). Додавання семантики ARIA лише надає додаткову інформацію API доступності браузера та не впливає на DOM сторінки [16]. Якщо сторінка розмічена семантично, це значно спрощує роботу визначення ролей, станів та властивостей ARIA.

Дотримання правил семантики під час верстки сторінок вирішує відразу кілька важливих завдань. Це – відповідність специфікації, доступність людям з обмеженими можливостями, підвищення релевантності сайту з погляду пошукових алгоритмів.

- **Відповідність специфікації.** Семантика кожного тегу чітко прописана у стандарті. Кожен тег має свій опис та чітку роль. Це значно допомагає верстальнику та веброзробнику читати та редагувати розмітку документа.
- **Доступність.** Користувачі з вадами зору не бачать вебсторінки, рухатись їм ними допомагають спеціальні прилади зчиту-

вання з екрану – скрінрідери. Ці програми проходять вебсторінкою і озвучують інформацію, спираючись на вміст і теги, які там є.

Семантика і доступність дуже тісно пов'язані. Такі теги, як <header>, <footer>, <aside>, <h1>, <nav>, <ul>, , , <label> та інші, допомагають людям з обмеженими можливостями сприймати вебсторінку повністю без проблем. Наприклад, якщо використовувати для навігації тег <nav, то пристрій зчитування легко озвучить, що користувач перебуває у ній, причому, якщо елементи навігації оформлені через тег ненумерованого списку <ul> та його елементи <li>, то в такому випадку пристрій повідомить приблизно таке: «навігація => список з такої-то кількості елементів => Про компанію посилання». При цьому звичайно треба до семантичних тегів додавати ARIA анотації, наприклад.

Для людей з вадами моторики важливо при створенні форм заповнення використовувати тег <label>, який дає змогу сфокусувати відповідний елемент її заповнення.

Як бачимо, такі люди отримають повну, зручну для них інформацію та зможуть користуватися сайтом у повній мірі.

- **Підвищення релевантності сайту з погляду пошукових алгоритмів.** Основна мета та завдання верстки – це передати зміст структурних блоків, з яких складається сайт. Для людини легко розділити сайт на смислові блоки, роботи ж спираються на теги, написані розробниками.

Існують пошукові роботи, які щодня аналізують безліч сайтів в інтернеті. Вони зчитують код, одержують відомості, передають дані на пошукові сервери тощо. Таким чином, сайт ранжується у пошуковій видачі. Пошуковики не розголошують правила ранжування, але відомо, що наявність семантичної розмітки сторінок допомагає пошуковим роботам краще розуміти, що розміщено на сторінці, і в залежності від цього ранжувати сайти в пошуковій видачі.

Прикладами не семантичних елементів, за якими важко визначити зміст є div і span. Проте семантичні елементи section, article, address, time, form, table, img, figure тощо чітко визначають

їх зміст. Але й применшувати значення використання div не можна [20].

Отже, семантичні елементи HTML доступно описують свій зміст або призначення як для браузерів, пошукових систем, приладів зчитування, так і для веброзробників.

Семантичні теги підтримуються усіма сучасними браузерами, крім експериментальних, які незабаром також можна буде використовувати скрізь.

Для того, щоб розмітити сторінку з погляду семантики, процес розмітки можна розділити на кілька кроків із різним ступенем деталізації.

1. Вивчити призначення та вкладеність кожного HTML-елемента.
2. Виділити великі смислові блоки на кожній сторінці вебпродукту. Теги: <header>, <main>, <footer>.
3. Виділити окремі розділи усередині смислових блоків. Теги: <nav>, <section>, <article>, <aside>.
4. Визначити заголовок усього документа та заголовки смислових розділів. Теги: <h1> – <h6>.
5. Виділити дрібні елементи у смислових розділах: списки, таблиці, демо-матеріали, параграфи та переноси, форми, цитати, контактна інформація тощо. Теги: <ul>, <ol>, <dl>, <table>, <p>,
, <form>, <blockquote>, <quote> ,<address> тощо.
6. Виділити зображення, посилання, кнопки, відео, час та дрібні текстові елементи. Теги: <img>, <audio>, <video>, <button>, <time>, <a> тощо.

Зазвичай базовою розміткою сторінки є:

```
<body>
  <header>...</header>
  <main>...</main>
```

```
  <footer>...</footer>
</body>
```

Далі вона розгортається у залежності від вмісту

```
<body>
  <header>
       це банер
       <nav>...</nav>
  </header>
  <main>
    <header>
     це не банер, може бути відсутнім
    </header>
    ...
  </main>
  <footer>
    <nav>...</nav>
        ...
  </footer>
</body>
```

Смисловий блок main розбивають на article, section, div, aside за потреби

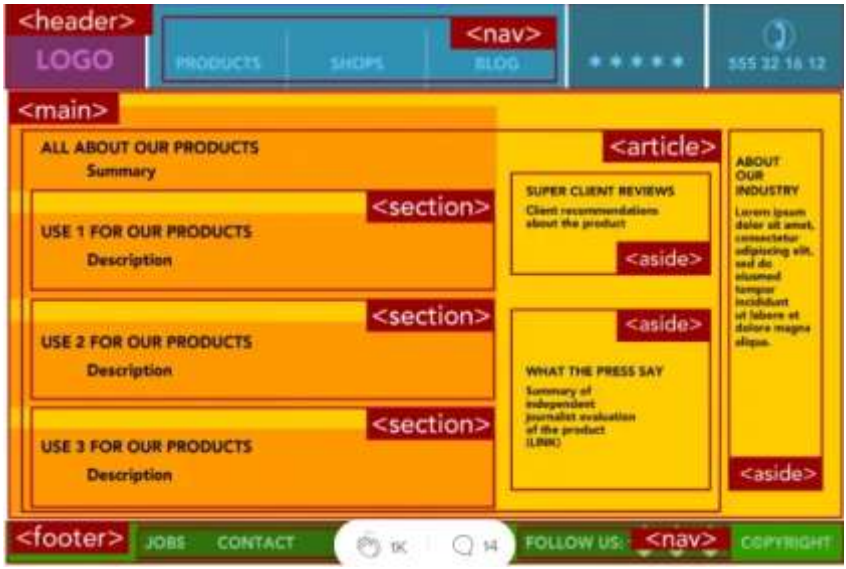
Існують прості правила для вибору потрібних тегів.

1. Якщо вдалось знайти найкращий смисловий тег, то слід використовувати його. Інакше для потокових контейнерів використовують , а для дрібних фразових елементів (слово чи фраза) - .
2. Правило для визначення <article>, <section> та :

1) якщо визначений розділ можна винести на іншу сторінку без втрати змісту сторінки, то це <article>. Яскравим прикладом є стаття у блозі, певний віджет, твіт тощо;

2) якщо визначений розділ не можна винести на іншу сторінку без втрати змісту сторінки, то це <section>. Найчастіше використовується у вигляді тематичної групи контенту із заголовком;
3) У інших випадках це .

Розглянемо ілюстративне зображення



Мал. 4 Ілюстративне зображення макету вебсторінки

Базовою розміткою для неї може бути

```
<body>
  <header>
    <nav>...</nav>
    ...
  </header>
```

```
    <main>
      <article>
        <div>
          <section>...</section>
          <section>...</section>
          <section>...</section>
        </div>
        <div>
          <aside>...</aside>
          <aside>...</aside>
        </div>
      </article>
      <aside>...</aside>
    </main>
    <footer>
      <nav>...</nav>
      ...
    </footer>
  </body>
```

Це лише один ілюстративний приклад. Для кожного конкретного макету сторінки слід детально продумувати розмітку із врахуванням доступності ARIA Anotations.

Слід розуміти, що у <section> можуть бути вкладені <article> і навпаки, <article> може містити декілька <section>. Щоб вибрати між <article> і <section>, потрібно проаналізувати вміст.

Також слід пам'ятати, що семантичні теги використовуються для логічної структури, а не для стилізації. Тому вони можуть бути обгорнуті, чи всередині містити універсальний блоковий тег .

Запитання для повторення

1. Що таке семантика у мові розмітки HTML?
2. Які завдання вирішує дотримання правил семантики при побудові HTML-коду?
3. Охарактеризуйте як пов’язані поняття семантика HTML-коду та доступність вебпродукту?
4. Охарактеризуйте як пов’язані поняття семантика HTML-коду та пошукові алгоритми?
5. Назвіть загальні підходи розмітки з погляду семантики.
6. Наведіть правила вибору потрібних тегів для розмітки вмісту вебсторінки.
7. Наведіть правило визначення <article>, <section> та .
8. Як можуть вкладатись теги <article>, <section>? З чим це пов’язано?
9. Що таке ARIA анотації та як вони пов’язані із семантикою HTML-коду?
10. Як впливає семантика на стилізацію?

Структура документа та метадані

На початку кожного документа вказується тип документа, який пояснює, яку версію HTML слід очікувати. По-перше, валідатори знають, відповідність якій специфікації вони повинні перевіряти цей документ. По-друге, тип документа також служить для того, щоб браузер відображав сторінку в так званому стандартному режимі. У стандартному режимі браузери зазвичай намагаються відобразити сторінку відповідно до специфікацій CSS, тобто передбачається, що документ створено з урахуванням вебстандартів.

Раніше в HTML були версії, остання – HTML5. Зараз **HTML Living Standard** – єдина специфікація мови HTML, у якій відмовилися від версій. Вона постійно оновлюється. Тобто HTML5 – це те саме, що «сучасний HTML» або HTML Living Standard.

Для HTML Living Standard тип документа визначається інструкцією оголошення `<!DOCTYPE html>`, яка ставиться на початку. Інструкція оголошення типу документа не є тегом HTML і не чутлива до регістру.

Базовий HTML-документ виглядає так:

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

    <head>
      <meta charset="utf-8">
      <title>Назва документа</title>
      <link rel="stylesheet" href="style.css">
    </head>

    <body>
      <!-- тіло сторінки -->
    </body>
```

```
</html>
```

Розглянемо кореневий елемент та елементи метаданих, які стосуються структури за специфікацією Living Standart [8].

Елемент html

Категорії вмісту: жодна.

Контекст, у якому цей елемент може бути використаний: кореневий елемент HTML-документа. Також скрізь, де дозволено фрагмент піддокумента у складеному документі, наприклад, усередині iframe.

Пропуск тегів: початковий тег <html> може бути пропущений, якщо за тегом не йде коментар. Тег </html> може бути пропущений, якщо перед ним немає коментаря.

Тобто допустима структура

```
<!DOCTYPE html>
    <head>
      <meta charset="utf-8">
      <title>Назва документа</title>
      <link rel="stylesheet" href="style.css">
    </head>

    <body>
      <!-- тіло сторінки -->
    </body>
```

Проте на практиці прийнято зазначати цей елемент, оскільки у ньому доцільно використовувати певні атрибути.

Елемент html є коренем HTML-документа (елемент верхнього рівня). Рекомендується вказувати атрибут lang із зазначенням мови документа. Це допомагає інструментам синтезу мови визначення вимови, інструментам перекладу визначення правил перекладу тощо.

Усі інші елементи мають бути нащадками елемента html. Все, що є за межами <html>...</html>, не сприймається браузером як HTML-код, і ніяк не обробляється.

Для елемента html доступні глобальні атрибути й атрибут manifest, який указує шлях до документа кеша маніфесту. У ньому перераховуються ресурси, які мають бути збережені в локальному кеші, наприклад:

```
<html manifest="list_events.appcache">
```

Згідно зі специфікацією Living Standart до метаданих відноситься вміст, який встановлює подання або поведінку іншого вмісту, відношення документа з іншими документами, або передає іншу «зовнішню» інформацію.

Це: `base, link, meta, noscript, script, style, template, title.`

Елемент head

Категорії вмісту: жодна.

Контекст, у якому цей елемент може бути використаний: як перший елемент в елементі html.

Пропуск тегів: початковий тег <head> може бути пропущений, якщо елемент head порожній, або якщо після нього йде інший HTML-елемент. Тег </head> може бути пропущений, якщо він не слідує відразу за пробілом або за коментарем.

Для елемента доступні глобальні атрибути.

Розділ <head>...</head> містить набір технічної інформації (метаданих) про поточну веб-сторінку: заголовок, опис, ключові слова для пошукових машин, кодування тощо. Введена інформація не відображається у вікні браузера, проте містить дані, які вказують браузеру, як слід обробляти сторінку.

Набір метаданих може бути як великим, так і маленьким, тобто містити будь-яку кількість необхідних метаданих

Наприклад,

```
<head>
     <meta charset="UTF-8">
     <meta name="viewport" content="width=device-width,
     initial-scale=1">
     <title> Заголовок сторінки</title>
     <base href="https://www.example.com/">
     <link rel="shortcut icon" href="img/logo.ico"
     type="image/x-icon">
     <link
     href="https://fonts.googleapis.com/css?family=Lato
     |Roboto:300,400,500&display=swap"
     rel="stylesheet">
     <link rel="stylesheet" href="styles/main.css">
     <script src="scripts/main.js" defer></script>
</head>
```

Елемент meta

Категорії вмісту: метадані.

Контекст, у якому цей елемент може бути використаний: якщо для елемента вказані атрибути charset і http-equiv, то в тезі <head>. Якщо значення атрибуту не є content-type, то всередині тегу <noscript>, що є дочірнім <head>. Якщо є атрибут name, то там, де очікуються метадані.

Пропуск тегів: відсутній тег, що закриває.

Для елемента доступні глобальні атрибути, а також атрибути:

- `charset` – визначає кодування символів у документі. У документі має бути один тег <meta> з атрибутом charset. Необхідно використовувати utf-8 або інше кодування, сумісне з ASCII;
- `content` – задає значення метаданих документа чи прагма директив;

- `http-equiv` – задає прагму директиву;
- `name` – встановлює назву/ім'я метаданих документа.

Елемент meta представляє різні види метаданих, які не можуть бути виражені з використанням елементів title, base, link, style та script

Для елемента обов'язково має бути визначений один із атрибутів: name, http-equiv або charset. Якщо зазначений атрибут name або http-equiv, також має бути присутнім атрибут content (або пропущений, якщо немає відповідних значень).

Мета-тегів може бути кілька, оскільки залежно від використаних атрибутів вони вказують різну інформацію.

Атрибут charset визначає кодування символів у документі Кодування сторінки необхідно вказати для того, щоб браузер коректно відобразив текст. Якщо цього не зробити або задати некоректне кодування, то замість символів браузер може відобразити ієрогліфи.

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

Атрибут content може набувати такі значення:

- `no-referrer` – не передає жодної інформації про реферера;
- `no-referrer-when-downgrade` – передає реферальні дані тільки сайтам на HTTPS. Поводження браузера за замовчуванням, якщо не вказано інше.
- `unsafe-url` – завжди передає повний URL реферера;
- `origin-when-cross-origin` – надсилає повну URL-адресу при переході на сторінки в рамках одного сайту, незалежно від протоколу, а на решту – лише базовий домен/піддомен;

- `viewport` – дозволяє визначати конкретні характеристики області перегляду, зокрема, ширину макета та коефіцієнт масштабування веб-сторінок у загальному або конкретному випадку або повна заборона масштабування.

Наприклад,

```
<meta name="viewport" content="width=device-width,
initial-scale=2">
```

указує на те, що розмір сторінки в браузері буде дорівнює 100% величини області перегляду, а співвідношення між фізичним пікселем і css пікселем буде 1: 2.

Код

```
<meta name="viewport" content="width = device-width,
max-imum-scale = 3, minimum-scale = 0.5">
```

дозволить збільшувати ширину сторінки до величини, що дорівнює 3-х кратної ширині екрану пристрою і зменшувати її до половини ширини екрану пристрою.

Позбавити користувачів можливості масштабування можна за допомогою user-scalable:

```
<meta name="viewport" content="initial-scale = 1.0,
user-scalable = no">
```

Якщо в тезі <meta> вказано атрибут http-equiv, елемент meta є прагма-директивою, яка надає додаткову інформацію про документ:

- `content-type` – є альтернативною формою встановлення атрибуту charset.

```
<meta http-equiv="content-type" content="text/html;
charset=utf-8">
```

- `default-style` – вказує назву альтернативної таблиці стилів, яка використовується за замовчуванням.

