МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

В. П. Молчанов

ОСНОВИ ПРОЕКТУВАННЯ WEB-ВИДАНЬ

Навчальний посібник

Харків
ХНЕУ ім. С. Кузнеця
2017

УДК 004.91(075)
ББК 32.973–018.2я7
М 76

Рецензенти: завідувач кафедри інформаційних технологій та вищої математики ХННІ ДВНЗ Університету банківської справи (м. Харків), д-р екон. наук, канд. техн. наук, доцент *С. В. Кавун*; завідувач кафедри інформаційних технологій Українського державного університету залізничного транспорту (м. Харків), д-р техн. наук, професор *А. О. Каргін*; канд. техн. наук, доцент кафедри медіасистем і технологій Харківського національного університету радіоелектроніки *А. В. Бізюк*.

Рекомендовано до видання рішенням ученої ради Харківського національного економічного університету імені Семена Кузнеця.
Протокол № 6 від 06.03.2017 р.

Самостійне електронне текстове мережеве видання

Молчанов В. П.

М 76 Основи проектування WEB-видань : навчальний посібник : [Електронне видання] / В. П. Молчанов. – Харків : ХНЕУ ім. С. Кузнеця, 2017. – 159 с.

ISBN 978-966-676-670-3

Подано матеріал, що допомагає засвоїти всі лекційні теми навчальної дисципліни. Теорію викладено в супроводі великої кількості ілюстративного матеріалу у вигляді відповідних вікон, рисунків і прикладів. Методикою викладення матеріалу передбачене активне залучення студента до навчального процесу. Наведено приклади, що дозволяють отримати цілісне уявлення про можливості технології і можуть використовуватися як довідковий матеріал під час виконання лабораторних робіт і завдань для самостійної роботи.

Рекомендовано для студентів спеціальності 186 "Видавництво та поліграфія".

УДК 004.91(075)
ББК 32.973–018.2я7

ISBN 978-966-676-670-3

Вступ

Навчальна дисципліна "Основи проектування WEB-видань" відноситься до професіонального циклу базових навчальних дисциплін. Вона забезпечує підготовку фахівця щодо вміння створювати документи для мережі Інтернет і розміщувати їх у мережі.

У рамках дисципліни розглянуто основні сучасні WEB-технології та технологічні засоби їх підтримки.

Отримані в ході вивчення навчальної дисципліни знання та вміння необхідні не тільки тим, хто створюватиме WEB-сайти самостійно, але і тим, хто для виконання цих робіт вважає за краще звернутися до професіоналів. Знання предмета стане для них запорукою ефективної й якісної взаємодії з розробником.

Метою вивчення навчальної дисципліни є формування у студентів системи теоретичних знань про технологічні компоненти сервісу WWW, їх місце серед інших комп'ютерних технологій і комплекс умінь зі створення документів для сервісу, їх розміщення в мережі Інтернет та аналізу функціонування.

Завданням навчальної дисципліни є оволодіння навичками зі створення ресурсів для сервісу WWW з урахуванням сучасних дизайнерських концепцій і можливостей технологічних засобів, навчитися оцінювати якість і ефективність створених ресурсів.

Об'єктом навчальної дисципліни є процеси створення документів для сервісу WWW.

Предметом навчальної дисципліни є документи та технологічні засоби сервісу WWW.

Структурно посібник відповідає програмі навчальної дисципліни та складається з чотирьох розділів, які відповідають чотирьом змістовим модулям. Структура всіх розділів однотипна: основна ідея, цілі вивчення, вступ, основний текст, висновки, теоретичні запитання та практичні завдання для самостійної роботи. Кожний розділ містить підрозділи, які відповідають темам.

Для викладення матеріалу розділів використано багато різних прикладів. Закінчені приклади, які можуть бути використані для запуску на комп'ютері, виділені рамкою та шрифтом. Інший тип прикладів – це теж цілком працездатні фрагменти, які також можна використати для створення коду.

У тексті розділів в рамці із позначкою  поміщені допоміжні матеріали, які під час першого прочитання можна пропустити.

Наприкінці кожного розділу наведено список контрольних запитань для самостійної перевірки засвоєння теоретичного матеріалу та набір завдань для самостійної практичної роботи.

Перший розділ присвячений створенню WEB-сторінок і розмітці тексту засобами HTML. Розглянуто використання всіх основних елементів, а також ряд проблемних питань, пов'язаних з семантичної розміткою та використанням мікроданих.

У другому розділі розглянуто форматування HTML-сторінок з використанням таблиць стилів. Розглянуто сам механізм застосування таблиць CSS, а також процес верстання, включаючи такі проблемні питання, як адаптивне верстання та нові модулі CSS3.

Третій розділ присвячений вбудовуванню скриптів у HTML-сторінку. Основну увагу приділено синтаксису мови й організації виконання скриптів в браузері.

Четвертий, заключний, розділ присвячений інтегрованому використанню трьох розглянутий базових технологій і сучасних засобів, що забезпечують ефективну розробку клієнтської частини WEB-ресурсу. Розглянуто використання бібліотек.

Автор висловлює глибоку вдячність рецензентам – доктору технічних наук, професору Каргіну А. О., доктору економичних наук, доценту Кавуну С. В. і кандидату технічних наук, доценту Бізюку А. В. за велику увагу до навчального посібника та надані рекомендації та поради стосовно його змістовного наповнення.

Розділ 1. Створення WEB-документів

Основна ідея розділу

Розділ присвячений характеристиці етапів і плануванню робіт зі створення сайта, а також створенню HTML-документів за допомогою мови розмітки тексту.

Ключові поняття розділу: етапи створення сайта, бриф, концепція сайта, дизайн сайта, мова розмітки тексту, WEB-сервіс, WEB-сторінка, теги, елементи, підстановки.

Питання розділу:

1.1. Проектування WEB-сайта.

1.1.1. Характеристика сервісу WWW.

1.1.2. Планування створення сайта.

1.2. Розмітка тексту з використанням HTML.

1.2.1. Введення в HTML.

1.2.2. Розмітка тексту.

1.2.3. Створення елементів навігації.

1.2.4. Розміщення об'єктів.

Цілі вивчення розділу

Інформація, подана у розділі, надає студентові можливість сформувати такі **компетентності:**

знання:

змістовності етапів створення сайтів;

вимог до дизайну документів для сервісу WWW;

складу та змісту документів, які визначають розробку;

мов розмітки гіпертексту, структури документів;

основних елементів WEB-сторінок, відповідних тегів та їх атрибутів;

уміння:

планувати роботу зі створення сайтів;

приймати основні дизайнерські рішення, створювати концепцію сайта;

створювати WEB-сторінки з використанням мов розмітки гіпертексту;

розміщувати основні елементі на WEB-сторінках з урахуванням особливостей відображення у різних браузерах;

керувати взаємодією елементів з метою їх відповідного відображення;

комунікації:

взаємодія із замовником під час приймання замовлення на розробку;

взаємодія з розробником під час супроводження розробки;

обґрунтування дизайнерських рішень перед членами команди;

автономність і відповідальність:

пошук шляхів вирішення проблем, що виникають у ході взаємодії з замовником;

самостійний пошук і використання методів забезпечення кросплатформеності відображення сторінок;

використання альтернативних рішень щодо розміщення елементів.

Вступ

Створення сайта – це процес, в якому беруть участь кілька фахівців різного профілю. Це і дизайнери (фахівці, які мають художню підготовку), і програмісти, і фахівці з верстання, зі створення контенту. Для отримання якісного результату необхідно організувати злагоджену роботу всього колективу. Крім того, в ході розроблення потрібно підтримувати ефективну взаємодію з замовником. З цією метою необхідно добре уявляти та документально оформляти цей процес.

Основним видом документів, які використовуються сервісом WWW, є WEB-сторінка. Вона містить текст, розмічений з використанням мови HTML. Незважаючи на велику кількість засобів, що дозволяють створювати ці документи і без знання мов розмітки, ефективна й якісна розробка неможлива без фахівців. Ці фахівці повинні вміти створювати, аналізувати HTML-документи, вносити необхідні зміни. Мова розмітки (HTML) дозволяє описати дизайнерський задум з використанням формальних конструкцій, що описують окремі елементи. Їх необхідно знати і вміти використовувати.

Ці питання і будуть розглянуті в даному розділі.

1.1. Проектування WEB-сайта

1.1.1. Характеристика сервісу WWW

Основний прорив у розвитку комп'ютерних технологій відбувся в області створення та використання мереж. Саме тут досягнуті найбільш значущі та відчутні результати, які повинен знати та вміти використовувати сучасний фахівець.

Особливе місце у процесі інформатизації суспільної та виробничої сфери належить мережі Інтернет. Вона не тільки розвивається й удосконалюється, як власне комп'ютерна мережа, а її технології переносяться, інтегруються з іншими, такими традиційними, як телефонія та телебачення. Інтернет-технології (тобто сукупність засобів і методів передавання та використання даних у комп'ютерній мережі Інтернет) розвиваються та проникають в такі сфери:

інтеграція з традиційними ЗМІ;

комунікаційні системи;

системи електронних платежів;

навчання;

розподілені бази даних;

електронна комерція;

реклама та ін.