```
<meta http-equiv="default-style" content="default">
```

- `refresh` – встановлює таймер для оновлення та перенаправлення. Наприклад, головній сторінці сайту новин треба забезпечити автоматичне перезавантаження на сервері кожні десять хвилин для оновлення даних:

```
 <meta http-equiv="refresh" content="600">
```

Послідовність сторінок може використовуватись як автоматичне слайд-шоу, якщо кожна сторінка оновлюється до наступної сторінки в послідовності з використанням наступної розмітки:

```
<meta http-equiv="refresh" content="20;
url=pageNumber.html">
```

- `content-security-policy` – дозволяє налаштувати політику захисту вмісту, за допомогою якої можна захищатись, наприклад, від міжсайтового скриптингу:

```
<meta http-equiv="content-security-policy"
content="script-src 'self'">
```

Атрибут name може набувати такі значення, які є чутливими до регістру:

- `application-name` – значення має бути коротким рядком довільної форми, що містить назву веб-програми, яку представляє сторінка. В одному документі має бути не більше однієї назви вебпродукту. Браузери можуть використовувати назву вебпродукту в інтерфейсі користувача замість <title>, оскільки <title> може міс-

тити повідомлення про стан тощо, що стосуються стану сторінки в певний момент часу, а не просто як назва програми.

- `author` – значення має бути рядком довільної форми із зазначенням імені одного з авторів сторінки.
- `description` – значення має бути рядком довільної форми, що описує сторінку у пошуковій системі.
- `generator` – значення має бути рядком довільної форми, що ідентифікує один із пакетів програмного забезпечення, використаних для створення документа. Наприклад,

```
<meta name="generator" content="WordPress 6.0.3">
```

- `keywords` – значення має бути набором розділених комами ключових слів, які стосуються сторінки. Багато пошукових систем не розглядають такі ключові слова, тому що ця функція історично використовувалася ненадійно і вплинула на результати пошуку.
- `referrer` – необов'язкове поле заголовка HTTP, яке дозволяє відстежувати переміщення користувачів між сторінками в інструментах аналітики, а також зрозуміти походження вхідного трафіку. Реферер передається при переході з http на будь-який тип сайту, при переході з https на https, і не передається під час переходу з https на http.

Елемент title

Категорії вмісту: метадані.
Контекст, у якому цей елемент може бути використаний: елемент head, що не містить інших елементів title.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути.

Елемент title представляє назву або назву документа (вебсторінки). Потрібно використовувати заголовки, які дають пошуковій системі зрозуміти, що міститься на сторінці, навіть якщо заголовки використовуються поза контекстом, наприклад, в історії,

закладках користувача або результатах пошуку. Заголовок документа може відрізнятись від заголовка першого рівня, оскільки <h1> повинен стояти окремо, коли він вирваний із контексту.

Текст усередині <title>… </title> відображається браузером у заголовку вікна. Також цей текст міститиме посилання на ваш сайт на сторінці результатів пошуку. Довжина заголовка повинна бути не більше 60 символів, щоб розміститися повністю.

В одному документі має бути не більше одного одного тегу <title>. Тег <title> є обов'язковим у більшості ситуацій, але якщо протокол вищого рівня надає інформацію про заголовок, наприклад, у рядку «Тема» електронного листа, коли HTML використовується як формат створення електронного листа, тег <title> може бути опущений.

Елемент base

Категорії вмісту: метадані.
Контекст, у якому цей елемент може бути використаний: елемент head, що не містить інших елементів base.
Пропуск тегів: відсутній тег, що закриває.
Для елемента доступні глобальні атрибути.

Тег <base> повинен розміщуватися перед будь-якими іншими елементами в дереві, які мають атрибути, визначені як такі, що приймають URL-адреси, крім тегу <html> (його атрибут manifest не піддається впливу елемента base).

В одному документі може бути тільки один тег <base> і він повинен мати атрибут href або разом з атрибутом target.

Елемент base за допомогою атрибута href надає базовий URL документа для парсингу всіх відносних URL-адрес на сторінці, встановлених атрибутами src та href. Атрибут target визначає тип вікна перегляду за умовчанням при переході по всіх гіперпосиланнях.

Елемент link

Категорії вмісту: метадані. Якщо його використання дозволено елементом body, то потоковий або текстовий вміст.

Контекст, у якому цей елемент можна використовувати: де очікуються метадані. Дозволено в <noscript>, що є дочірнім тегом тегу <head>. Якщо елемент дозволено <body> то там, де очікується текстовий вміст.

Пропуск тегів: відсутній тег, що закриває.

Для елемента доступні глобальні атрибути, а також атрибути: `href, crossorigin, rel, rev, media, nonce, hreflang, type, referrerpolicy, sizes, title.`

Атрибут href тегу <link> дозволяє зв'язувати HTML-документ з різними видами ресурсів, наприклад, таблицями стилів, скриптами, альтернативними формами документа та посиланнями навігації (зміст, попередні та наступні сторінки, повідомлення про авторські права тощо).

Тип пов'язаного ресурсу задається значенням обов'язкового атрибута rel.

За допомогою тегу <link> можна створити дві категорії посилань: посилання на зовнішні ресурси та гіперпосилання.

Наприклад, посилання на зовнішній файл стилізації сторінки

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

Наступний елемент посилання створює два гіперпосилання (на ту саму сторінку):

```
<link rel="author license" href="/about">
```

Семантика першого полягає в тому, що цільова сторінка містить інформацію про автора поточної сторінки, семантика другого полягає в тому, що цільова сторінка містить інформацію про ліцензію, під якою надається поточна сторінка.

Гіперпосилання, створені за допомогою тегу <link> та його атрибута rel, застосовуються до всього документа. Це відрізняється від атрибута rel тегів <a> і <area>, який вказує тип посилання, контекст якого визначається розташуванням посилання у документі. Якщо значення атрибута rel містять лише ключові слова, дозволені в <body>, тег <link> можна використовувати там, де очікується фразовий зміст, тобто всередині <body>.

Використання значення preload для атрибута rel дає змогу завантажувати файли одразу, щоб вони були доступні, як тільки знадобляться для відтворення сторінки пізніше. Наприклад:

```
<link rel="preload" href="style.css" as="style" />
<link rel="preload" href="main.js" as="script" />
```

Дають змогу прискорити роботу з:

- ресурсами, на які вказує CSS, як-то шрифти чи зображення;
- ресурсами, які може запитувати JavaScript, наприклад JSON, імпортовані сценарії або вебворкери;
- великими зображеннями та відеофайлами.

Атрибут preload має й інші переваги. Використання as для визначення типу вмісту для попереднього завантаження дозволяє браузеру точніше визначте пріоритети завантаження ресурсів.

Ще одним прикладом використання елемента link є додавання іконки сайта. Щоб біля назви сторінки відображалось зображення (іконка сайта) також використовується тег <link>

```
<head>
  <title> Моя сторінка</title>
  <link rel="SHORTCUT ICON" href="/favicon.ico"
  type="image/x-icon">
</head>
```

де favicon.ico – назва зображення розміром 16x16 px і зазвичай розташовується у кореневому каталозі разом зі сторінкою.

Крім розширення ico є й інші: png, jpeg, gif, анімований gif та інші. Проте слід дивитись на підтримку браузерів цих розширень.

Розширення .ico та .gif підтримуються сучасними браузерами.

```
<link rel="SHORTCUT ICON" href="/favicon.gif"
type="image/gif">
```

Елемент style

Категорії вмісту: метадані.

Контекст, у якому цей елемент можна використовувати: де очікуються метадані. Всередині елемента noscript є дочірнім елементом head . Усередині body, де очікується потоковий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути, а також атрибути:

- `media` – вказує, до яких медіа застосовуються стилі. Значення має бути допустимим списком медіазапитів. Якщо атрибут media пропущено, за замовчуванням він набуває значення all, тобто стилі застосовуються до всіх видів медіа;
- `nonce` – представляє одноразовий криптографічний номер, який може використовуватися політикою безпеки вмісту, щоб визначити, чи буде стиль, указаний елементом, застосовуватися до документа;
- `type` – встановлює мову таблиць стилів, значення має бути допустимим типом MIME. Значення за замовчуванням є text/css;
- `title` – визначає альтернативне ім'я таблиць стилів;

Якщо тег <style> використовується, то бажано використовувати його всередині розділу head. Цей підхід застосовується для оптимізації швидкості відтворення сторінки, додаючи найважливіші стилі безпосередньо в HTML-документ. Ця просунута техніка називається Critical CSS.

Структура документа HTML складається з розділів і підрозділів. Розділи можуть бути представлені у вигляді схем документа за

аналогією зі змістом. Кожен розділ має власну схему, тому кожен розділ можна починати зі заголовка.

Схема складається зі списку одного або кількох вкладених розділів. Розділ є контейнером, який відповідає деяким вузлам у вихідному дереві DOM. Розділ у разі не є елементом section, він має на увазі його концепцію. Кожен розділ може мати один заголовок, пов'язаний із ним, а також будь-яку кількість додаткових вкладених розділів.

Також окремо виділяють кореневі секційні елементи. Вони відрізняються від секційних елементів, але можуть мати схему. Розглянемо елементи, що формують розділи HTML-документу.

Елемент body

Категорії контенту: кореневий секційний.

Контекст, у якому цей елемент може бути використаний: як другий елемент в елементі html.

Пропуск тегів: початковий тег <body>може бути опущений, якщо елемент порожній, або якщо першим елементом, що всередині елемента body, не є пробіл або коментар, а також, коли перше, що не йде за тегом <body>, є елементи meta, link, script або style.

Тег </body> може бути опущений, якщо перед ним немає коментаря.

```
<!DOCTYPE html>
    <head>
      <title>Назва сторінки</title>
    </head>
    <body>
      <h1>Заголовок сторінки</h1>
      <p>Текст</p>
    </body>
```

або

```
<!DOCTYPE html>
    <title>Назва сторінки</title>
    <h1>Заголовок сторінки</h1>
    <p>Текст</p>
```

Елемент body надає вміст документа. Для елемента доступні глобальні атрибути, а також атрибути, що пов'язані подіями об'єкту Window.

Елемент header

Категорії вмісту: потоковий вміст, відчутний вміст.
Контекст, у якому цей елемент можна використовувати: де очікується потоковий вміст.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути.

Елемент header представляє вступний вміст для його найближчого предка – елемента main або елемента з категорії секційного вмісту або кореневого секційного елемента. Елемент header зазвичай містить групу вступних або навігаційних елементів.

Якщо для елемента header найближчим предком є елемент body і не знаходиться усередині main, він представляє вступний вміст для сторінки в цілому.

```
<body>
  <header> це банер </header>
  <main>
    <header> це не банер </header> <!-- може бути
     відсутнім -->
    ...
  </main>
  <footer>...</footer>
</body>
```

Елемент header зазвичай містить назву розділу (елементи h1 – h6), але це не обов'язково. Тег <header> також можна використовувати як обгортку елемента для змісту розділу, форми пошуку або будь-яких доречних логотипів. У документі може міститися одночасно кілька елементів header і вони можуть розташовуватись у будь-якій частині сторінки.

Елемент header не є секційним вмістом, не вводить новий розділ.

```
<article>
  <header>
    <h2>Заголовок статті</h2>
    <p> Анонс статті</p>
  </header>
  <p>Вступна частина</p>
  <section>
    <header>
      <h3>Заголовок розділу</h3>
      ...
    </header>

    <p>Текст розділу </p>
    ...
  </section>
  <section>
    <header>
      <h3>Заголовок розділу</h3>
      ...
    </header>

    <p>Текст розділу </p>
    ...
  </section>
</article>
```

Елемент footer

Категорії вмісту: потоковий вміст, відчутний вміст.
Контекст, у якому цей елемент можна використовувати: де очікується потоковий вміст.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути.

Елемент footer представляє нижній колонтитул для його найближчого предка елемента main або елемента з категорії секційного вмісту або кореневого секційного елемента.

Зазвичай містить інформацію про автора статті, дані про копірайт і т.д. Якщо використовується як колонтитул усієї сторінки, вміст доповнюється відомостями про авторські права, посилання на умови використання, контактну інформацію, посилання на пов'язаний вміст тощо.

```
<body>
  <header>...</header>
  <main>
    <header>
      <h1>Головний заголовок сторінки</h1>
    </header>
    <article>
      <h2>Заголовок</h2>
      <p>Текст</p>
      <footer> Висновки</footer>
    </article>
    <article>
      <h2>Заголовок</h2>
      <p>Текст</p>
      <footer> Висновки</footer>
    </article>
  </main>
```

```
  <footer>...</footer>
</body>
```

В одному вебдокументі може бути кілька тегів <footer>. Як кожна сторінка, так і кожна стаття може мати тег <footer>. Також у footer можна помістити елемент blockquote, щоб указати джерело цитування.

Елемент article

Категорії вмісту: потоковий вміст, секційний вміст, відчутний вміст.

Контекст, у якому цей елемент можна використовувати: де очікується потоковий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути.

Елемент article є завершеним або автономним твором у документі, сторінці, додатку або сайті. Може дублюватися на інших сторінках сайту та містити всередині інші елементи article, які за змістом мають близьке відношення до змісту зовнішньої статті. Якщо на сторінці є лише одна стаття із заголовком та текстовим вмістом, вона не потребує обгортки елементом article.

Загальне правило полягає у тому, що елемент article доречним тільки в тому випадку, якщо вміст елемента буде вказано в схемі документа. Кожна стаття повинна бути ідентифікована, зазвичай шляхом включення заголовка (теги <h1> – <h6>) як дочірній елемент тегу <article>.

Елемент section

Категорії вмісту: потоковий вміст, секційний вміст, відчутний вміст.

Контекст, у якому цей елемент можна використовувати: де очікується потоковий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути.

Елемент section представляє загальний розділ документа чи програми, групуючи тематичний вміст. Кожен розділ повинен бути ідентифікований, зазвичай шляхом включення заголовка (теги <h1> – <h6>) як дочірній елемент у тезі <section>.

Прикладами розділів можуть бути розділи, сторінки у вкладках або пронумеровані розділи. Домашня сторінка вебсайту може бути розбита на розділи для вступу, новин та контактної інформації.

Авторам рекомендується використовувати елемент article замість елемента section, коли контент завершений або самодостатній.

Тег <section> не є універсальним контейнерним елементом. Коли елемент потрібний лише для стилізації або для зручності написання сценаріїв, рекомендується використовувати тег . Загальне правило полягає в тому, що елемент section доречний тільки в тому випадку, якщо вміст елемента буде вказано в схемі документа.

```
<body>
  <header>...</header>
  <main>
    <header>
      <h1>...</h1>
    </header>
    <section>
      <h2>Заголовок секції зі списком ілюстра-
       цій</h2>
      <ul>
        <li>
          <div>
            <img src="" alt="">
            <p>підпис</p>
          </div>
        </li>
        ...
```

```
        <li>
          <div>
            <img src="" alt="">
            <p>підпис</p>
          </div>
        </li>
      </ul>
    </section>
    <section>
      <h2>Заголовок секції</h2>
      <p>Текст секції, змістовно пов'язаної зі сто-
       рінкою</p>
      <footer> Примітки</footer>
    </section>
  </main>
  <footer>...</footer>
</body>
```

Можна створювати не лише батьківські елементи article з дочірніми елементами section, але й батьківські елементи section з вкладеними елементами article, у яких є один або кілька елементів article. Не всі сторінки повинні бути влаштовані саме так, але це допустимий спосіб вкладання елементів. Наприклад, основна область контенту сторінки містить два блоки із статтями різної тематики. Можна на цьому акцентувати, помістивши кожну статтю однієї тематики всередину елемента section

```
<section>
  <h2>Заголовк секції</h2>
  <p>Текст секції</p>
  <article>
    <h3>...</h3>
    <p>...</p>
  </article>
  <article>
```

```
      <h3>...</h3>
      <p>...</p>
    </article>
  </section>
  <section>
    <h2>Заголовк секції</h2>
    <img src="" alt="">
    <p>Текст секції</p>
    <article>
      <h3>...</h3>
      <p>...</p>
    </article>
    <article>
      <h3>...</h3>
      <p>...</p>
    </article>
  </section>
```

Остання структура може бути взята за основу, наприклад, при розмітці блогів.

Елемент nav

Категорії вмісту: потоковий вміст, секційний вміст, відчутний вміст.
Контекст, у якому цей елемент можна використовувати: де очікується потоковий вміст.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути.

Елемент nav є розділом сторінки з навігаційними посиланнями, який посилається на інші сторінки або частини всередині сторінки, при цьому не обов'язково повинен бути всередині header. На сторінці може бути кілька тегів <nav>.

Якщо вміст елемента nav представляє список елементів, рекомендується використовувати розмітку списку. Не замінює теги <ul> або <ol> він просто їх обрамляє.

Не всі групи посилань на сторінці повинні бути тегом <nav>. Цей елемент призначений головним чином для розділів, що складаються з основних блоків навігації. Зокрема, нижні колонтитули зазвичай мають короткий список посилань на різні сторінки сайту, такі як умови обслуговування, домашня сторінка та сторінка про авторські права. Для таких випадків достатньо одного тегу <footer>. Також можна додавати заголовки всередину елемента.

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

або

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

або зі списком та заголовком

```
<nav>
   <h2> </h2>
   <ul>
     <li><a href=""></a></li>
     <li><a href=""></a></li>
     <li><a href=""></a></li>
   </ul>
 </nav>
```

Елемент aside

Категорії вмісту: потоковий вміст, секційний вміст, відчутний вміст.

Контекст, у якому цей елемент можна використовувати: де очікується потоковий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути.

Елемент aside представляє розділ сторінки, що складається з вмісту, що опосередковано пов'язаний з батьківським секційним елементом і який можна розглядати окремо від нього. Найчастіше елемент позиціонується як бічна колонка (як у книгах) і включає групу елементів: nav, цифрові дані, цитати, рекламні блоки, архівні записи. Не підходить для блоків, які просто позиціоновані осторонь.

Наведемо можливу розмітку блогу з [21]

```
<body>
  <header>
    <h1>Назва блогу</h1>
    <p>Кредо</p>
  </header>
  <aside>
    <nav>
      <h2> Перша група посилань</h2>
      <ul>
        <li><a
        href="https://blog.example.com/">...</a>
      </ul>
    </nav>
    <nav>
      <h2> Друга група посилань</h2>
```

```
    <ol reversed>
      <li><a href="/last-post">...</a>
      <li><a href="/first-post">...</a>
    </ol>
  </nav>
</aside>
<aside>
  <h2>  Зовнішні повідомлення</h2>
   <blockquote
   cite="https://twitter.example.net/t1234567">
     Текст
  </blockquote>
  <blockquote
   cite="https://twitter.example.net/t1234568">
     Текст
  </blockquote>
</aside>
<article>
  <h2>Повідомлення 1</h2>
  <p>Текст</p>
  <footer>
    <p><a href="/last-post" rel=bookmark>...</a>
  </footer>
</article>
<article>
  <h2>Повідомлення 2</h2>
  <p>Текст</p>
  <aside>
    <h3>Додаткові міркування</h3>
    <p>Текст</p>
  </aside>
  <footer>
    <p><a href="/first-post"
    rel="bookmark">...</a></p>
```

```
    </footer>
  </article>
  <footer>
    <nav>
      <a href="/archives">Архіви</a> -
      <a href="/about">Про автора</a> -
      <a href="/copyright">Copyright</a>
    </nav>
  </footer>
</body>
```

Проте aside – це не тільки бічна панель, це може бути й додаткова інформація до section чи article. Наприклад, цитата.