Водночас мережа виступає як транспортна система для даних. Вона будується на основі значної кількості протоколів, що утворюють ієрархічну структуру, але базовими є TCP/IP, які визначають правила адресації та пересилання даних мережею. Самі дані, які транспортуються, можуть мати різні формати, і їх використання забезпечується власними протоколами та програмним забезпеченням.

З точки зору користувача, Інтернет є набором послуг (сервісів), кожна з яких працює зі своїми даними, за своїми протоколами, зі своїм програмним забезпеченням. **Сервіси** – це надбудова над мережею Інтернет. Вони постійно розвиваються, вдосконалюються. Одні перестають використовуватися, з'являються нові. Найбільш популярними сервісами на даний час є:

електронна пошта (e-mail) – забезпечує обмін між користувачами текстовими повідомленнями з можливістю пересилання файлів;

FTP – забезпечує пересилання мережею довільних файлів, доступ до файлових систем комп'ютерів;

World Wide Web (WWW) – сервіс, що забезпечує доступ до системи гіпертекстових мультимедійних документів (WEB-сторінок).

Останній сервіс є найпоширенішим і затребуваним, саме він і буде предметом подальшого розгляду. WWW-технологією або **WEB-технологією** називають комплекс засобів і методів створення та використання ресурсів для сервісу WWW. Ці ресурси отримали назву WEB-сторінок.

Web-сторінка є текстовим файлом, що містить розмічений текст (текст з виділеними смисловими елементами), за інтерпретації цього тексту браузером (програмою перегляду) у вікні створюється зображення сторінки.

Сторінка може зберігатися на сервері у вигляді файла з розміченім текстом, а може формуватися динамічно програмою, розробленою з використанням мови програмування. Однак і в останньому випадку браузеру повинен відправлятися розмічений текст.

Сьогодні для розмітки тексту найчастіше використовується мова HTML (Hyper-Text Markup Language). Це основа, серцевина всієї технології. Розширення цих файлів – htm (html).


Хоча більшість нових технологічних рішень заснована на використанні мови XML (Extensible Markup Language – розширювана мова розмітки), переважну більшість ресурсів в мережі все ж таки складають HTML-документи.

Сторінка зазвичай містить дані, які займають один-два екрану комп'ютера. За необхідності розміщення більшого обсягу даних вони розбиваються на кілька сторінок і зв'язуються між собою за допомогою посилань. Така сукупність кількох пов'язаних сторінок називається **сайтом**. Сторінки сайта зазвичай пов'язані не тільки інформацією, а й єдиним оформленням, місцем розміщення й адресою (URL).

Основною версією мови, підтримуваної на поточний момент найбільшою кількістю браузерів (включаючи старі версії) є HTML4. Однак, все нові браузери підтримують специфікацію HTML5, яка істотно розширює як образотворчі засоби мови, так і надає в розпорядження розробника нові можливості.


У HTML5 реалізовано безліч нових синтаксичних особливостей. Наприклад, елементи <video>, <audio> і <canvas>, а також можливість використання SVG і математичних формул. Ці нововведення розроблені для спрощення створення й управління графічними та мультимедійними об'єктами в мережі без необхідності використання сторонніх API і плагінів. Інші нові елементи – такі, як <section>, <article>, <header> і <nav>, розроблені для того, щоб збагачувати семантичний вміст документа (сторінки). Нові атрибути були введені з тією ж метою, хоча ряд елементів і атрибутів був видалений. Деякі елементи (наприклад, <a>, <menu> і <cite>) були змінені, перевизначені або стандартизовані. API і DOM стали основними частинами специфікації HTML5. HTML5 також визначає деякі особливості обробки помилок верстання, тому синтаксичні помилки повинні розглядатися однаково всіма сумісними браузерами.

Можливості мови розмітки тексту з управління відображенням сторінок у вікні браузера досить обмежені. Тому для форматування використовується CSS (Cascading Style Sheets – каскадні таблиці стилів). CSS-код – це список інструкцій для браузера з управління зовнішнім виглядом відображення елементів сторінок.


Специфікації CSS постійно удосконалюються. В даний час основною версією є CSS2, однак здійснюється поступовий перехід (шляхом упровадження в браузери нових можливостей) до версії CSS3. CSS3 забезпечує можливість створювати закруглені кути блоків, тіні, лінійні та радіальні градієнти, анімовані елементи та трансформації без використання програм. Специфікація може застосовуватися не тільки до HTML-документів, але й до будь-яких інших XML-документів, наприклад, SVG або XUL.