```
<article>
  ...
  <aside>
    <q>Текст цитування</q>
  </aside>
  ...
</article>
```

Елементи h1, h2, h3, h4, h5 та h6

Категорії вмісту: потоковий вміст, секційний вміст, відчутний вміст.
Контекст, у якому цей елемент можна використовувати: де очікується потоковий вміст.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути.

Елементи h1 – h6 представляють заголовки своїх розділів. Ці елементи мають ранг, що визначається числом у їхньому імені. Елемент h1 має найвищий ранг, елемент h6 має найменший ранг, а два елементи з однаковим ім'ям мають однаковий ранг. Слід ви-

користовувати ранг елементів заголовка, щоб створити схему документа.

Пошукові роботи звертають особливу увагу на заголовки вебдокумента, тому їхнє коректне використання надзвичайно важливе.

Теги <h1> – <h6> не повинні використовуватися для розмітки підзаголовків, альтернативних заголовків та слоганів, якщо вони не призначені для заголовка нового розділу або підрозділу.

Розглянемо схему із використанням ієрархії заголовків.

```
<main>
  <h1> Головний заголовок</h1>
  <article>
    <h2> Заголовок розділу</h2>
    <section>
      <h3> Заголовок підрозділу</h3>
    </section>
    <section>
      <h3> Заголовок підрозділу</h3>
    </section>
  </article>
  <article>
    <h2> Заголовок розділу</h2>
    <section>
      <h3>Заголовок підрозділу</h3>
    </section>
    <section>
      <h3>Заголовок підрозділу</h3>
    </section>
  </article>
</main>
```

Хоча за замовчуванням заголовки вищого рівня ієрархії мають більший розмір, проте їх можна завжди змінити за допомогою CSS.

Запитання для повторення

1. Якою інструкцією оголошується тип документа, що відповідає специфікації HTML Living Standard?
2. Яку структуру має базовий HTML-документ?
3. До якої категорії належить елемент html? Його призначення та атрибути?
4. До якої категорії належить елемент head? Його призначення та атрибути?
5. Назвіть елементи категорії метаданих та їх призначення.
6. Чи важливий порядок задання елементів категорії метаданих?
7. Скільки разів можна використовувати елементи meta та link?
8. Де треба зазначати тег <base>? Чому?
9. Де відображається текст, що зазначений у тезі <title>… </title>?
10. Яке призначення елемента link? Наведіть приклади використання цього елемента.
11. Що таке іконка вебсторінки? Де вона відображається? Як її задати?
12. Призначення елемента style? Яка техніка називається Critical CSS?
13. Назвіть елементи, що формують розділи HTML-документа.
14. Скільки разів можуть бути присутні елементи main та h1 на сторінці?
15. Призначення header та footer? Де їх можна використовувати?
16. Яка семантична відмінність між елементами article та section? Наведіть приклади їх використання.
17. Призначення елемента nav? Де може зазначатися тег <nav>?
18. Наведіть приклади розмітки навігації.

19.Скільки разів може бути присутнім елемент nav на сторінці?
20.Яке семантичне призначення елемента aside?
21.Як пов’язанні елементи article і aside, section і aside?
22.Які є елементи заголовків у HTML-документі?
23.Як визначається ієрархія заголовків у HTML-документі?
24.Чому важливо правильно задавати заголовки?
25.Що не доцільно оформлювати як заголовки?

Елементи групування вмісту

Розглянемо HTML-елементи, які також використовують для групування вмісту в середині секцій [8].

Елемент `main`

Категорії контенту: потоковий вміст, відчутний вміст.
Контекст, у якому цей елемент може бути використаний: де очікується вміст потоку, але лише якщо це ієрархічно правильний елемент main.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути.

Елемент main представляє домінуючий вміст документа.

Вміст main елемента має бути унікальним для документа. Вміст, який повторюється у наборі документів або розділах документа, як-от бічні панелі, навігаційні посилання, інформація про авторські права, логотипи сайту та форми пошуку, не слід включати, якщо тільки форма пошуку не є основною функцією сторінки.

Область основного вмісту складається з вмісту, який безпосередньо пов'язаний із основною темою документа чи основною функціональністю програми або розширює її.

Елемент main не допомагає структурувати документ, він є елементом групування. На відміну від таких елементів, як body, заголовки, заголовки, секції та подібні, елемент main не впливає на концепцію DOM щодо структури сторінки. Він є строго інформативним щодо унікальності вмісту.

Документ не повинен містити більше одного тегу <main>, який не має зазначеного атрибута hidden.

Ієрархічно правильним елементом main є той, у якого елементи-предки обмежені елементами html, body, div, без доступної назви form та автономними спеціальними елементами. Кожен елемент main має бути ієрархічно правильним елементом.

Наприклад, у [8] наведені два приклади використання елемента main.

У першому прикладі тег <main> використовується один раз

```
<!DOCTYPE html>
    <html lang="en">
    <title>RPG System 17</title>
    <header>
      <h1>System Eighteen</h1>
    </header>
    <nav>
      <a href="../16/">← System 17</a>
      <a href="../18/">RPXIX →</a>
    </nav>
    <aside>
      <p>This system has no HP mechanic, so there's no
      healing.
    </aside>
    <main>
      <h2>Character creation</h2>
      <p>Attributes (magic, strength, agility) are
      purchased at the cost of one point per level.</p>
      <h2>Rolls</h2>
      <p>Each encounter, roll the dice for all your
      skills. If you roll more than the opponent, you
      win.</p>
    </main>
    <footer>
      <p>Copyright © 2013
    </footer>
    </html>
```

У другому прикладі наведено кілька елементів main, оскільки сценарій використовується для того, щоб навігація працювала без

звернення до сервера та для встановлення атрибуту hidden для тих main, які не є поточними.

```
<!doctype html>
    <html lang=en-US>
    <meta charset=utf-8>
    <title> ... </title>
    <link rel=stylesheet href=spa.css>
    <script src=spa.js async></script>
    <nav>
      <a href= />Home</a>
      <a href=/about>About</a>
      <a href=/contact>Contact</a>
    </nav>
    <main>
      <h1>Home</h1>
      ...
    </main>
    <main hidden>
      <h1>About</h1>
      ...
    </main>
    <main hidden>
      <h1>Contact</h1>
      ...
    </main>
    <footer>Made with ❤ by <a href=https://example.com
/>Example 😗</a>.</footer>
```

Елемент div

Категорії вмісту: потоковий вміст, відчутний вміст.
Контекст, у якому цей елемент можна використовувати: де очікується **потоковий** вміст; як дочірній елемент dl.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути .

Елемент div сам собою нічого не означає. Він репрезентує свої дочірні елементи. Може використовуватися з атрибутами class, lang і title для розмітки семантики, спільної для групи послідовних елементів.

Рекомендується використовувати тег у випадках, коли інший елемент не підходить. З іншого боку, тег може бути корисним для стилістичних цілей або для обгортання кількох елементів усередині розділу, що мають спільні властивості.

У шаблоні нижче тег використовується як спосіб завдання мови двох абзаців відразу, замість того щоб встановлювати мову для двох елементів абзацу окремо.

```
<article lang="uk">
   <h2>Заголовок оповідання</h2>
   <p>Текст абзацу українською</p>
   ...
   <div lang="en-GB">
     <p>Текст абзацу англійською</p>
     <p>Текст абзацу англійською</p>
   </div>
   <p>Текст абзацу українською</p>
   ...
 </article>
```

Елемент p

Категорії вмісту: потоковий вміст, відчутний вміст.

Контекст, у якому цей елемент можна використовувати: де очікується потоковий вміст.

Пропуск тегів: тег може бути пропущений, якщо одразу за ним слідують теги <address>, <article>, <aside>, <blockquote>, <details>, , <dl>, <fieldset>, <figcaption>, <figure>, <footer>, <form>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <header>, <hr>, <main>, <nav>, <ol>, , <pre>, <section>, <table>, ul>, а також коли у батьківському елементі немає вмісту, і батьків-

ський елемент не є <a>, <audio>, <del>, <ins>, <map>, <noscript>, <video>.
Для елемента доступні глобальні атрибути.

Елемент p є абзац. Абзаци – це блоки тексту, фізично відокремлені від суміжних блоків порожніми рядками. У середині тегу <p> можуть міститися лише рядкові теги.

Тег <p> не повинен використовуватися, коли доречний конкретніший елемент.

HTML-списки використовуються для угруповання пов'язаних між собою фрагментів інформації. Існує три види списків:

- маркований список <ul> кожен елемент списку <li> відзначається маркером,
- нумерований список <ol> кожен елемент списку <li> відзначається цифрою,
- список визначень <dl> складається з пар визначення <dt> – <dd>.

Кожен список є контейнером, всередині якого розташовуються елементи списку або пари термін-визначення.

Елементи ul, ol

Категорії вмісту: потоковий вміст. Якщо містить хоча б один елемент li – видимий вміст.
Контекст, у якому цей елемент можна використовувати: де очікується потоковий вміст.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента ul доступні глобальні атрибути.

Елемент ul є список елементів, де порядок елементів не важливий, тобто коли зміна порядку не призведе до істотної зміни змісту документа.

```
<p> Мови, якими оперує інтернет:</p>
<ul>
  <li>HTML – це мова розмітки,</li>
  <li>CSS – мова стилю,</li>
  <li>JavaScript - мова програмування</li>
</ul>
```

Для елемента ol доступні глобальні атрибути, а також атрибути:

- `reversed` – логічний атрибут. Якщо він є, це означає, що список спадний: …, 3, 2, 1. Якщо атрибут опущений, список є зростаючий: 1, 2, 3, ….
- `start` - ціле число, яке визначає порядковий номер першого елемента списку. За замовчуванням дорівнює 1 (якщо відсутній атрибут reversed).
- `type` – використовується для вказівки типу маркера. За замовчуванням використовується десяткова нумерація. Можливі значення:

  1– десяткова нумерація.
  A – нумерація списку в алфавітному порядку, великі літери латинського алфавіту (A, B, C, D).
  a – нумерація списку в алфавітному порядку, малі літери латинського алфавіту (a, b, c, d).
  I – нумерація римськими великими цифрами (I, II, III, IV).
  i – нумерація римськими малими цифрами (i, ii, iii, iv).

Елемент ol є списком елементів, які були спочатку впорядковані таким чином, що зміна порядку змінила б зміст документа. Елементами списку є елементи li.

Елемент li

Категорії контенту: відсутні.

Контекст, у якому цей елемент можна використовувати: всередині елементів ol, ul, menu.

Пропуск тегів: тег , що закриває, може бути опущений, якщо тег <li> відразу слідує за іншим тегом <li>, або якщо більше немає вмісту в елементі, в який вкладено тег.

Для елемента доступні глобальні атрибути. Якщо елемент є дочірнім елементом ol, то атрибут value – необов'язковий.

Атрибут value, якщо він присутній, має бути допустимим цілим числом, що задає порядковий номер списку.

Елемент li є елементом списку. Він може бути контейнером інших списків.

```
<ul>
  <li>item 1,</li>
  <li>item 2,</li>
  <li>item 3:
    <ol>
      <li>elem 1,</li>
      <li>elem 2,</li>
    </ol>
  </li>
  <li>item 4.</li>
</ul>
```

Списки використовуються не лише для групування текстової інформації, але й для іншого потокового вмісту.

```
<section>
  <h2> Заголовок секції</h2>
  <ul>
    <li>
```

```
        <img src="/images/team_card1.jpg" alt=" Ім’я
             Прізвище" width="270">
        <h3>Ім’я Прізвище</h3>
        <p lang="en">
          Product Designer
        </p>
      </li>
      ...
      <li>
        <figure>
          <img src="/images/team_card5.jpg" alt="
             Ім’я Прізвище" width="270">
          <h3>Ім’я Прізвище</h3>
          <p lang="en">
             Frontend Developer
          </p>
        </figure>
      </li>
    </ul>
  </section>
```

Елемент dl

Категорії вмісту: потоковий вміст. Якщо є хоча б одна пара <dt> – <dd>, то видимий вміст.

Контекст, у якому цей елемент можна використовувати: де очікується потоковий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути.

Елемент dl представляє список описів, що складається з нуля або більше груп термін-опис. Термін представлений елементом dt, опис – елементом dd.

Групи термін-опис можуть бути термінами та визначеннями, питаннями та відповідями, категоріями та темами тощо.

Елементи dt, dd

Категорії контенту: відсутні.

Контекст, у якому цей елемент може бути використаний: перед елементами dd або dt всередині елементів dl .

Пропуск тегів: тег </dt> може бути опущений, якщо за тегом <dt> відразу слідує інший тег <dt> або елемент dd; тег </dd> може бути опущений, якщо за тегом <dd> відразу слідує інший тег <dd> або тег <dt>, або якщо в батьківському елементі більше немає вмісту.

Для елементів доступні глобальні атрибути.

Елемент dt представляє термін у списку описів dl.

Елемент dd представляє опис у списку описів dl.

```
<dl>
  <dt>Автор</dt>
  <dd> <cite> Ім'я автора</cite>
  </dd>
  <dt>Переклад</dt>
  <dd><cite> Ім'я перекладача</cite></dd>
</dl>
```

При використанні елемента dl елемент dt не обов'язково представляє визначений термін. Для цього можна використовувати елемент визначення dfn (від англ. definition).

```
 <dl>
  <dt lang="en-us"><dfn>Color</dfn></dt>
  <dt lang="en-gb"><dfn>Colour</dfn></dt>
  <dd>A sensation which (in humans) derives from
      the ability of the fine structure of the eye
      to distinguish three
      differently filtered analyses of a view.
  </dd>
</dl>
```

Елемент pre

Категорії вмісту: потоковий вміст, видимий вміст.
Контекст, у якому цей елемент можна використовувати: де очікується потоковий вміст.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути.

Елемент pre є блоком попередньо відформатованого тексту, в якому структура представлена друкарськими угодами, а не елементами. Використовується, наприклад, для відформатованого подання віршів, кодів тощо

```
<pre>
   Садок вишневий коло хати,
   Хрущі над вишнями гудуть,
   Плугатарі з плугами йдуть,
   Співають ідучи дівчата,
   А матері вечерять ждуть.
    (Т.Шевченко, уривок)
</pre>
```

Для представлення блоку комп'ютерного коду елемент pre може використовуватися з елементом code. Для представлення результату виконання програмного коду або скрипта елемент pre може використовуватися з елементом samp. Аналогічно, елемент kbd може використовуватися в елементі pre , щоб вказати текст, який повинен ввести користувач.

Елемент figure

Категорії контенту: потоковий вміст, кореневий секційний, видимий вміст.
Контекст, у якому цей елемент можна використовувати: де очікується потокове вміст.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути.

Елемент figure є автономним вмістом (необов'язково з підписом), що є самостійним елементом основного потоку. За допомогою елемента figure можна додавати короткі характеристики до ілюстрацій, фотографій, діаграм, фрагментів коду і т.д.

```
<figure>
  <img src="" alt="">
  <figcaption>Підпис до зобра   жен-
  ня</figcaption>
</figure>
```

Коли елемент figure посилається з основного вмісту документа, ідентифікуючи його за заголовком (наприклад, за номером малюнка), це дозволяє легко переміщати такий вміст з основного вмісту, наприклад, у бічну панель або іншу сторінку додатку.

Елемент figcaption

Категорії контенту: відсутні.
Контекст, у якому цей елемент можна використовувати: як нащадок елемента figure.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути.

Елемент figcaption представляє заголовок або легенду для решти батьківського елемента figure.