Власне HTML забезпечує створення статичного документа. Для надання документу динамічних властивостей (різноманітних ефектів, зміну змісту відповідно до деяких умов) він доповнюється програмними компонентами. Нині використовується дві технології вбудовування програм у Web-сторінку для виконання на боці клієнта (на комп'ютері користувача): сценарії (script) та аплети (applet).

Сценарій є програмою на мові високого рівня, яка управляє вікном браузера. На практиці в якості мови сценаріїв використовується підмножина мови Java або VBasic. Сам текст такої програми розміщується усередині HTML-файла. Ці програми дозволяють змінювати фон сторінки, "оживлювати" меню, створювати рухомі рядки тощо. Сьогодні у мережі важко знайти сторінку, яка б не містила сценаріїв.

Аплет є програмою, що відкомпілювалася не в машинні коди (як завжди), а в байт-коди (проміжна мова). Для виконання такої програми потрібна спеціальна програма-інтерпретатор, названа віртуальною машиною або оболонкою часу виконання. Сьогодні цей підхід реалізований з використанням мови Java. Байт-код, що відкомпілювався, зберігається в окремому файлі на сервері (файл має розширення class), у HTML-файлі знаходиться тільки його адреса. Під час відтворення на екрані Web-сторінки браузер посилає на сервер запит і, отримавши байт-код, передає його віртуальній машині, яка виконує запрограмовані дії. Віртуальна машина (оболонка часу виконання) вбудована в усі популярні браузери.

Усі розглянуті підходи, що розширюють можливості WEB-сторінок, припускають пересилку на комп'ютер користувача та подальший запуск на ньому деякої програми, так або інакше пов'язаної з отриманим HTML-документом. Проте зміна сторінок може виконуватися і на сервері перед їх відправкою користувачеві. Технології, що забезпечують це, спрямовані не на створення спецефектів на екрані, а на адаптацію сторінок до умов, які можуть міститися в запиті, або автоматичне оновлення змісту сторінок. За деякими оцінками, більше половини всіх професійних сторінок у мережі генерується й оновлюється динамічно – на основі інформації з баз даних, у відповідь на дії користувача або залежно від зовнішніх даних (поточна дата і т. п.). Найбільшого поширення для розроблення динамічних ресурсів отримали так звані технології активних сторінок ASP.NET і PHP.

Загальний підхід до створення такого роду ресурсів реалізований в рамках стандарту **CGI** (Common Gateway Interface – загальний інтерфейс шлюзів), який містить правила генерації вихідних документів (Web-сторінок). Сутність його полягає в тому, що у відповідь на запит (URL) клієнтської програми (браузера) сервер запускає програму (CGI-програму), яка сформує та відправить браузеру HTML-документ.

Сама програма знаходиться на сервері у вигляді виконуваного файла. У ході виконання програми її вихідний потік (це повинен бути розмічений текст) сервер направляє клієнту у відповідь на запит.

ASP.NET – технологія створення WEB-додатків і WEB-сервісів, яка розроблена Microsoft, є складовою частиною платформи Microsoft.NET. Володіє дуже широкими можливостями. Створена сторінка може змінюватися на сервері перед відправкою, події, що виникають на клієнті (в тому числі і дії користувача), можуть бути оброблені як на клієнті, так і на сервері. Її підтримка включена в середовище VisualStudio; завдяки цьому істотно полегшується процес розробки і відлагодження.

Водночас забезпечується можливість використання всіх переваг об'єктного програмування. Однак, слід враховувати пропрієтарний характер усіх компонент і можливість використання тільки на серверах Microsoft.

PHP є вільно поширюваною альтернативою ASP.NET. Поширена навіть більш широко, ніж ASP.NET. Заснована на мові PHP. Інструкції мови вбудовуються безпосередньо в текст сторінки та виконуються на сервері перед відправкою сторінки клієнту. Можливості мови істотно розширені за рахунок великої кількості бібліотек самого різного призначення. Підтримується всіма серверами, в тому числі і Microsoft.

Таким чином, WEB-сторінка, яка відображається в браузері, може містити не тільки дані, що зберігаються в файлі на сервері, а й результати виконання програм як на сервері, так і на клієнті. У підсумку сайт набуває риси додатка, забезпечуючи виконання різних функцій. Загальною тенденцією розвитку технології є створення не просто сайтів (групи зв'язаних сторінок), а інтерактивних WEB-додатків (табл. 1.1).