```
<p> У  <a href="#p"> CSS-коді </a> наведено стилі
   для абзаців .</p>
<figure id="p">
  <figcaption> CSS-код. Приклад стилізації абзаців
  </figcaption>
  <pre><code> p {
    font-family: Lato, sans-serif;
    font-size: 4vw;
    line-height: 1.2;
```

```
        text-align: justify;
        }</code></pre>
    </figure>
    <p> Ці стилі застосовуються до усіх абзаців.</p>
```

Елемент address

Категорії вмісту: потоковий вміст, видимий вміст.
Контекст, у якому цей елемент можна використовувати: де очікується потокове вміст.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути.

Елемент address представляє контактну інформацію про людину чи організацію. Він повинен включати фізичне та/або цифрове розташування/контактну інформацію та засоби ідентифікації особи (осіб) або організації, до якої належить ця інформація. У браузері зазвичай відображається курсивом.
Наприклад,

```
<address>
    <ul>
      <li><a href="https://goo.gl/maps/CPtrU1FHBa2aNy"
      target="_blank" rel="noopener noreferrer
      nofollow">м. Місто, вул. Назва вулиці, буд</a>
      </li>
      <li><a
      href="mailto:info@email.com">info@email.com</a>
      </li>
      <li><a href="tel:+380961111111">+38 096 111 11
      11</a>
      </li>
    </ul>
</address>
```

Елемент blockquote

Категорії контенту: потоковий вміст, кореневий секційний, видимий вміст.

Контекст, у якому цей елемент можна використовувати: де очікується потокове вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути і необов'язковий атрибут cite, що містить посилання на джерело цитати. Браузери можуть дозволяти користувачам переходити за такими посиланнями, але вони насамперед призначені для приватного використання (наприклад, серверними скриптами, що збирають статистику використання цитат сайту), а не читачам.

Елемент blockquote представляє вміст, цитований з іншого джерела, необов'язково з посиланням на джерело цитування, яка вказується у тегах <footer> або <cite>, і, необов'язково, із змінами тексту, такими як анотації та скорочення.

```
<blockquote>
  <p>Тоді лише пізнається цінність часу, коли він втра-
    чений.</p>
  <p>Хто думає про науку, той любить її, а хто її лю-
    бить, той ніколи не перестає вчитися, хоча б зовні
    він і здавався бездіяльним.</p>
  <cite>Афоризми Григорія Сковороди</cite>
</blockquote>
```

Зауважимо, що вміст елемента blockquote виводиться окремим блоком. Якщо потрібно навести цитату чи пряму мову в реченні, то слід використовувати елемент q.

Запитання для повторення

1. Яке призначення елемента main? Яким має бути його вміст?
2. Чи допомагає елемент main структуризувати документ?
3. Скільки елементів main може містити документ?
4. Що означає ієрархічно правильний елемент main?
5. Яке призначення елемента div? Коли варто його застосовувати? Наведіть приклади його використання.
6. Як задаються абзаци у вебдокументі?
7. Які елементи можуть міститись усередині тегу <p>?
8. Які види списків є в HTML? Коли вони використовуються?
9. Чи може бути тег <li> контейнером інших списків?
10. Як можна задавати маркери упорядкованого списку?
11. Як описуються групи термін-опис у списку описів?
12. Яке призначення елемента pre? Коли доцільно його використовувати?
13. Наведіть приклади використання елементів figure та figcaption?
14. Яке призначення елемента address? Як відображається його вміст за замовчуванням у браузері?
15. Для чоого слід використовувати blockquote? Як відображається його вміст за замовчуванням у браузері?

Розмітка тексту

Одне з основних завдань HTML – надавати тексту структуру і зміст, семантику так, щоб браузер зміг відобразити текст коректно.

Розглянемо HTML-елементи, які найчастіше використовують для розмітки текстового вмісту [8].

Елемент span

Категорії вмісту: потоковий вміст, текстовий вміст, видимий вміст.

Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути.

Елемент span сам собою нічого не означає, але може бути корисним при використанні разом з глобальними атрибутами, наприклад, class, lang або dir. Він репрезентує свої дочірні елементи. Доцільно використовувати там, де не можна підібрати прямий семантичний елемент.

```
<p>  Кольори прапору України – <span class="blue-
color"> блакитний</span> і <span class="yellow-
color">жовтий</span>.</p>
```

Елементи em, i, strong, b, u, mark, bdi, bdo

Категорії вмісту: потоковий вміст, текстовий вміст, видимий вміст.

Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути.

Візуально обидва теги та <i> однакові, вони виділяють текст курсивом. Елемент **i** представляє довідкові виділення окремих слів та фрагментів тексту. Зазвичай використовується для коротких формулювань, термінів і правил у навчальних і наукових фахових виданнях, фраз із іноземних мов тощо. Для позначення нових термінів або слів, які пояснюються у подальшому тексті, слід використовувати елемент dfn. Елемент em створює логічні посилення у тексті, підкреслює слова чи словосполучення, уточнюючи зміст фрази в такий спосіб, щоб читачеві було зрозуміло, що хотів висловити автор. Рівень акценту, який має певний уривок тексту, визначається кількістю тегів .

Візуально обидва теги і <b> однакові, вони виділяють текст напівжирним. Проте, тег визначає важливість зазначеного тексту, а тег <b> призначений для виділення тексту без надання йому особливої ваги.

```
<p> Мовою розмітки візуального представлення вебсторінок є мова гіпертекстової розмітки <b>HTML</b>  (<i> Hyper-Text  Markup  Language </i>) </p>
```

Елемент mark є фрагментом тексту в одному документі, позначеним або виділеним для довідкових цілей, таких як:

- привернення уваги до певної частини цитати;
- виділення частини цитованого тексту, яка спочатку не була виділена;
- виділення частин документа, які відповідають будь-якому рядку пошуку.

```
<p>Зверніть увагу на пункти <mark>1.1, 3.2</mark> та <mark>5.4</mark> у тексті документа.</p>
```

У сучасних браузерах текст усередині тегу <mark> підсвічується жовтим тлом, яке можна змінити за допомогою CSS-правил.

Елемент u представляє уривок тексту з невираженою, хоч і явно відображуваною нетекстовою анотацією, такою як позначка тексту як правильного імені в китайському тексті або маркування тексту як помилка. У браузері виділяється підкресленням, у зв'язку з чим рекомендується уникати використання тегу <u> там, де можна сплутати з гіперпосиланням.

Елемент bdi представляє собою фрагмент тексту, який має бути ізольований від решти тексту для двонаправленого форматування тексту. Використовується для текстів, написаних одночасно мовами, що читаються зліва направо та справа наліво.

Елемент bdo дозволяє явно управляти форматуванням спрямованості тексту для дочірніх елементів. Для цього необхідно задати атрибут dir зі значенням ltr, щоб перевизначити напрямок зліва направо, і зі значенням rtl, щоб перевизначити напрямок справа наліво.

Елементи data, time

Категорії вмісту: потоковий вміст, текстовий вміст, видимий вміст.
Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.
Пропуск тегів: жоден з тегів не може бути пропущений.

Для тегу <data> доступні глобальні атрибути, а також обов'язковий атрибут value.

Елемент data представляє свій вміст разом з машинозчитувальною формою цього вмісту в атрибуті value.

Елемент data може бути використаний для таких цілей:

- у поєднанні з мікроформатами або мікроданими для покращення результатів видачі в пошукових системах за рахунок більш точного опису вмісту елемента;

- у поєднанні зі сценаріями на сторінці для зберігання та обробки даних.

У [8] наведено приклад:

```
<script src="sortable.js"></script>
    <table class="sortable">
      <thead>
        <tr>
          <th> Game
          <th> Corporations
          <th> Map Size
      <tbody>
        <tr>
          <td> 1830
          <td> <data value="8">Eight</data>
          <td> <data value="93">19+74 hexes (93
          total)</data>
        <tr>
          <td> 1856
          <td> <data value="11">Eleven</data>
          <td> <data value="99">12+87 hexes (99
          total)</data>
        <tr>
          <td> 1870
          <td> <data value="10">Ten</data>
          <td> <data value="149">4+145 hexes (149
          total)</data>
    </table>
```

Для тегу <time> доступні глобальні атрибути , а також необов'язковий атрибут datetime.

Елемент time представляє свій вміст разом з машинозчитувальною формою цього вмісту в атрибуті datetime. Тип вмісту обмежений різними типами дат, часу, зсуву часових поясів і тривалос-

ті. Вміст елемента <time> має бути валідним значенням атрибуту datetime. Наприклад,

```
<time>2023-11-18</time>
<time>14:54:39</time>
<time>14:54</time>
<time>2011-11-18 14:54</time>
```

Зауваження. Повний перелік валідних значень зазначено у специфікації HTML Living Standart [8]. Проте, якщо атрибут datetime присутній, вмістом елемента time може бути довільний текст.

```
<p> Виставка працюватиме з
   <time class="dtstart" datetime="2023-01-10"> 10
   </time> по <time class="dtend" datetime="2023-01-
   24"> 24 січня 2023 року </time>.
 </p>
```

Елементи code, var, sub та sup, samp, kbd

Категорії вмісту: потоковий вміст, текстовий вміст, видимий вміст.
Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути.

Елемент code є фрагментом комп'ютерного коду, вміст елемента відображається в браузері моноширинним шрифтом. Немає формального способу вказати мову розмітки комп'ютерного коду. Для підсвічування синтаксису за допомогою скрипта можна додати елементу клас із префіксом language-.

```
<pre><code class="language-css">p {
     font-size: 14px;
```

```
    line-height: 1.2;
}</code></pre>
```

Елемент var представляє змінну. Це може бути фактична змінна в математичному виразі або контексті програмування, ідентифікатор, що представляє константу, символ, що ідентифікує фізичну величину, параметр функції або просто термін, що використовується у тексті. Вміст елемента відображається у браузері курсивом.

Елемент sub представляє нижній індекс, а елемент sup – верхній індекс. Ці елементи повинні використовуватися тільки для позначення друкарських умовних позначень з конкретними значеннями, а не для друкарського подання у презентації. Як правило, ці елементи слід використовувати лише в тому випадку, якщо їхня відсутність змінить зміст вмісту.

Тег <sub> може використовуватися всередині тегу <var> для змінних, які мають індекси.

```
<p>  Розглянемо першу вершину ламаної  <var>A</var>
(<var>x<sub>1</sub></var>,<var>y<sub>1</sub></var>).
</p>
```

Елемент samp є прикладом виведення з іншої програми або пристрою. У браузері відображається моноширинним шрифтом.

```
<p>  Результатом програми є <samp > отримане значення
<span id="output"></span> </samp></p>
```

Елемент kbd представляє введення користувача (зазвичай введення з клавіатури, хоча він також може використовуватися для подання іншого введення, такого як голосові команди). У браузері відображається моноширинним шрифтом.

Коли тег <kbd> вкладено в тег <samp>, він представляє дані, що вводяться у тому вигляді, в якому вони були відображені системою.

Коли тег <kbd> містить тег <samp>, він представляє дані, що вводяться, засновані на вихідних даних системи.

Коли тег <kbd> вкладено в інший тег <kbd>, він представляє поточну клавішу або іншу одиницю введення відповідно до механізму введення.

```
<p>Для того щоб змінити мову натисніть комбінацію кла-
віш <kbd><kbd>Ctrl</kbd>+<kbd>Shift</kbd></kbd>.</p>
```

Елементи small, s, cite

Категорії вмісту: потоковий вміст, текстовий вміст, видимий вміст.
Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути.

Елемент small є примітками, набраними дрібним шрифтом. Такий текст зазвичай містить заяви про відмову від відповідальності, застереження, юридичні обмеження чи авторські права. Дрібний шрифт також іноді використовується для атрибуції або задоволення вимог ліцензування. Елемент small призначений для коротких уривків тексту, він не повинен використовуватися для розширених уривків тексту, таких як кілька абзаців, списків чи розділів тексту.

У [8] наведено приклад:

```
<aside>
  <h1>Example Corp</h1>
   <p>This company mostly creates small software and
   Websites.</p>
```

```
  <p>The Example Corp company mission is "To provide
  entertainment and news on a sample basis".</p>
  <p><small>Information obtained from <a
  href="https://example.com/about.html">example.com</a>
  homepage.</small></p>
</aside>
```

Елемент s представляє вміст, який більше не є точним або актуальним. Елемент s не підходить для позначення правок документа. Щоб позначити область тексту як видалену з документа, слід використовувати del.

Елемент cite є посиланням на творчий твір. Він повинен включати назву роботи або ім'я автора (людини, людей або організації) або посилання на URL, або посилання у скороченій формі відповідно до угод, які використовуються для додавання метаданих цитування.

До творчих творів належать: книга, газета, есе, вірш, партитура, пісня, сценарій, фільм, телешоу, гра, скульптура, картина, театральна постановка, п'єса, опера, мюзикл, виставка, звіт про судову справу, комп'ютерна програма, вебсайт, вебсторінка, повідомлення або коментар у блозі, на форумі, твіт, письмову або усну заяву тощо.

```
<p> Прем'єра вистави <cite>Назва вистави</cite>
    українського драматурга <cite>Автор</cite> очіку-
    ється <time datetime="2023-01-24"> 24 січня 2023 ро-
    ку. </time>
</p>
```

Елементи ins, del

Категорії вмісту: потоковий вміст, текстовий вміст, видимий вміст.

Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути, а також необов'язковий атрибут cite, що вказує на джерело цитати або додаткову інформацію про редагування, і datetime, що визначає дату і (необов'язково) час зміни.

Елемент ins є відображенням додавання до документа.

Елемент del є відображенням видалення з документа (перекреслений текст).

```
<aside>
   <ins>
     <p>Додати список справ</p>
   </ins>
 </aside>

 <h3>Список справ:</h3>
 <ul>
   <li><del datetime="2023-02-10">Опрацювати лекцію</del></li>
   <li><del datetime="2023-02-15">Зробити лабораторну роботу на 15.02.2023</del></li>
   <li>Подивитись фільм <cite> Назва</cite></li>
 </ul>
```

Атрибут cite можна використовувати для посилання на документ, який пояснює зміну. Коли цей документ є довгим, наприклад, протоколом зборів, авторам рекомендується включати фрагмент, що вказує на конкретну частину цього документа, в якому обговорюються зміни.

Атрибут datetime може використовуватись для вказівки часу та дати зміни. Якщо присутній, атрибут datetime має бути допустимим рядком дати з необов'язковим часом.

Елемент q

Категорії вмісту: потоковий вміст, текстовий вміст, видимий вміст.

Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути, а також необов'язковий атрибут cite, що містить посилання (URL) на джерело цитати або додаткову інформацію про редагування.

Елемент q представляє деякий вміст, що цитується з іншого джерела. Розділові знаки (такі як лапки), додаються браузером автоматично, тому їх не потрібно ставити в тексті вручну. Посилання на джерело цитати переважно призначені для приватного використання (наприклад, серверними сценаріями, що збирають статистику використання цитат сайту), а не для читачів.

```
<p> Відомий афоризм <cite>Григорія Сковороди:</cite>
    <q cite="https://chz.org.ua/krylati-vyslovy-ta-
    tsytaty-ukrains-koho-heniia-hryhoriia-skovorody/">
    Тоді лише пізнається цінність часу, коли він втраче-
    ний.</q>
</p>
```

Елементи dfn, abbr

Категорії вмісту: потоковий вміст, текстовий вміст, видимий вміст.

Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути, а також необов'язковий атрибут title.

Елемент abbr є абревіатурою, необов'язково з розшифровкою (у браузері зазвичай підкреслюється пунктирною лінією). Розшифровка абревіатури здійснюється за допомогою атрибута title, вона з'являється при наведенні курсору миші на текст.

Елемент dfn використовується для виділення терміна (у деяких браузерах відображається курсив). Визначення (опис) терміна має перебувати у тому самому абзаці p , списку опису dl чи розділі section, як і елемент dfn .

При подальшому використанні одного і того ж терміна в тексті він не потребує обгортки тегом <dfn>.

Якщо тег <dfn> містить атрибут title, значення атрибута title перевизначає вміст тегу <dfn>. При цьому з'являється вміст title при наведенні курсора на елемент

Елемент dfn часто використовується в поєднанні з елементом abbr ля визначення першого входження абревіатури для подальшого використання. Значення атрибута title тегу <abbr> має бути точним значенням цього терміна.

```
<p><dfn><abbr title="Certificate Signing Request"> CSR
  </abbr></dfn> – це ключ, що генерується при запиті на
  видачу SSL-сертифікату.
</p>
```

Для тегу <dfn> можна визначити атрибут id, щоб використовувати його для зв'язування майбутньої аббревіатури з терміном.

```
<p><dfn id="csr"><abbr title="Certificate Signing
  Request" > CSR </abbr></dfn> – це ключ, що генеруєть-
  ся при запиті на видачу SSL-сертифікату.</p>
```

```
<p>Для генерації <a href="#csr"><abbr
  title="Certificate Signing Request">CSR</abbr></a>
   необхідно...
</p>
```

Елемент br

Категорії вмісту: потоковий вміст, текстовий вміст.
Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.
Пропуск тегів: відсутній тег, що закриває.
Для елемента доступні глобальні атрибути.

Елемент br представляє розрив рядка. Теги
 повинні використовуватися тільки для розривів рядків, які фактично є частиною вмісту, як у віршах чи адресах, наприклад:

```
<address>
    вул. Великопільська,  буд.1, <br>
    м. Київ, <br>
    Україна
</address>
```

Тег
 не повинен використовуватися для поділу тематичних груп в абзаці. Наведений нижче приклад є неправильним:

```
<p>
    <label>Ім’я: <input name="name"></label><br>
    <label>Адреса: <input name="address"></label>
</p>
```

Коректним буде запис:

```
<p><label>Ім’я: <input name="name"></label></p>
<p><label>Адреса: <input name="address"></label></p>
```

Елемент wbr

Категорії вмісту: потоковий вміст, текстовий вміст.
Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.
Пропуск тегів: відсутній тег, що закриває.
Для елемента доступні глобальні атрибути.

Елемент wbr представляє можливість розриву довгого рядка, щоб текст можна було переносити в легкочитаному вигляді.

Елемент a

Категорії контенту: потоковий вміст, текстовий вміст, видимий вміст; якщо елемент має атрибут href, то інтерактивний вміст.
Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути, а також атрибути:

- `href` – указує адресу гіперпосилання або якоря.
- `target` – встановлює контекст за замовчуванням для навігації за гіперпосиланнями.
- `download` – визначає, чи завантажувати цільовий ресурс замість переходу на нього.
- `rel` – встановлює зв’язок поточного документа (або підрозділу/теми) з цільовим.
- `rev` – встановлює зворотний зв'язок відношення цільової сторінки з поточним документом (або підрозділом/темою).
- `hreflang` – описує мову цільового ресурсу.
- `type` – додає підказку для типу ресурсу посилання.
- `referrerpolicy` – встановлює політику заголовка HTTP, а саме кількість інформації про вихідну сторінку, з якої здійснено перехід на цільову сторінку.

Якщо тег <a> має атрибут href, то він є гіперпосиланням на зовнішній або якорем – посиланням на ідентифікатор фрагмента, який є значенням атрибута id елемента у зв'язаному документі.

```
<a href="https://goo.gl/maps/CPtrU1FHBa2aNyZL9"
target="_blank" rel="noopener noreferrer nofollow"> м.
Київ, пр-т Лесі Українки</a>
```

Значення target="_blank" означає завантаження сторінки у новій вкладці/вікні браузера, а rel="noopener noreferrer nofollow" – безпечний перехід.

Так, значення

- noopener – указує браузеру перейти до документа за посиланням без надання новому контексту перегляду доступу до поточного документа, забороняючи новій вкладці використовувати властивість window.opener зміни початкової сторінки для використання інформації та поширення шкідливого коду.
- nofollow – указує пошуковим роботам ігнорувати зв'язок між посиланнями.
- noreferrer – інформація про referer не надсилається під час переходу за посиланням.

Атрибут target може набувати значення:

_self – сторінка завантажується у поточне вікно;
_blank – сторінка відкривається у новому вікні браузера;
_parent – сторінка завантажується у фрейм-батьківський елемент;
_top – сторінка завантажується у повне вікно браузера.

Приклад якорю

```
<p><a href="#anchor"> посилання </a></p>
…
<p id="anchor">якір</p>
```

При необхідності зображення можна зробити посиланням, для цього потрібно всього лише помістити тег <img> всередину тегу <a>.

```
<a href="#"> <img src="image.jpg" alt="пояснення до зо-
браження" width="270"> </a>
```

Тег <a> може бути обгорнутий навколо цілих абзаців, списків, таблиць тощо, навіть цілих розділів, за умови, що всередині відсутній інтерактивний контент (наприклад, кнопки або інші посилання).

```
    <aside class="ads">
        <a href="https://">
            <section>
                <img src="./images/image.jpg"
alt="пояснення до зображення" width="270">
                <h3>Підпис до зображення</h3>
                <p>Короткий текст реклами</p>
            </section>
        </a>
    </aside>
```

Запитання для повторення

1. Призначення та використання елемента span?
2. Особливості використання елементів em та i, strong та b.
3. Призначення елемента mark? Як відображається його вміст у браузері за замовчуванням?
4. Особливості використання data і time?
5. Охарактеризуйте елементи code, var, sub та sup, samp, kbd.
6. У якому випадку використовують тег <cite>?
7. Особливості використання та відображення за замовчуванням елемента small?

8. Використання та відображення за замовчуванням вмісту елементів ins, del?
9. Порівняйте використання елементів quoteblock і q.

10. Призначення та відображення за замовчуванням елемента dfn? Наведіть приклади.
11. Елемент abbr? Його відолбраження за замовчуванням у браузері? Який атрибут слід використовувати для розшифровування його вмісту?
12. Особливості використання елементів br та wbr.
13. Для чого використовуються тег <a>?
14. Наведіть приклади організації якорів у вебдокументі.
15. Як організувати безпечне гіперпосилання на зовнішній ресурс?

Убудовані й інтерактивні елементи

Убудований вміст робить сторінки інтерактивними та мультимедійними, а інтерактивний призначеними для взаємодії з користувачем.

У вебсторінку можна вбудовувати відео, аудіо, pdf-документи тощо як із внутрішніх, так і зовнішніх джерел.

Розклянемо елементи, які найчастіше використовують, детальнішу інформацію та решту елементів можна прочитати у [8].

Елемент details

Категорії вмісту: потоковий вміст, кореневий секційний вміст, інтерактивний вміст, видимий вміст.

Контекст, у якому цей елемент можна використовувати: де очікується потоковий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути, а також атрибут open, який робить додаткову інформацію видимою під час завантаження сторінки.

Елемент details представляє віджет розкриття інформації, який використовується, щоб показати або приховати додаткову інформацію у конкретній області інтерфейсу. Не підходить для приміток.

Якщо елемент details має дочірній елемент summary, то перший дочірній елемент summary є написом на елементі. Якщо дочірній елемент summary відсутній, браузер надає свій власний напис у залежності від мови інтерфейсу браузера.

Підходить для створення віджетів акардеон

Елемент summary

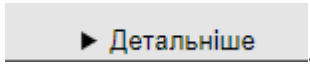
Категорії контенту: відсутня.

Контекст, у якому цей елемент може бути використаний: як перший дочірній елемент елемента details.

Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути.

Елемент summary представляє заголовок або невелике пояснення для решти вмісту батьківського елемента details. Натиснення елемента summary перемикає стан батьківського елемента у відкритий або закритий стан.

```
<section>
  <img src="/" alt="">
  <h3>Підпис</h3>
  <details>
    <summary> Детальніше</summary>
    <p> Текст деталізації</p>
  </details>
</section>
```

Атрибут open тегу <details> додається та видаляється автоматично, коли користувач взаємодіє з елементом управління, тому його можна стилізувати по-різному залежно від стану відкриття або закриття елемента. За замовчуванням представлений у вигляді трикутника 

.

Зображення додаються на вебсторінки за допомогою тегу <img>.

Елемент img

Категорії контенту: потоковий вміст; текстовий вміст; вбудований вміст; елемент, пов'язаний із формою; якщо елемент має атрибут usemap – інтерактивний вміст; видимий вміст.
Контекст, у якому цей елемент може бути використаний: де очікується убудований вміст.
Пропуск тегів: відсутній тег, що закриває.

Для елемента доступні глобальні атрибути, а також атрибути:

- `alt` – додає текст опису зображення, яке виводиться на місці появи зображення до його завантаження або за втрати посилання на нього. З погляду SEO не повинно дублювати назву файлу і має бути складено таким чином, щоб органічно вписатись у контекст;
- `crossorigin` – дозволяє завантажувати зображення <canvas> з ресурсів іншого домену за допомогою CORS-запитів. Зображення, завантажені за допомогою CORS-запитів, можна використовувати повторно. Значення атрибуту:
  - anonymous – запит виконується за допомогою заголовка HTTP, при цьому облікові дані не передаються. Якщо сервер не дає облікові дані серверу, з якого запитується контент, зображення буде зіпсовано і його використання буде обмежено.
  - use-credentials – запит виконується з надсиланням облікових даних.
- `decoding` – підказка для браузерів як декодувати зображення паралельно або разом із рештою вмісту сторінки. Зображення спочатку завантажується із сервера; потім дані зображення декодуються. Декодовані зображення відображаються на сторінці. Декодування великих зображень може блокувати основний потік, перериваючи плавну анімацію та взаємодію користувача. Декодування може бути за замовчуванням, синхронним та асинхронним.
- `height` – задає висоту зображення у px.
- `width` – задає ширину зображення у px.

Для ширини та висоти одиниці вимірювання не задаються. width="500" height=“500”.

Рекомендується указувати розміри зображень за допомогою атрибутів ширини та висоти, навіть якщо вони явно задані в CSS, щоб запобігти зміщенню макета сторінки після завантаження зображення [22].

- `ismap` – указує на те, що зображення є зображенням з інтерактивними областями. При натисканні на карту зображення координати передаються на сервер у вигляді рядка запиту URL-адреси. Атрибут ismap допускається лише у випадку, якщо тег <img> є нащадком тегу <a> із дійсним атрибутом href. Значення відсутнє.
- `loading` – указує на політику завантаження зображень, що знаходяться за межами області перегляду. Завантаження може бути негайним та «лінивим». У багатьох випадках використання значення «lazy» покращує продуктивність.
- `referrerpolicy` – встановлює політику заголовка HTTP про кількість інформації про вихідну сторінку, з якої здійснено перехід на цільову сторінку
- `src` – указує URL-адресу зображення. Адреса зображення може бути абсолютною (http:// site.com/images/photo.jpeg) чи відносною щодо вебдокумента (../images/ photo.jpg) або кореневого каталогу сайту (/images/ photo.jpg).
- `srcset` – створює список зображень для використання в різних ситуаціях. Значенням атрибута є один або декілька рядків URL-адрес, розділених комою або URL-адресою з дескрипторами ширини або дескриптором щільності пікселів, які підказують браузеру зображення якої ширини перебувають за цією адресою.

Наприклад,
srcset="extra-small.jpg 480w, small.jpg 760w medium.jpg 960w large.jpg 1200w, extra-large.jpg 1600w" (дескрипторами ширини)
srcset="image.png 2x" (дескриптором щільності пікселів)

Не можна змішувати дескриптори ширини з дескрипторами щільності пікселів в одному атрибуті srcset. Також є неприпустимим повторення дескрипторів однакової щільності.

- `sizes` – задає ширину зображення між переломними точками дизайну при заданому атрибуті srcset. Значенням атрибута є один або кілька рядків, що містять умову медіа-запиту (має бути пропущено для останнього елемента) та значення розміру джерела. Якщо для ширини не встановлено медіа-запит, це буде значення за замовчуванням. Якщо атрибут srcset відсутній або не містить значення з дескриптором 'w', то атрибут sizes не матиме ніякого ефекту. Відсоткові значення заборонені, натомість використовують значення vw (ширина вікна перегляду).

Наприклад,
sizes="75vw" (значення за замовчуванням)
sizes="(min-width: 768px) 700px"
sizes="(min-width: 768px) 600px, (min-width: 1280px) 80vw, 90vw"

- `usemap` – значення асоціюється зі значенням атрибута id тегу <map> та створює зв'язок між ними. Значення обов'язково має починатися із символу # (usemap="#mymap"). Атрибут не можна використовувати, якщо тег <map> є нащадком <a> або <button>.

```
<img src="/images/ba.jpg" alt="Прізвищенко Іван, біз-
нес-аналітик" loading="lazy" width="300" height="320">
```
або з використанняс атрибутів srcset та sizes
```
<img src="images/ba.jpg" srcset="
           images/ba-mobile.jpg 320w,
           images/ba-wide-mobile.jpg 480w,
           images/ba-tablet.jpg 768w,
           images/ba-desktop.jpg 1024w,
           images/ba-hires.jpg 1280w"
      sizes="(min-width: 768px) 75vw,
           (min-width: 1280px) 80vw,
           90vw"
      alt=" Прізвищенко Іван, бізнес-аналітик">
```

Елемент source

Категорії контенту: відсутні.

Контекст, у якому цей елемент можна використовувати: як дочірній елемент елемента picture перед елементом img. Як дочірній елемент audio або video перед будь-яким потоковим вмістом або елементом track.

Пропуск тегів: відсутній тег, що закриває.

Для елемента доступні глобальні атрибути, а також атрибути:

- `type` – задає MIME-тип вбудованого ресурсу. Якщо браузер не підтримує цей тип, він переходить до наступного елемента source. Його параметр codecs, який визначає типи MIME для елементів audio або video, використовується, щоб точно вказати, як кодується ресурс.

Наприклад:
type="image/svg+xml"
type='video/ogg; codecs="theora, vorbis" '

- `media` – задає список медіа-запитів.

Наприклад:
media="(min-width: 1024px)"

- `src` – указує URL-адресу медіа-ресурсу для елементів audio та video.
- `srcsset` – Створює список зображень для використання в різних ситуаціях.
- `sizes` – Задає ширину зображення між переломними точками дизайну при заданому атрибуті srcset.

Зауваження. Призначення та використання атрибутів `src, srcsset, sizes` аналогічне як і для елемента img.

Елемент source дозволяє вказувати кілька альтернативних джерел зображень елемента img або кілька альтернативних медіа-ресурсів для елементів audio або vidio.

Якщо батьківським елементом є picture, то атрибут srcset є обов'язковим, також можуть бути присутніми атрибути .media і type.

Якщо батьківським елементом є audio або video, то атрибут src є обов'язковим, атрибут type може бути присутнім. Атрибути srcset, sizes і media не повинні бути присутніми.

Елемент picture

Категорії вмісту: потоковий вміст, текстовий вміст, вбудований вміст.
Контекст, у якому цей елемент може бути використаний: де очікується убудований вміст.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути.

Елемент picture разом з елементом source зазвичай використовується для надання багатьох джерел зображення. Це дає браузеру можливість вибору оптимальної версії зображення, залежно від багатьох факторів. Якщо найбільш відповідної версії зображення серед елементів source знайдено не буде, буде відображено файл, указаний у резервному елементі img .

```
<picture>
    <source media="(min-width:768px)"
      srcset="photo768.jpg">
    <source media="(min-width:480px)"
      srcset="photo480.jpg">
    <img src="photo.jpg" alt="Name" width="270">
</picture>
```

Зауваження. Хоча атрибути media і sizes містять умови медіа, вони не використовуються однаково. Атрибут sizes спеціально використовується для створення колекції розмірів макета з шириною макета, указаною після кожної умови.

Атрибут media містить лише медіа-умову, і лише якщо вона оцінюється як true, <source> активує елемент, до якого він приєднаний.

Елементи video, audio

Категорії контенту: потоковий вміст; текстовий вміст; вбудований вміст; видимий вміст; якщо є атрибут controls, то інтерактивний вміст.

Контекст, у якому цей елемент може бути використаний: де очікується убудований вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елементів доступні глобальні атрибути, а також атрибути:

- `src` – указує URL-адресу медіафайлу. Необов'язковий атрибут, натомість можна використовувати елемент <source> усередині <video>/<audio>.
- `crossorigin` – дозволяє завантажувати відео/аудіо <canvas> з ресурсів іншого домену за допомогою CORS-запитів. Відео/аудіо файли, завантажені за допомогою CORS-запитів, можна використовувати повторно.
- `playsinline` (лише video) – логічний атрибут, який указує на те, що відео має відтворюватися в області відтворення елемента, а не у повноекранному режимі (для Android- та iOS-пристроїв).
- `poster` (лише video) – указує URL-адресу зображення, яке відображатиметься перед відтворенням відео.
- `preload` – підказує браузеру, який контент завантажити перед відтворенням відео/аудіо для покращення взаємодії з користувачем. Значення за замовчуванням відрізняється для кожного браузера.
- `autoplay` – логічний атрибут, який указує на те, що відео/audio файл може запускатися автоматично під час завантаження сторінки.

- `loop` – логічний атрибут, який указує, що відео/аудіо слід циклічно повторювати.
- `muted` – логічний атрибут, який указує, що звук відео/аудіо має бути вимкнено за замовчуванням.
- `controls` – відображає елементи керування, які дозволяють користувачеві керувати відтворенням відео/аудіо, включаючи гучність, пошук та призупинення/відновлення відтворення.
- `width` (лише video) – задає ширину області відображення відео в px.
- `height` (лише video) – визначає висоту області відображення відео в px.

Елемент video використовується для відтворення відео або фільмів, а також аудіофайлів із субтитрами. Елемент audio призначений для відтворення звуку або аудіопотоку. Усередині тегів <video> й <audio> може бути наданий текстовий вміст для старих браузерів у разі неможливості відтворення медіаконтенту. Порядок подання ресурсів важливий.

```
<video controls poster="example.jpg" width="500"
     height="500"  >
    <source type="video/webm" src="example.webm">
    <source type="video/mp4" src="example.mp4">
    <a href="example.webm" download> Скачайте відео</a>
 </video>

<audio controls muted>
    <source src="music.ogg" type="audio/ogg;
     codecs=vorbis">
    <source src="music.mp3" type="audio/mpeg">
</audio>
```

Приклад автоматичного програвання відео.

```
<video controls  autoplay loop width="500"
    height="500">
   <source type="video/mp4" src="example.mp4">
</video>
```

Зауваження. Відображення елементів video/audio можуть відрізнятися у різних браузерах.

Елемент track

Категорії контенту: відсутні.

Контекст, у якому цей елемент може бути використаний: як дочірній елемент елементів video і audio перед будь-яким потоковим вмістом.

Пропуск тегів: відсутній тег, що закриває.

Для елемента доступні глобальні атрибути, а також атрибути:

- `kind` – указує, який тип текстової доріжки потрібно використовувати.
- `chapters` – виводить заголовки, призначені для навігації по медіаресурсу. Відображається як інтерактивний (потенційно вкладений) список в інтерфейсі браузера.
- `metadata` – виводить дані, призначені для використання скриптами. Не відображаються для користувачів.
- `srclang` – указує мову текстової доріжки. Якщо для атрибута kind встановлено значення subtitles, цей атрибут має бути визначеним.
- `label` – указує заголовок текстової доріжки, який використовується браузером під час виведення списку доступних текстових доріжок.
- `default` – указує, що ця звукова доріжка є за замовчуванням.

Елемент track дозволяє чітко зазначати зовнішні текстові ресурси для медіа елементів.