Таблиця 1.1

Відмінності між WEB-сайтом і WEB-додатком

WEB-сайт	WEB-додаток
Контент	
інформаційний (як статичний, так і динамічний)	інтерактивний, динамічний
Призначення (розв'язувані завдання)	
вивчення інформації	виконання деякої роботи (пошук даних, обчислення, бізнес-логіка, гра ...)
Розподіл функцій	
реалізація більшості функцій на клієнті	реалізація більшості функцій на сервері, браузер використовується в основному для реалізації інтерфейсу з користувачем
Використовувані технології	
HTML, CSS, JScript	HTML, CSS, JScript, ASP.NET, PHP та ін.

Отже, поняття WEB-технологія достатньо складне та багатогранне. Воно включає багато складових, однак серед них є основні (базові). Це – HTML, CSS, JavaScript. Усі вони реалізуються на машині користувача в браузері. Їх й потрібно вивчати у першу чергу. Знання цих компонент є обов'язковим для всіх учасників розробки WEB-ресурсів.

1.1.2. Планування створення сайта

Для початку необхідно зазначити, що процес розроблення сторінки (сайта) включає дві складові – технічну та художню.

Художня, оформлювальна робота (WEB-дизайн) повинна виконуватися фахівцями, які мають відповідну художню підготовку й ознайомлені з основними особливостями WEB-технології.

Технічна – вимагає знання всіх технологічних компонентів.

Створення сайта – це не разовий акт, а достатньо складний багатоетапний проект. Практика, що склалася, дозволяє розглядати його як послідовність таких етапів:

планування сайта;

розроблення дизайну, включаючи створення елементів оформлення (логотипи, флеш, шрифти і т. п.);

HTML-верстання сайта;

контент-наповнення сайта;

розміщення сайта на хостінги та тестування.

Етап планування. На цьому етапі замовник спільно з розробником формулюють цілі та завдання проекту, визначають його роль і цільову аудиторію, розробляють загальну стратегію просування проекту. У ході цієї роботи широко використовується *бриф* – документ, що складається з запитань до замовника, відповіді на які дозволяють розробнику скласти уявлення про бачення проекту замовником. Правильно поставлене завдання дозволить розробнику врахувати особливості замовлення й оптимізувати сайт.

На додаток до з'ясування цілей, сформульованих замовником, доцільно провести аналіз області діяльності замовника, особливостей його бізнесу (сезонність, цільова споживча група, обсяг ринку і т. д.). Цей аналіз включає і вивчення конкурентів, а також їх WEB-ресурсів, дозволяючи оцінити переваги та недоліки конкурентних компаній і їх ресурсів, на підставі чого можна створити сайт, який буде вигідно відрізнятися.

Цільова аудиторія – це ті люди, для яких створюється сайт, тобто відвідувачі, в яких зацікавлений власник сайта. Аналіз аудиторії необхідний для врахування їх особливостей і переваг. Від цього безпосередньо залежатиме і відвідуваність, і кількість потенційних і реальних клієнтів. За наявності ресурсів (коштів) для визначення можливої аудиторії (кола користувачів) можуть проводиться соціологічні дослідження.

У ході спілкування із замовником, з'ясування цілей публікації і цільової аудиторії розробник формулює концепцію майбутнього сайта, в якій у довільній формі описує своє бачення результату. Зазвичай концепція включає опис ключових рішень за проектом. Концепція має бути зрозумілою та не містити спеціальних технічних деталей. Концепція узгоджується з замовником.

На основі концепції і побажань замовника розробляється технічне завдання (ТЗ). Воно слугує основою для планування розробки за термінами й оцінювання її вартості. У цьому документі чітко прописуються всі важливі технічні деталі й особливості майбутнього проекту. ТЗ містить інформацію про назву сайта, дизайн і навігацію, ширину сторінок і оптимізацію їх відображення в різних браузерах і режимах, функції та інші особливості ресурсу, а також обов'язково – терміни виконання робіт. Саме ТЗ керуватимуться дизайнери, програмісти й інші фахівці в ході своєї роботи.

ТЗ повинне якнайповніше специфікувати всі елементи розробки. Це основа для планування, тестування та здачі проекту. *Задача ТЗ* – максимально детально визначити всі аспекти робіт на сайті, створити єдине бачення (це дуже важливо) проекту з боку замовника та виконавця. Обов'язково повинні враховуватися моменти, пов'язані з підготовкою необхідної інформації й узгодження етапів роботи, особливо ті, що виконуються різними виконавцями.

Після складання та перевірки виконавцями, ТЗ затверджує замовник. Воно є невід'ємною частиною договору. У разі виникнення спірних або конфліктних ситуацій, що стосуються обсягів виконаних робіт, це важливий документ для врегулювання конфлікту, оскільки він є основним для розв'язування всіх питань, що виникають під час передання продукту замовнику.