```
<video src="name.webm">
        <track kind=subtitles src=name.en.vtt
srclang="en" label="English">
        <track kind=captions src=name.en.hoh.vtt
srclang="en" label="English for the Hard of Hearing">
        <track kind=subtitles src=name.uk.vtt
srclang="uk" lang="uk" label="Ukrainian">
    </video>
```

Елемент iframe

Категорії вмісту: потоковий вміст, текстовий вміст, вбудований вміст, інтерактивний вміст, видимий вміст.

Контекст, у якому цей елемент може бути використаний: де очікується убудований вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути, а також атрибути: `src, srcdoc, name, sandbox, allow, width, height, referrerpolicy, loading.`

Елемент iframe використовується для убудовування іншого HTML-документа, відеоконтенту, карт тощо в поточний, при цьому повністю ізольований від JavaScript і CSS батьківського елемента. За замовчуванням елемент має рамку.

Елемент iframe має як переваги, так і ризики використання. Серед переваг зазначимо:

- швидке підвантаження відеоматеріалів, карт, презентацій та іншого вмісту;
- фрейми дозволяють показати відвідувачу одночасно кілька сторінок, які абсолютно самостійні та незалежні;
- можна перемикати екрани, не перезавантажуючи сторінку;
- можна розбити структуру вебресурсу по блоках, що дозволить редагувати лише окремі блоки.

Проте, використовувати елемент iframe треба з відповідальністю за сторонній контент. Унього можна вбудовувати вміст лише з перевірених та надійних джерел, оскільки коли використовують iframe, то переважно мають справу з контентом, отриманим від третьої сторони, яку не можливо контролювати. Таким чином, підвищується ризик потенційної вразливості у додатку. Щоб зменшити ризики, потрібно правильно налаштувати атрибути. Крім того, потрібно враховувати, що контент, який підвантажується, не буде проіндексований, як частина вебдокументу.

Приклад убудованого відеоконтенту з Youtube

```
<iframe width="693" height="390"
src="https://www.youtube.com/embed/heycg7D5VZw"
frameborder="0" allow="accelerometer; autoplay;
encrypted-media; gyroscope; picture-in-picture"
allowfullscreen></iframe>
```

Приклад убудованої карти

```
<iframe id="inlineFrameExample" title="Inline Frame
Example" width="300" height="200"

src="https://www.openstreetmap.org/export/embed.html?bb
ox=-
0.004017949104309083%2C51.47612752641776%2C0.0003057718
2769775396%2C51.478569861898606&layer=mapnik">
</iframe>
```

Елемент map

Категорії вмісту: потоковий вміст, текстовий вміст, вбудований вміст.

Контекст, у якому цей елемент може бути використаний: де очікується убудований вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути, а також атрибут name, що визначає ім'я зображення-картки для посилання з атрибута usemap.

Елемент map у поєднанні з елементом img та будь-якими нащадками елемента area визначає зображення-карту. Елемент репрезентує свої дочірні елементи, які задані координатами. Тобто зображення представляє собою карту (map), яка поділена на області (area). Область визначається фігурою та координатати. Якщо вона має гіперпосилання на певний вміст, то, при натисненні на область карти-зображення, можна перейти на потрібний вміст.

Елемент area

Категорії вмісту: потоковий вміст, текстовий вміст.

Контекст, у якому цей елемент може бути використаний: там, де очікується текстовий вміст, але тільки як предок елемента map або template.

Пропуск тегів: відсутній тег, що закриває.

Для елемента доступні глобальні атрибути, а також атрибути:

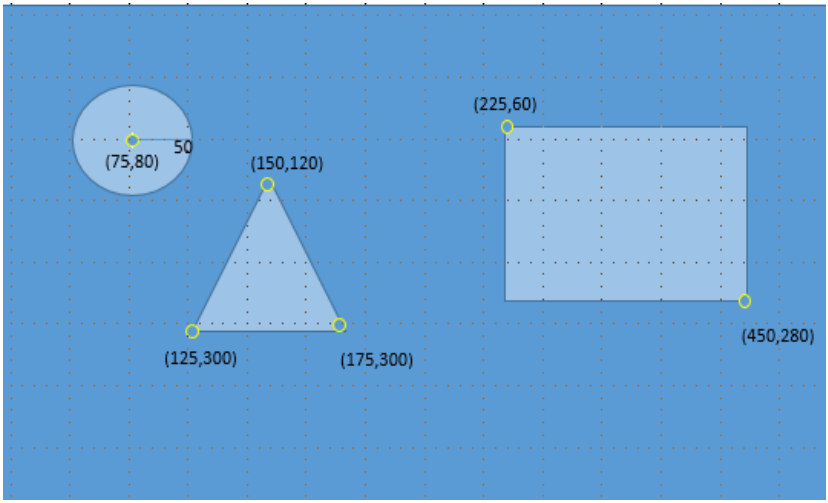
- `alt` – задає альтернативний текст для картки, коли зображення недоступні.
- `coords` – задає координати форми, що описується атрибутом shape. Координати задаються за допомогою цілих чисел і поділяються комами:
  - для кола: координати центру та радіус кола;
  - для прямокутника: координати верхнього лівого та правого нижнього кутів;
  - для багатокутника: координати вершин багатокутника у потрібному порядку, також рекомендується вказувати останні координати, рівні першим, для логічного завершення фігури.
- `download` – дозволяє завантажувати ресурс замість переходу до нього.
- `href` – указує URL-адресу для посилання

- `hreflang` – визначає мову пов'язаного веб-документу. Використовується лише якщо заданий атрибут href.
- `ping` – задає URL-адреси ресурсів, які отримають сповіщення, якщо користувач перейде за посиланням.
- `rel` – встановлює зв'язок поточного документа (або підрозділу/теми) із цільовим ресурсом. Може містити більше значення.
- `shape` – визначає форму області, яка створюється на зображенні-карті:
  - rect: активна область прямокутної форми;
  - circle: активна область у формі кола;
  - poly: активна область у формі багатокутника;
  - default: активна область займає усю площу зображення.
- `target` – вказує, куди буде завантажено документ під час переходу за посиланням.
- `type` – указує MIME тип зв'язаного ресурсу.
- `referrerpolicy` – встановлює політику заголовку HTTP, а саме кількість інформації про вихідну сторінку, з якої здійснено перехід на цільову сторінку.

Елемент area є або гіперпосилання (якщо заданий атрибут href) з деяким текстом і відповідною областю на зображенні-карті, або неактивною областю. Якщо елемент area є гіперпосиланням, то атрибут alt обовєязковий. Якщо елемент area не має атрибута href, то область карти не може бути вибрана, тому атрибут alt повинен бути опущеним.

Атрибут shape визначає coords області карти. Якщо опустити ці атрибути, то область на карті займатиме усе зображення.

Атрибути target, download, rel, hreflang, type і referrerpolicy вказуються тільки тоді, якщо заданий атрибут href.



Мал. 5 Ілюстративне зображення карти на зображенні

```
<aside>
    <img src="photo.jpg" alt="description" width="500"
     height="500" usemap="#anchor">
    <map name="anchor">
        <area shape="circle" coords="75,80,50"
     href="https://" alt="area1" target="_blank">
        <area shape="poly"
     coords="125,300,150,120,175,300,125,300"
     href="https://" alt="area2" target="_blank">
        <area shape="rect" coords="225,60,450,280"
     href="https://" alt="area3" target="_blank">
    </map>
</aside>
```

Запитання для повторення

1. Для чого призначені вбудовані та інтерактивні елементи HTML?
2. Призначення елемента details? Яка його поведінка за замовчуванням у браузері?
3. Які елементи є дочірніми details?
4. Як можна подати зображення на вебсторінку?
5. Які атрибути елемента img є обовєязковими?
6. Чому важливо задавати розміри зображення у HTML-коді?
7. Поясність застосування атрибутів src, srcset та sizes у вбудованих елементах, де вони доступні.
8. Чому зазвичай використовують декілька дочірніх елементів sourse для одного вбудованого елемента?
9. Призначення елементів video та audio? Їхні основні атрибути?
10. Для чого використовується елемент track?
11. Які переваги та ризики використання має елемент iframe?
12. Наведіть приклади доцільності використання елемент iframe.
13. Як створити карту-зображення за допомогою HTML-елементів?
14. Які фігури можна задавати на карті-зображенні як інтерактивні області?
15. Наведіть приклад коду карти-зображення.

Таблиці

HTML-таблиці упорядковують та виводять на екран дані за допомогою рядків або стовпців. Таблиці використовують за своїм призначенням, коли у її комірки треба вивести інформацію. Комірки таблиць можуть містити будь-які HTML-елементи, такі як заголовки, списки, текст, зображення, елементи форм, а також інші таблиці. Кожній таблиці можна додати пов'язаний із нею заголовок.

Якщо таблиця використовуватиметься для розмітки вебсторінки, як-то друкованої форми документа, конверта тощо, то її необхідно позначити атрибутом role="presentation", щоб браузер правильно представляв таблицю для програм зчитування з екрана.

Розглянемо основні частини таблиці.

Елемент table

Категорії вмісту: потоковий вміст, видимий вміст.

Контекст, у якому цей елемент можна використовувати: де очікується потоковий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути та атрибут border, який явно вказує, що тег <table> представляє табличні дані і не використовується для розмітки сторінки. Значення атрибута має бути порожнім рядко, або цілим значенням, наприклад, «1».

Елемент caption

Категорії контенту: відсутні.

Контекст, у якому цей елемент може бути використаний: як перший дочірній елемент елемента table.

Пропуск тегів: тег, що закриває, може бути опущений.

Для елемента доступні глобальні атрибути.

Елемент caption надає заголовок таблиці. Якщо елемент <table> є єдиним вмістом для елемента figure з дочірнім figcaption, елемент caption слід опустити на користь figcaption.

Елемент colgroup

Категорії контенту: відсутні.

Контекст, у якому цей елемент може бути використаний: як дочірній елемент елемента table після елемента caption і перед будь-якими елементами thead, tbody, tfoot і tr.

Пропуск тегів: тег </colgroup> може бути опущений, якщо за елементом colgroup не слідує пробіл або коментар. Тег <colgroup>, що відкриває, може бути опущений, якщо першим всередині нього є тег <col>, і йому безпосередньо не передує інший елемент colgroup, без тегу, що закриває елемент.

Для елемента доступні глобальні атрибути, а також атрибут span, що вказує кількість стовпців, що об'єднуються елементом. Використовується у розмітці для візуального форматування групи стовпців таблиці.

Якщо елемент colgroup не містить елементів col, то для нього можна вказати атрибут span із натуральним.

Елемент col

Категорії контенту: відсутні.

Контекст, у якому цей елемент можна використовувати: як дочірній елемент елемента colgroup, для якого не заданий атрибут span.

Пропуск тегів: відсутній тег, що закриває.

Для елемента доступні глобальні атрибути, а також атрибут span, що вказує кількість стовпців, що об'єднуються елементом. Використовується у розмітці для візуального форматування стовпців таблиці.

Елемент tbody

Категорії контенту: відсутні.

Контекст, у якому цей елемент може бути використаний: як дочірній елемент елемента table після елементів caption, colgroup і thead, але якщо немає елементів tr, які є дочірніми елементами елемента table.

Пропуск тегів: початковий тег <tbody> може бути опущений, якщо першим дочірнім його тегом є тег <tr>, і якщо йому безпосередньо не передують теги </tbody>, </thead> або </tfoot>. Тег <tbody> не можна опустити, якщо елемент tbody порожній. Тег </tbody> елемента tbody може бути опущений, якщо за елементом відразу слідує елемент tbody (внутрішньої таблиці) або tfoot, або якщо в батьківському елементі більше немає вмісту.
Для елемента доступні глобальні атрибути.

Елемент tbody представляє блок рядків, що є основнми даними таблиці.

Елемент thead

Категорії контенту: відсутні.
Контекст, у якому цей елемент може бути використаний: як дочірній елемент елемента table після елементів caption і colgroup перед елементами tbody, tfoot і tr, але тільки якщо немає інших елементів thead.
Пропуск тегів: тег </thead> може бути опущений, якщо елемент thead слідує одразу за елементами tbody або tfoot.
Для елемента доступні глобальні атрибути.

Елемент thead представляє блок рядків, що складається із заголовків стовпців таблиці.

Елемент tfoot

Категорії контенту: відсутні.
Контекст, у якому цей елемент може бути використаний: як дочірній елемент у table після будь-яких елементів caption, colgroup, thead, tbody і tr, але тільки якщо немає інших елементів tfoot.
Пропуск тегів: тег </tfoot> може бути опущений, якщо більше немає вмісту в батьківському елементі.
Для елемента доступні глобальні атрибути.

Елемент tfoot представляє блок рядків, що, складається з нижніх колонтитулів.

Елемент tr

Категорії контенту: відсутні.

Контекст, у якому цей елемент можна використовувати: як дочірній елемент елемента thead, tbody, tfoot. Також, як дочірній елемент table після будь-яких елементів caption, colgroup і thead, але тільки якщо немає елементів tbody .

Пропуск тегів: тег</tr> може бути опущений, якщо за елементом tr відразу слідує інший елемент tr або, якщо більше немає вмісту в батьківському елементі.

Для елемента доступні глобальні атрибути.

Елемент tr пердставляє рядок комірок у таблиці.

Елемент td

Категорії контенту: відсутні.

Контекст, у якому цей елемент можна використовувати: як дочірній елемент елемента tr.

Пропуск тегів: тег <td> може бути опущений, якщо за елементом відразу ж слідує інший елемент td або th , або якщо в батьківському елементі більше немає вмісту.

Для елемента доступні глобальні атрибути, а також атрибути:

- `colspan` – задає кількість стовпців, що об'єднуються.
- `rowspan` – задає кількість рядків, що об'єднуються
- `headers` – задає список рядків, розділених пробілами, кожен з яких відповідає атрибуту id тегів <th>, пов'язаних із цим елементом.

Елемент td є коміркою даних у таблиці.

Елемент th

Категорії контенту: відсутні.

Контекст, у якому цей елемент можна використовувати: як дочірній елемент елемента tr.

Пропуск тегів: тег </th> може бути опущений, якщо за елементом th відразу слідує елемент td або th, або якщо в батьківському елементі більше немає вмісту.

Для елемента доступні глобальні атрибути, а також атрибути: `colspan, rowspan, headers` із призначенням як у td. А також атрибути:

- `scope` – визначає комірки, до яких відноситься заголовок, визначений в елементі th. Дозволені значення:
  - row – заголовок відноситься до всіх комірок рядка, до якого він належить.
  - col – заголовок відноситься до всіх комірок стовпця, якому він належить.
  - rowgroup – заголовок належить групі рядків і відноситься до всіх її комірок. Ці осередки можуть бути розміщені праворуч або ліворуч від заголовка, залежно від значення атрибута dir тегу <table>.
  - colgroup – заголовок належить colgroup і відноситься до всіх її комірок.
- `abbr` Задає альтернативний скорочений опис вмісту комірки. Програми для читання промови можуть представляти цей опис перед самим контентом.

Елемент th визначає комірку як заголовок групи комірок таблиці.

Чек покупця

№	Продукт	Ціна	Кількість	Вартість
1	Хліб	20,00	2	40,00
2	Сир	60,00	1	60,00
3	Молоко	30,00	2	60,00
Всього				160,00
Дякуємо за покупку!				

Мал. 6 Дані покупки, подані таблично

```
<table  border="1" width="377">
    <caption>Чек покупця</caption>
```

```
<thead>
    <tr>
        <th>№</th>
        <th>Продукт</th>
        <th>Ціна</th>
        <th>Кількість</th>
        <th>Вартість</th>
    </tr>
</thead>
<tbody>
    <tr>
        <td>1</td>
        <td>Хліб</td>
        <td>20,00</td>
        <td>2</td>
        <td>40,00</td>
    </tr>
    <tr>
        <td>2</td>
        <td>Сир</td>
        <td>60,00</td>
        <td>1</td>
        <td>60,00</td>
    </tr>
    <tr>
        <td>3</td>
        <td>Молоко</td>
        <td>30,00</td>
        <td>2</td>
        <td>60,00</td>
    </tr>
    <tr>
        <td colspan="4">Всього</td>
        <td>160,00</td>
```

```
        </tr>
    </tbody>
    <tfoot>
        <tr>
            <td colspan="5">Дякуємо за покупку!</td>
        </tr>
    </tfoot>
</table>
```

Дані, подані таблично, можна використовуватикомбінувати з інтерактивними елементами та елементами угрупування. У [8] наведені ілюстративні приклади:

- використання details для деталізації заголовку таблиці

```
<table>
    <caption>
       <strong>Characteristics with positive and nega-
tive sides.</strong>
       <details>
          <summary>Help</summary>
          <p>Characteristics are given in the second
column, with the negative side in the left column and
the positive side in the right column.</p>
        </details>
    </caption>
    <thead>
        <tr>
            <th id="n"> Negative
            <th> Characteristic
            <th> Positive
    <tbody>
        <tr>
            <td headers="n r1"> Sad
            <th id="r1"> Mood
```

```
            <td> Happy
        <tr>
            <td headers="n r2"> Failing
            <th id="r2"> Grade
            <td> Passing
</table>
```

- table як пояснення для елемента figure