Розроблення дизайну публікації. Це специфічний творчий етап. Вибір має бути обґрунтований, пов'язаний з іншими рішеннями (цілі, аудиторія) й узгоджений із замовником. У цьому процесі повинні враховуватися: загальні особливості дизайну Web-сторінок; підходи до типології й інформаційна структура створюваного Web-сайта; фізіологічні та психологічні особливості сприйняття аудиторією користувачів. Дизайнерські рішення можуть бути єдиними для всього сайта або відрізнятися для різних рівнів.

У ході цієї роботи ухвалюються рішення з:

основних принципів композиції Web-сторінок;

вибору колірного рішення;

побудови модульної сітки Web-сторінок.

На основі цих рішень надалі виготовляються ескізи, вибирається (створюється) модульна сітка, шліфуються композиція, колірне рішення, логіка посилань і ув'язка рішень на окремих сторінках.

Для детальнішого та більш професійного ознайомлення з цією частиною роботи слід звернутися до спеціальної літератури, наприклад, книги Д. Кирсанова [3].

Ще однією стороною дизайну (а може головною) є встановлення відповідності оформлення функціональному призначенню. Це юзабіліті. Стосовно комп'ютерної техніки терміном юзабіліті визначають концепцію розробки призначених для користувача інтерфейсів програмного забезпечення, орієнтовану на максимальну психологічну й естетичну зручність для користувача. Тут теж є свої закони і правила [6].

Верстання сайта. *Верстання* – це процес розроблення HTML-коду і CSS-специфікацій для створення в вікні браузера зображення сторінки відповідно до розробленого дизайна. У ході верстання використовуються раніше створені елементи оформлення (зображення, логотипи, фонові зображення і т. п.).

Одночасно з верстанням може розроблятися додатковий функціонал (скрипти, ефекти тощо).

Контент-наповнення сайта. На цьому етапі інформація, яка надана замовником, розміщується на сайті або записується в базу даних. Це може бути текст, графіка. За необхідності виконується її перетворення.

Розміщення сайта на хостінги та тестування. Тестування сайта має проводитися на всіх етапах, включаючи останній, після розміщення його на сервері. У ході тестування зазвичай перевіряється функціонування (робота меню, форм, ефекти і т. д.), відповідність верстання макетів за різних умов перегляду, зручність інтерфейсу, відповідність наповнення сторінок вихідного контенту, орфографія та пунктуація текстів.

1.2. Розмітка тексту з використанням HTML

1.2.1. Введення в HTML

1.2.1.1. Поняття розмітки тексту

Процес розмітки полягає у виділенні структурних елементів документа, співвідношення їх з класифікаційними ознаками (розстановка міток) і визначення їх змісту. Сама класифікація може бути визначена поза документом (HTML – набір міток для опису Web-сторінок) або усередині (XML Сайт готовий до використання засіб опису набору міток). Дещо спрощено, але правильно за сутністю. Сьогодні основним засобом розмітки сторінок слугує HTML 5. Тому під час розгляду розмітки сторінок доцільно орієнтуватися саме на цю версію.


Під час вибору формату для розповсюдження документів у мережі Інтернет у рамках нового сервісу WWW розробники зупинилися на **мовах розмітки**.

У той період (1990-ті роки) користувачі мережі працювали на різних платформах, включаючи комп'ютери з текстовими моніторами. Тому потрібний був формат, який був би максимально незалежний від параметрів апаратури кінцевого користувача, але зберігав усі необхідні властивості документів (гіпертекстовий характер, поєднання тексту з графікою й іншими мультимедійними компонентами). Була розроблена мова опису таких документів – HTML. У 1993 р. була створена версія HTML 1.2; у 1995 р. прийнятий стандарт HTML 2.0; 1996 р. – HTML 3.0; 1997 – HTML 4.0 і, нарешті, в 2014 р. – HTML5.

Необхідно зазначити, що стандарт – це компроміс між ідеологами та розробниками стандартів, з одного боку, та розробниками браузерів – з іншого. Причому останні дуже часто (як правило, в цілях конкурентної боротьби) відступають від стандартів, зазвичай випереджаючи їх. Так, остання версія мови багато в чому підтримувалась браузерами навіть раніше, ніж був прийнятий стандарт.

1.2.1.2. Синтаксис мови HTML

Синтаксис – це набір правил, за якими створюються всі конструкції мови. HTML містить всього три основні конструкції: елементи, теги та підстановки.

Елементи – це складові частини документа (абзаци тексту, заголовки, малюнки і т. п.).

Теги – це одиниці розмітки. Тег (*tag*) – дослівно "бірка", "ярлик". Теги, по-перше, розділяють початковий неформатований текст на елементи, по-друге, створюють нові елементи, яким ніщо не відповідало в нерозміченому тексті (наприклад, графічні вставки, Jawa-аплети). У HTML 5 усі теги парні (це пари відкривальних та закривальних тегів), вони охоплюють який-небудь фрагмент тексту, на який і розповсюджується їх дія.

У документі теги зображують у вигляді тексту, взятого в кутові дужки <…>. Відразу за відкривальною дужкою розміщається ключове слово (ім'я), яке визначає тег. Для закривальних тегів перед ім'ям ставиться "/":

<ім'я-тега>текст або інші теги</ім'я-тега>.

Якщо у тега немає вмісту, то допускається його скорочений запис:

<імя'-порожнього-тега/>.

Теги повинні вкладатися один в одний без перетину, наприклад:
<B><I>текст</I></B>.

Для уточнення або зміни дії тега він може доповнюватися атрибутами. Атрибути можуть бути тільки у відкривальному тегу, наприклад:
<H1 align="LEFT">Вступ</H1>.

Теги разом з укладеними між ними даними іноді називають контейнерами. Вмістом контейнера може бути елемент або група елементів (таблиця, абзац тексту, заголовок і т. п.).

Регістр букв в іменах тегів і атрибутів ігнорується, а у значеннях – ні.

Значення атрибута необхідно брати в лапки, використовуючи або одинарні ('80'), або подвійні лапки ("80"). Рядок у лапках не повинен містити такі ж лапки всередині себе. Доцільно використовувати подвійні лапки, оскільки око людини важко відрізняє одинарні лапки від символів, подібних до символів акцентування.

Атрибути мають бути відокремлені пропусками або незаповненими рядками.

У більшості випадків атрибути є необов'язковими, за їх відсутності браузер використовує значення за замовчуванням.

Якщо видалити з документа всі теги, то вийде звичайний текстовий документ, за смислом еквівалентний початковому.

Підстановки. У документі можуть зустрічатися символи, відсутні на клавіатурі або які мають у синтаксисі мови спеціальний сенс (наприклад, "<", ">"). Для їх введення в документ у синтаксисі HTML використовують підстановки (entities) двох видів: мнемонічні та числові.

Мнемонічні підстановки мають вигляд &мнемонічний-код, наприклад, для символу "<" підстановка виглядає &lt, для символу "&" – &amp. Мнемонічні коди визначаються у стандарті мови.

У числових підстановках замість мнемонічного коду використовується числовий код необхідного символу (код символу береться з кодування Unicode незалежно від кодування основного тексту документа). Наприклад, підстановка для кириличної букви "А" завжди має вигляд &#1040.

Головними конструкціями мови, що підлягають вивченню, є теги. Усі теги за їх призначенням і зоною дії можна розподілити на такі основні групи:

теги, що визначають структуру документа;

теги розмітки та форматування тексту (абзаци, списки, таблиці, малюнки);

теги гіпертекстових посилань і закладок;

теги форм для організації діалогу;

теги виклику програм та інших об'єктів.

1.2.1.3. Структура HTML-документа

Теги, використовувані для створення структури, приведені в табл. 1.2.

Таблиця 1.2

Теги завдання структури

Ім'я тега	Вміст тега
<HTML>.</HTML>	Контейнер гіпертекстового документа
<HEAD>.</HEAD>	Контейнер заголовка документа
<BODY>.</BODY>	Контейнер тіла документа
<TITLE>.</TITLE>	Контейнер назви документа
<IFRAME>.</IFRAME>	Плаваючий фрейм

Структура HTML-документа (або Web-сторінки) є вкладеними один в одний контейнерами. Власне, сам документ – це один великий контейнер, який починається з тега <HTML> і закінчується тегом </HTML>:

< HTML> Зміст документа </HTML>.

Контейнер HTML, або гіпертекстовий документ, складається з двох вкладених у нього контейнерів: заголовка документа (HEAD) і тіла документа (BODY).

У загальному випадку структура документа має вигляд:

```
<!DOCTYPE html>
<HTML>
 <HEAD>
  <TITLE>назва документа </TITLE>
…
 </HEAD>
 <BODY>
…
 </BODY>
</HTML>
```

Ці теги добре підкреслюють структуру документа, але обов'язковим є тільки <TITLE> </TITLE>. У разі їх відсутності структура документа буде визначена браузером за контекстом. Зазвичай сторінка займає все вікно браузера.