```
<figure>
    <figcaption>Characteristics with positive and nega-
    tive sides</figcaption>
    <p>Characteristics are given in the second column,
    with the negative side in the left column and the
    positive side in the right column.</p>
    <table>
        <thead>
            <tr>
                <th id="n"> Negative
                <th> Characteristic
                <th> Positive
        <tbody>
            <tr>
                <td headers="n r1"> Sad
                <th id="r1"> Mood
                <td> Happy
            <tr>
                <td headers="n r2"> Failing
                <th id="r2"> Grade
                <td> Passing
    </table>
</figure>
```

У прикладах з [8] наведені зразки правильного не використання тегів, що закривають вміст відповідних елементів thead, tbody, th, td, tr. Вибір за верстальником використовувати повний синтаксис подвійних тегів, як у першому прикладі, чи скорочений., як у двох наступних.

Запитання для повторення

1. Призначення та використання елемента table у HTML-розмітці.
2. Назвіть блоки, з яких може складатись таблиця та якими елементами ці блоки створюються?
3. За що відповідають елементи th, tr, td?
4. За що відповідають елементи thead, tbody, tfoot?
5. Призначення елементів colgroup і col?
6. Як об’єднати декілька комірок по рядках чи стовпцях?
7. Як зв’язати елементи заголовків та комірок?
8. Де можна використовувати таблиці?
9. Чи може таблиця міститись у іншій таблиці?
10. Наведіть приклади використання таблиць у розмітці.

Форми

Вебформи – це набори текстових полів для введеня даних, списків вибору, прапорців та перемикачів, кнопок та інших елементів керування, за допомогою яких відвідувачі вебсторінки можуть надавати необхідну інформацію для відправки на сервер для подальшої обробки, отримувати результати пошуку, обчислень, відбору тощо.

Форми на вебресурсах засосовуються часто. Завдяки ним створюють облікові записи, форми автентифікації, замовлення товарів у інтернет-магазинах, здійснення фінансових транзакцій та багато іншого. Однією з найпростіших форм є текстове поле пошукової системи.

Створюють форми на вебсторінках вебресурсів за допомогою елементів керування HTML. Сучасний HTML надає можливість створювати форми різноманітної складності та навіть здійснювати валідацію на стороні клієнта. Валідація на стороні клієнта – це первинна перевірка введених даних, яка суттєво покращує зручність взаємодії з інтерфейсом, оскільки виявлення введення некоректних або незаповнення обов'язкових даних на стороні клієнта дозволяє користувачеві негайно їх виправити. Якщо ж перевірку здійснювати лише на сервері, процес заповнення може бути більш трудомістким, оскільки вимагає повторення тих самих дій відправлення даних на сервер для отримання зворотної відповіді з повідомленням про те, що потрібно виправити. Проте варто зазначити, що валідація на стороні клієнта не є достатньою, оскільки її можна обійти. Тому будь-які дані, що надсилаються через форму, необхідно додатково перевіряти на безпеку й та на стороні сервера.

Є різні типи валідації на стороні клієнта, зокрема, вбудована HTML5-валідація та JavaScript-валідація. Вбудована валідація форм використовує функціонал валідації HTML5. HTML5-валідація зазвичай не вимагає великої кількості JavaScript-коду і демонструє

кращу продуктивність, але не настільки гнучка, як валідація за допомогою JavaScript.

Здійснюється убудована валідація за допомогою атрибутів елементів керування на введення символів відповідно до типу даних, заповнення обов'язкових полів тощо. Стилізація форм та повідомлень валідації відбувається за допомогою CSS.

Основою форми є елемент form, який є контейнером для усіх її елементів керування форми (полів). В одну форму потрібно включати поля, об'єднані логічно. На сторінці може бути декілька форм. Проте слід пам'ятати, що не варто вкладати форму у форму, оскільки таке розміщення може призвести до непередбачуваної поведінки форм залежно від браузера.

Використання правильної структури при створенні HTML форм допоможе гарантувати їхню зручність і доступність користувачеві та програмам зчитування.

Елемент form

Категорії вмісту: потоковий вміст, видимий вміст.

Контекст, у якому цей елемент можна використовувати: де очікується потоковий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути, а також атрибути:

- `accept-charset` – визначає кодування символів, яке потрібно використовувати для надсилання форми.
- `action` – указує URL-адресу обробника форми. Коли форма відправляється, дані у формі перетворюються на структуру відповідно до зазначеного типу кодування enctype, а потім відправляються у місце, вказане action із використанням методу відправки даних method. Якщо атрибут не вказаний, дані форми відправляються на ту саму сторінку, на якій розміщується форма. Запис <form action="#"> вважається застарілим.

- `autocomplete` – указує, чи може значення елементів input автоматично заповнюватися браузером. Якщо значення:
  - off, то користувач повинен щоразу явно вводити значення у поля форми;
  - on, то браузер може автоматично доповнювати значення на підставі попередньо введених користувачем значень або за вказівками, визначеними браузерами.
- `enctype` – указує MIME-тип даних форми для надсилання на сервер лише у випадку method="post". Це значення може бути перевизначено атрибутом formenctype в елементах button або input з типом type="image" та type="submit".
- `method` – указує HTTP-метод для надсилання форми. Це значення перевизначається атрибутом formmethod в елементах button або input з типом type="image" та type="submit". Значення:
  - post: дані форми включаються у тіло HTTP-запиту. Метод є більш надійним та безпечним, ніж get і не має обмежень на розмір даних. Використовується для відправки конфіденційних даних;
  - get: дані форми (пара ім'я-значення) додаються в URL-адресу за допомогою роздільника ? і відправляються на сервер. Цей спосіб має обмеження на розмір даних, що відправляються, і не підходить для відправлення конфіденційної інформації. Метод є методом за замовчуванням і є найкращим варіантом для відправки незахищених даних, наприклад, рядків запитів пошуку інформації;
  - dialog: використовується для закриття діалогового вікна, яке містить форму.
- `novalidate` – логічний атрибут, який вказує на те, що форма не повинна перевірятися під час відправлення. Якщо цей атрибут не встановлений, його можна перевизначити атрибутом formnovalidate в елементах button або input з типом type="image" та type="submit".

- `target` – указує, у якому вікні виводити результат після надсилання форми. Це значення перевизначається атрибутом в formtarget в елементах button або input з типом type="image" та type="submit".
- `rel` – визначає зв'язок між пов'язаним ресурсом та поточним документом.

Елемент fieldset

Категорії контенту: потоковий вміст; кореневий секційний вміст; елемент, пов'язаний із формою; видимий вміст.
Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути, а також атрибути:

- `disabled` – логічний атрибут, відключає від редагування та взаємодії усі дочірні елементи керування форми, за винятком тих, що містяться усередині елемента legend
- `form` – указує на зв'язану форму. Значенням атрибута є id елемента form у тому самому документі.

Елемент fieldset групує елементи керування форми та написи label до них. Якщо до групи потрібно додати заголовок, то використовують елемент legend. За замовчуванням у браузері відображається рамкою навколо згрупованих елементів за шириною вікна браузера. Елемент fieldset можна використовувати для поділу форми. Зокрема, набір перемикачів, рекомендується розміщувати усередині <fieldset>. Елемент fieldset може розташовуватись і поза тегами <form>…</form>. Прив’язка елемента до відповідної форми відбувається завдяки зв’язці, наприклад, <form id=”formID”>, <fieldset form=” formID”>.

```
<form  id="formID">
   <fieldset>...</fieldset>
    ...
   <fieldset>...</fieldset>
</form>
  ...
<fieldset form="formID">...</fieldset>
```

Елемент legend

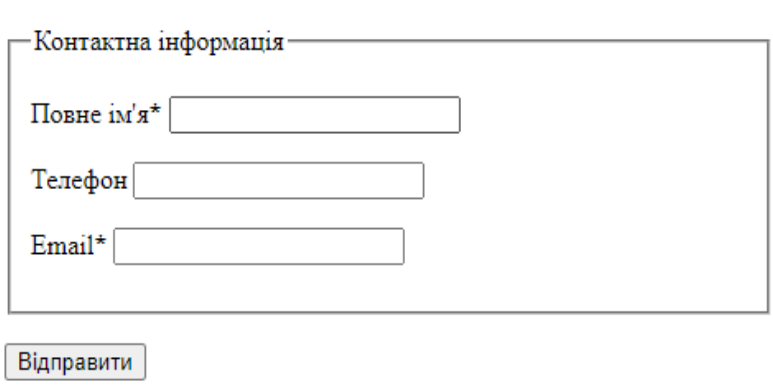
Категорії контенту: відсутні.

Контекст, у якому цей елемент може бути використаний; як перший дочірній елемент елемента fieldset.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути.

Елемент legend є заголовок для решти вмісту батьківського елемента fieldset. За замовчуванням текст заголовку вставляється у рамку.



Мал. 7 Форма. Згруповані дані

```
<form action="http://" method="post">
    <fieldset>
      <legend>Контактна інформація</legend>
      <p>
```

```
        <label for="name"> Повне ім'я<strong>*</strong>
        </label>
        <input id="name" type="text" required>
      </p>
      <p>
        <label for="telephone">Телефон</label>
        <input id="telephone" type="tel">
      </p>
      <p>
        <label for="email"> Email<strong>*</strong>
        </label>
        <input id="email" type="email" required>
      </p>
    </fieldset>
    <p>
        <button type="submit">Відправити</button>
    </p>
</form>
```

Для використання пристроїв зчитування вебсторінок цей код слід доповнити

```
<form action="http://" method="post">
   <fieldset>
      <legend>Контактна інформація</legend>
      <p>
         <label for="name"> Повне ім'я
         <strong><span aria-label="required">*</span>
        </strong>
         </label>
         <input id="name" type="text" required>
      </p>
      <p>
         <label for="telephone">Телефон</label>
         <input id="telephone" type="tel">
```

```
    </p>
    <p>
       <label for="email">Email
       <strong><span aria-label="required">*</span>
       </strong>
       </label>
       <input id="email" type="email" required>
    </p>
  </fieldset>

    <p><button type="submit">Відправити</button></p>
</form>
```

Елемент label

Категорії вмісту: потоковий вміст, текстовий вміст, інтерактивний вміст, видимий вміст.

Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути, а також атрибут for, який визначає, до якого поля форми прив'язаний цей елемент. Значення атрибута має містити ідентифікатор поля форми.

Елемент label подає напис до зв’язаного елемента керування форми. Елемент може бути пов'язаний з конкретним полем форми за допомогою атрибута for або шляхом приміщення елемента управління форми всередину label.

```
<p>
    <label for="name">Укажіть повне ім'я:</label>
    <input id="name">
</p>
```

або

```
<p>
    <label>Укажіть повне ім'я: <input ></label>
</p>
```

Використання напису дає змогу сфокусувати елемент керування не лише натиснувши на ньому, але й на його написі-підказці. А це важливо особливо для людей з обмеженими можливостями.

Одним із найпоширенішим елементом форми є елемент input. У залежності від значення атрибуту type набуває різної форми. Цей лемент має багато різних атрибутів. Деяні з них допустимі при наявності інших. Не всі браузери їх усі поки що підтримують. Тому, перед застосуванням потрібно читати довідкову інформацію, наприклад [16].

Елемент input

Категорії контенту: потоковий вміст; текстовий вміст; якщо немає атрибуту type="hidden", то інтерактивний вміст; видимий вміст.
Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.
Пропуск тегів: відсутній тег, що закриває.
Для елемента доступні глобальні атрибути, а також атрибути:
`accept, alt, autocomplete, checked, dirname, disabled, form, formaction, formenctype, formmethod, formnovalidate, formtarget, height, list, max, maxlength, min, name, minlength, multiple, name, pattern, placeholder, readonly, required, size, src, step, type, value, width`.

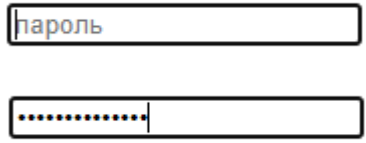
Розглянемо призначення та особливості застосування деяких з них.

Так, зокрема, атрибути:

- `disabled` – логічний атрибут, який указує на те, що користувач не може редагувати та копіювати вміст поля. Зазвичай відображаються більш тьмяним кольором.

- `readonly` – логічний атрибут, який вказує на те, що користувач не може змінювати значення поля, виділення та копіювання вмісту при цьому доступно. Може зазначатися для полів типу text, search, number, password, url, tel, email, date, month, week, time, datetime-local.
- `required` – логічний атрибут, який вказує на те, що це поле є обов'язковим для заповнення. При спробі надіслати форму, не ввівши в це поле необхідне значення, на екрані з'явиться попереджувальне повідомлення, вигляд, і вміст якого залежить від браузера, що використовується.
- `name` – ім'я елемента керування, який подається з даними форми. Якщо ім'я не вказано або ім'я порожнє, значення поля не надсилається разом із формою.
- `minlength, maxlength` – визначають відповідно мінімальну та максимальну кількість символів (як кодові одиниці UTF-16), які користувач може ввести в поле.
- `min, max` – визначають відповідно мінімальне та максимальне значення в діапазоні допустимих значень. Може зазначатися для полів типу date, month, week, time, datetime-local, number і range.
- `step` – указує величину збільшення чи зменшення значень у процесі регулювання діапазону для елементів числових типів введення date, month, week, time, datetime-local, number і range.
- `multiple` – логічний атрибут, який надає змогу вводити адреси електронної пошти, розділені комами або вибирати більше одного файлу під час завантаження. Для полів типу email і file.
- `value` – це поточне редаговане значення для полів типу text і password; текст на кнопці для полів типу button, reset і submit; визначене значення, яке відправляється на сервер для полів типу checkbox, radio та hidden.
- `placeholder` – коротка підказка про те, яка інформація очікується у полі. Відображається у полі введення до заповнення.

```
<input type="password" name="password" min="8"
placeholder="пароль" required>
```



Мал. 8 Стан поля вводу до заповнення та після

Зауваження. Не варто скрізь використовувати атрибут placeholder замість елемента label для поля форми лише із-за стилізації. Їхні призначення різні: атрибут label описує роль елемента форми, тобто, він указує, який тип інформації очікується; а атрибут placeholder є підказкою щодо формату, який повинен прийняти вміст. Форма має бути зрозумілою без нього.

Одним із основних атрибутів для елемента input є type. Атрибут type може набувати різних значень. Проте слід зауважити, що не всі значення підтримуються усіма браузерами на цей час і відображення елементів також залежить від браузера.

Можливі значення атрибуту `type`:

- text – значення за замовчуванням. Однорядкове текстове поле, розриви рядків автоматично видаляються зі значення, що вводиться.
- hidden – створює елемент керування, який не відображається, але значення якого надсилається на сервер.
- search – однорядкове текстове поле для введення рядків пошуку, розриви рядків автоматично видаляються зі значення, що вводиться. Деякі браузери відображають значок видалення x, який можна використовувати для очищення поля.
- tel – поледля введеня номеру телефону. Відображає клавіатуру телефону, на деяких пристроях із динамічною клавіатурою.

- url – поле для введення URL-адрес. Виглядає як поле для введення тексту, але має параметри перевірки та відповідну клавіатуру для підтримки браузерів та пристроїв із динамічною клавіатурою.
- email – поле для введення адреси електронної пошти. Виглядає як введення тексту, але має параметри перевірки та відповідну клавіатуру для підтримки браузерів та пристроїв із динамічною клавіатурою. Якщо опустити знак @, то при відправленні форми видає помилку.

- password – текстове поле, в якому символи, що вводяться користувачем, замінюються на зірочки, маркери, або інші, встановлені браузером значки. Видає попередження, якщо вебсайт небезпечний.
- date – надає змогу відкрити засіб вибору дати – календар (рік, місяць і день, без часу).
- month – надає змогу відкрити засіб вибору місяця і року.
- week – надає змогу відкрити засіб вибору номера тижня та року.
- time – надає змогу вводити час у 24-годинному форматі за шаблоном гг:хх.
- datetime-local – надає змогу вводити дату і час за шаблоном дд.мм.рррр гг:хх.
- number – надає змогу вводити цілі числа. Можна також використовувати атрибути step, min і max.
- range – надає змогу вводити число, точне значення якого не має значення. Відображається як віджет діапазону із середнім значенням за замовчуанням. Використовується разом з min і max для визначення діапазону допустимих значень. Для відображення значення слід прописати код JavaScript.
- color – відкриває віджет вибору кольору у шістнадцятковому форматі.
- checkbox – відображає прапорець, який дозволяє вибирати/скасувати вибір окремих значень.

- radio – відображає перемикач, що дозволяє вибрати одне з кількох варіантів з однаковим значенням name.
- file – поле для вибору одного або кількох файлів зі сховища пристрою. Після вибору файли можна завантажити на сервер за допомогою надсилання форми або маніпулювати ними за допомогою коду JavaScript та File API.
- button – створює кнопку без поведінки за замовчуанням, написом до кнопки є значення атрибута value.
- image – створює графічну кнопку надсилання форми із зображенням, визначене атрибутом src або значенням атрибута alt, якщо зображення відсутнє.
- reset – створює кнопку, яка скидає вміст форми до значень за замовчуанням.
- submit – відображає кнопку надсилання форми.

```
<label>Запис до лікаря: <input type=time
name="appointment" min="09:00" max="14:00"
="14:00"></label>
```

У формі можуть використовуватись групи перемикачів, тому для відокремлення їх і правильної роботи слід указувати одне name для групи.

```
<fieldset>
    <p>Зазначте мову  своєї доповіді</p>
    <ul>
        <li>
            <label for="uk">
              <input type="radio" id="uk" name="lg-
               report" value="uk">
              українська
            </label>
       </li>
       <li>
           <label for="en">
```

```
                <input type="radio" id="en" name="lg-
                report" value="en">
                англійська
            </label>
        </li>
        <li>
            <label for="pl">
                <input type="radio" id="pl" name="lg-
                report" value="pl">
                 польська
             </label>
         </li>
    </ul>
</fieldset>
```

Елемент textarea

Категорії контенту: потоковий вміст; текстовий вміст; інтерактивний вміст; елемент, пов'язаний із формою; видимий вміст.

Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути, а також атрибути: `autocomplete`, `rows`, `cols`, `dirname`, `disabled`, `form`, `minlength`, `maxlength`, `minlength`, `placeholder`, `wrap`.

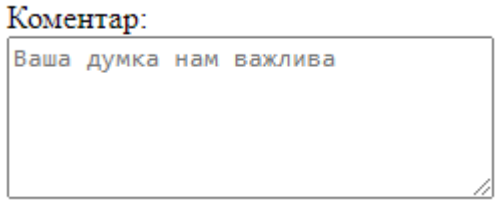
Розглянемо деякі атрибути, які не описані вище:

- `rows` – указує кількість видимих рядків тексту для елемента.
- `cols` – встановлює ширину текстової області за кількістю символів. Коли користувач вводить більше тексту, з'являється смуга прокручування. Значення за замовчуванням 20.
- `wrap` – указує, як текст переноситься в елементі:

- soft: браузер не вставляє жодних додаткових розривів рядків. Значення за замовчуванням.
- hard: браузер автоматично вставляє розриви рядків, щоб ширина кожного рядка не перевищувала ширину елемента; атрибут cols також має бути вказаний.
- off: браузер не вставляє ніяких додаткових розривів рядків, а сегменти рядків, що перевищують cols, не переносяться, і елемент прокручується по горизонталі.

Елемент textarea є багаторядковим елементом керування для введення звичайного тексту. Використовується, наприклад, для розгорнутого тесту у тестах, формах зворотного зв'язку.

```
<label for="comments">Коментар:</label> <br>
<textarea id="comments" placeholder="Ваша думка нам ва-
жлива" rows="5" cols="30" wrap="hard"></textarea>
```



Мал. 9 Елемент textarea

Списки дають можливість розташувати велику кількість пунктів форми компактно, оскільки дозволяють вибрати одне або кілька значень із запропонованої множини. За замовчуванням у полі списку відображається перший елемент. Організовують списки за допомогою елементів select, datalist, option, optgroup.

Елемент select

Категорії контенту: потоковий вміст; текстовий вміст; інтерактивний вміст; елемент, пов'язаний із формою; видимий вміст.

Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути, а також атрибути: `disabled, form, multiple, name, required, size.`

`Атрибут multiple –` це логічний атрибут, який вказує на те, що у списку можна вибрати кілька параметрів.

Елемент option

Категорії контенту: відсутня.

Контекст, в якому цей елемент може бути використаний: як дочірній елемент елементів select, datalist або optgroup.

Пропуск тегів: тег елемента </option> може бути опущений, якщо за елементом option відразу слідує інший елемент option, або якщо за ним відразу слідує елемент optgroup, або якщо в батьківському елементі більше немає вмісту.

Для елемента доступні глобальні атрибути, а також атрибути: `disabled, selected, value, label.`

Атрибут:

- `selected –` логічний атрибут, який позначає даний варіант як обраний спочатку.
- `label –` показує текст для напису, що вказує значення параметра. Якщо атрибут label не визначено, його значення дорівнює текстовому вмісту елемента.

Елемент optgroup

Категорії контенту: відсутня.

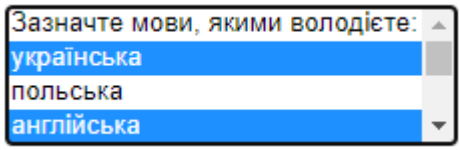
Контекст, у якому цей елемент можна використовувати: як дочірній елемент елемента select.

Пропуск тегів: тег </optgroup> може бути опущений, якщо за елементом optgroup відразу слідує інший елемент optgroup або якщо в батьківському елементі більше немає вмісту.

Для елемента доступні глобальні атрибути, а також атрибути: `disabled, label.`

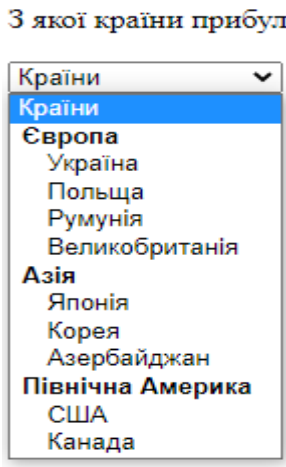
Елемент optgroup є групою елементів option із загальним написом. Браузери відображають такі групи, як пов'язані один з одним, окремо від інших елементів option.

Розглянемо випадки оформлення списків вибору та згрупованого вибору.



Мал. 10 Множинний вибір із списку

```
<select name="tech"  multiple required>
    <option value="">Зазначте мови, якими володієте:
    </option>
     <option value="1">українська</option>
     <option value="2">польська</option>
     <option value="3">англійська</option>
     <option value="4">німецька</option>
     <option value="5">французька</option>
     <option value="6">іспанська</option>
     <option value="7">румунська</option>
</select>
```



Мал. 11 Згрупований вибір із списку

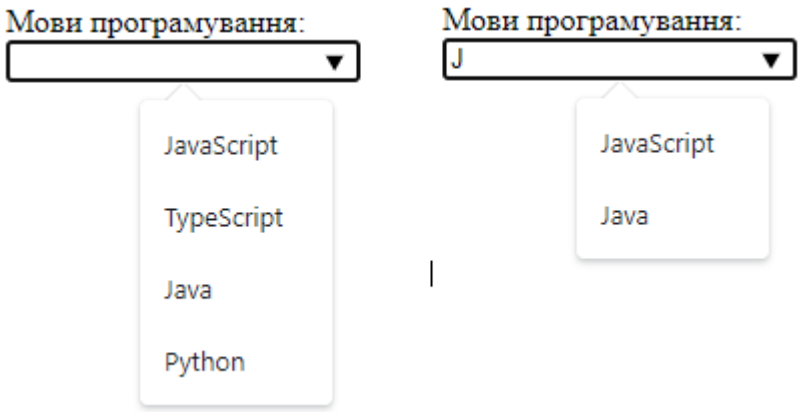
```
<form action="https://" method="get">
   <p> З якої країни прибули? </p>
   <select name="lang">
      <option value="">Країни </option>
      <optgroup label=" Європа">
         <option value="1.1">Україна </option>
         <option value="1.2">Польща</option>
         <option value="1.3">Румунія</option>
         <option value="1.4">Великобританія</option>
      <optgroup label="Азія">
         <option value="2.1">Японія</option>
         <option value="2.2">Корея</option>
         <option value="2.3">Азербайджан</option>

      <optgroup label="Північна Америка">
         <option value="3.1">США</option>
         <option value="3.2">Канада</option>
     </select>
     <p><input type=submit value="Надіслати"></p>
  </form>
```

Елемент datalist

Категорії вмісту: потоковий вміст, текстовий вміст.
Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.
Пропуск тегів: жоден з тегів не може бути пропущений.
Для елемента доступні глобальні атрибути.

Елемент datalist містить набір елементів option, які представляють рекомендовані значення для вибору в елементі input. Ці значення відображаються для input вигляді списку, що розкривається, і відфільтровуються якщо вводити дані у поле. Збіг шукається по всьому значенню. Елемент datalist підключається до елемента input за допомогою атрибута list, при цьому значення id елемента datalist має збігатися зі значенням атрибута list елемента input .

При рендерингу сторінки елемент datalist нічого не відображає, він прихований разом із дочірніми елементами.



Мал. 13 Список datalist

```
<label>
     Мови програмування: <br>
      <input name="program-lang" list="list-lang">
      <datalist id="list-lang">
          <option value="JavaScript">
          <option value="TypeScript">
          <option value="Java">
          <option value="Python">
      </datalist>
 </label>
```

Елементи progress, meter

Категорії вмісту: потоковий вміст, текстовий вміст, маркований елемент, видимий вміст.

Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента progress доступні глобальні атрибути, а також атрибут value, який задає поточне значення елемента, і атрибут max, який задає верхню межу діапазону. Якщо не вказано максимальне значення, воно за промовчанням дорівнює 1.

Елемент progress виводить на екран віджет, який відображає індикатор виконання завдання. З його допомогою користувачі можуть відслідковувати перебіг тривалої операції. Індикатор виконання може показати або приблизний відсоток завершення, або вказати, що операція виконується.

```
<progress id="p" max="100" value= ="50">... </progress>
```

Для того, щоб відображалось значення віджету, потрібний скрипт. Якщо значення атрибут value явно не зазначати, то бігунок прогресу буде постійно рухатись.

Для елемента meter доступні глобальні атрибути, а також атрибути `value, min, max, low, high, optimum.`

Атрибут:

- `low` – задає нижню числову межу верхньої межі діапазону. Якщо не вказано чи більше максимального значення, то дорівнює максимальному значенню.
- `high` – задає верхню числову межу нижньої межі діапазону. Якщо не зазначено або менше мінімального, то дорівнює мінімального значення.
- `optimum` – указує на оптимальне числове значення в межах діапазону. При використанні з атрибутом low і high вказує, що кращим вважається вищий діапазон. Якщо значення між атрибутом min і low, переважним вважається нижчий діапазон.

Елемент meter є скалярним виміром у межах заданого діапазону або дробове значення, що визначає частину чогось.

```
<p>На олімпіаді із 50 учасників третє завдання виконали
лише 10 учасників:</p>
    <meter min="0" max="50" value="10">10</meter>
```

Елемент output

Категорії контенту: потоковий вміст; текстовий вміст; елемент, пов'язаний із формою; видимий вміст.

Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути, а також атрибути:

- `for` – указує розділений пробілами список ідентифікаторів елементів із вхідними даними, які брали участь у обчисленнях.
- `form` – указує ідентифікатор зв'язаної форми.

- `name` – визначає ім'я, на яке можна буде посилатися з обробників подій елементів керування форми. Саме значення елемента не передається під час відправлення форми.

Елемент output представляє елемент-контейнер, у який можнавиводити результати обчислень або результат дії користувача.
У [16] наведений приклад використання елемента output


```
<form oninput="result.value=parseInt(a.value)+
parseInt(b.value)">
    <input type="range" id="b" name="b" value="50"> +
    <input type="number" id="a" name="a" value="10">=
    <output name="result" for="a b">60</output>
</form>
```

Кнопки у формі дозволяють користувачам надсилати дані форми на сервер, передавати дані до неї, очищати вміст форми або робити будь-які інші дії. Кнопки можна створити елементами керування input і button. Для коректного відображення елемента різними браузерами потрібно вказувати атрибут type, а для кнопок, створених за допомогою input, ще й атрибут value (крім type="image"), значення якого й буде відображатись як напис на кнопці.

У випадку type="image" написом кнопки буде зображення.

```
    <input type="submit">,  <input
type="image">,
    <input type="reset">,  <input type="button">
```

Слід зазначити, що кнопки можна використовувати й поза формами. Для кнопок прописують дії за допомогою скриптів.

Елемент button

Категорії контенту: потоковий вміст; текстовий вміст; інтерактивний вміст; елемент, пов'язаний із формою; видимий вміст.

Контекст, у якому цей елемент можна використовувати: де очікується текстовий вміст.

Пропуск тегів: жоден з тегів не може бути пропущений.

Для елемента доступні глобальні атрибути, а також атрибути: `formenctype, formaction, formmethod, formnovalidate, formtarget, form, disabled, name, type, value.`

Атрибут:

- name – визначає ім'я кнопки для відправлення форми. На сервер відправляється пара ім'я=значение, де ім'я задається атрибутом name, а значення – атрибутом value. Значення може збігатися з текстом на кнопці.
- type – керує поведінкою кнопки під час її активації:
  - submit – надсилає дані форми, значення за промовчанням.
  - reset – скидає дані форми на їх вихідні значення.
  - button – кнопка не має стандартної поведінки і нічого не робить при натисканні за замовчуванням. Якщо кнопка на сторінці не призначена для надсилання даних форми на сервер, встановлення цього значення обов'язкове.
- value – визначає значення, пов'язане з ім'ям кнопки, коли вона відправляється разом із даними форми. Значення включається у відправку форми тільки в тому випадку, якщо кнопка використовувалася для ініціювання відправки форми. Також атрибут value використовується для доступу до даних через скрипти.

Наприклад, у середині форми:

```
<input type="submit" value="Відправити">
або
<button type="submit"> Відправити</button>
```

Для написання дій користувача не за замовчуванням використовують розмітку

```
<button type="button"> Напис кнопки</button>
```

Запитання для повторення

1. Для чого використовуються та за допомогою чого створюються вебформи? Наведіть приклади застосування.
2. Де варто проводити валідацію? Чому? Що таке вбудована валідація вебформ? Якими засобами вона здійснюється?
3. Який HTML-елемент є контейнером для елементів форми? Яка відмінність між get- і post-методами для form?
4. Для чого призначений елемент fieldset? Як задати для елемента fieldset заголовок?
5. Чому важливо задавати мітки-написи для елементів форми? Яким HTML-елемент вони створюються.
6. Якими елементами можна створювати списки на формах? Наведіть їхні особливості.
7. Елемент input, його атрибут type, приклади та особливості застосування.
8. Призначення елемента textarea.
9. Як створюються і діють кнопки у вебформах?
10. Чи можна елементи форм використовувати поза формами?

Література

1. Веброзробка. [Електронний ресурс] – Режим доступу : https://uk.wikipedia.org/wiki/Веброзробка
2. Графічні редактори. [Електронний ресурс] – Режим доступу : https://hexlet.io/courses/layout-designer-basics/lessons/code-editors/theory_unit
3. Список HTML редакторів. [Електронний ресурс] – Режим доступу : https://en.wikipedia.org/wiki/List_of_HTML_editors
4. Які існують технології для розробки сайтів та кому вони підходять. [Електронний ресурс] – Режим доступу : https://icstudio.online/post/tehnologii-dlya-rozrobky-sajtov
5. Тестування веб-проєктів: основні етапи та поради. [Електронний ресурс] – Режим доступу : https://qalight.ua/baza-znaniy/testuvannya-veb-pro%D1%94ktiv-osnovni-etapi-ta-poradi/
6. Що таке Хостинг? Які види хостингу бувають. [Електронний ресурс] – Режим доступу : https://hyperhost.ua/info/uk/shcho-take-khosting-yaki-vidi-khostingu-buvayut
7. What Is SEO – Search Engine Optimization? [Електронний ресурс] – Режим доступу : https://searchengineland.com/guide/what-is-seo
8. HTML Living Standart. [Електронний ресурс] – Режим доступу : https://html.spec.whatwg.org/
9. Що таке сучасна верстка сайту: підходи у створенні веб сторінок. [Електронний ресурс] – Режим доступу : https://frontend.lviv.ua/shho-take-suchasna-verstka-sajtu
10. Ethan Marcotte. Responsive Web Design. [Електронний ресурс] – Режим доступу : https://pdfgoes.com/download/4266980-Responsive%20Web%20Design%20Ethan%20Marcotte.pdf

11. The Power of Figma as a Design Tool. [Електронний ресурс] – Режим доступу : https://www.toptal.com/designers/ui/figma-design-tool
12. W3C Accessibility Guidelines (WCAG) 3.0. [Електронний ресурс] – Режим доступу : https://www.w3.org/TR/wcag-3.0/
13. Chrome DevTools. [Електронний ресурс] – Режим доступу : https://developer.chrome.com/docs/devtools/
14. HTML. [Електронний ресурс] – Режим доступу : https://en.wikipedia.org/wiki/HTML
15. Replaced elements. [Електронний ресурс] – Режим доступу : https://developer.mozilla.org/en-US/docs/Web/CSS
16. HTML elements reference. [Електронний ресурс] – Режим доступу : https://developer.mozilla.org/en-US/docs/Web/HTML/Element
17. A free guide to HTML. [Електронний ресурс] – Режим доступу : https://htmlreference.io/
18. HTML5 підручник/ [Електронний ресурс] – Режим доступу : https://w3schoolsua.github.io/html/index.html
19. HTML Currency Symbols, Currency Entities and ASCII Currency Character Code Reference. [Електронний ресурс] – Режим доступу : https://www.toptal.com/designers/htmlarrows/symbols/
20. 20. Div divisiveness. [Електронний ресурс] – Режим доступу : https://www.scottohara.me/blog/2022/01/20/divisive.html
21. Sections. [Електронний ресурс] – Режим доступу : https://www.w3.org/TR/2021/NOTE-html53-20210128/sections.html
22. Barry Pollard. Setting Height And Width On Images Is Important Again. [Електронний ресурс] – Режим доступу : https://www.smashingmagazine.com/2020/03/setting-height-width-images-important-again/

Навчальне видання

Готинчан Тетяна Іванівна

Основи веброзробки: HTML і CSS

Частина 1

Навчальний посібник

Відповідальний за випуск **Черевко І.М.**
Комп'ютерний набір **Готинчан Т.І.**