У версії мови HTML 5 допускається використання так званих плаваючих фреймів. Це вікно, в якому відображається довільна сторінка. Вікно може містити практично всі елементи основного вікна браузера. Місце розташування вікна визначається його місцем у тексті сторінки та розташуванням раніше виведених браузером елементів. Створюються такі вікна за допомогою тегів <IFRAME></IFRAME>. Окрім обов'язкового атрибуту SRC, можна використовувати width і height для задання ширини та висоти вікна. У наближеному вигляді контейнер цього елемента можна записати так:

```
<IFRAME SRC="URL" width="500" height ="400">
Ваш браузер не підтримує фрейми. Цю сторінку можна проглянути
<A HREF="URL">посилання</A>
</IFRAME>
```

Заголовок документа. Зазвичай найпершим рядком коду сторінки є елемент <!DOCTYPE>, який містить відомості про мову документа. Наприклад, для HTML 5 <!DOCTYPE html>. За його відсутності браузер розглядає документ за правилами попередніх версій (включається режим зворотної сумісності).

Заголовок документа (контейнер <HEAD></HEAD>) містить назву (<TITLE></TITLE>) і різні дані, які можна назвати службовими, оскільки вони звичайному користувачу візуально недоступні.

Текст назви, що міститься в тегу <TITLE> </TITLE>, на екрані не відображається, але він буде в рядку заголовка вікна браузера, в закладці, журналі.

Окрім назви, в розділі заголовка можуть розташовуватися декілька додаткових тегів. Найчастіше це: LINK, BASE, SCRIPT, META, STYLE.

Тег LINK дозволяє визначати відношення поточного документа до інших документів і ресурсів, наприклад, він може містити адресу файла таблиці стилів:

```
<LINK REL=STYLESHEET HREF="http://www.ser.com/mys.css" TYPE="text/css>
```

Елемент BASE визначає базову адресу для відносних посилань, наприклад:

```
<BASE href="http://www.acme.com">
…
<IMG SRC="icons/logo.gif">
```

У цьому випадку зображення завантажуватиметься з файла http://www.acme.com/icons/logo.gif.

За відсутності елемента BASE як база використовується місцезнаходження поточного документа.

Тег SCRIPT містить сценарій (програму на мові JavaScript або VBAScript) для створення якого-небудь ефекту на екрані.

Тег META містить службову інформацію, яка не відображається під час проглядання Web-сторінки. Усередині тега розміщені два атрибути, перший з яких містить ім'я (NAME або HTTP-EQUIV, воно описує тип даних, семантику), а другий (CONTENT) – зміст. З їх допомогою можна визначити деякі властивості документа: інформацію про автора, перелік ключових слів і т. п. Ці властивості стануть доступні пошуковим засобам і програмам обслуговування.

 Різниця між NAME і HTTP-EQUIV полягає в тому, що останній має особливе значення для програм, що реалізовують протокольні функції (протокол HTTP). Наприклад, HTTP сервери можуть використовувати ім'я, специфіковане атрибутом HTTP-EQUIV, для створення заголовка у стилі RFC 822 для відповіді HTTP. А NAME, за задумом, призначений для пошукових засобів.

Для NAME найчастіше зустрічаються такі значення:

Author – відомості про автора (особистий сайт);

Copyright – відомості про організацію (корпоративний сайт);

Description – опис сторінки (до 200 символів);

Document-state – управління індексацією (Static – не потрібний, Dynamic – можна);

Keywords – перелік ключових слів;

Generator – інформація про засіб створення;

Resource-type – стан документа (Document, Build, Classification, Creation, Formatter і ін.);

Revisit – необхідна періодичність індексації (число);

Robots – управління індексацією (index, noindex, follow та ін.);

URL – адреса основного "дзеркала" сайта.

Приклади:

<META NAME="Author" CONTENT="Molchanov">

<META NAME="keywords" CONTENT="Інтернет, HTML, WWW, керівництво, публікація, гіпертекст">

<Meta name="Description" content="Інформація про комп'ютерне залізо: тести, відгуки, коментарі експертів.">

<Meta name="URL" content= "http://www.site.ru/">

Для http-equiv можуть використовуватися такі значення:

Content-Language – мова (ru, en, fr і т. д.);

Content-Script-Type – мова скриптів;

Content-Style-Type – мова таблиці стилів;

Content-Type – тип документа з кодуванням символів (charset=..);

Expires – дата для управління кешуванням (якщо поточна менше, то можна брати з буферу);

PICS-Label – тип змісту для обмеження (sex і т. п.);

Refresh – затримка та перехід на іншій URL.

Приклади:

<Meta http-equiv="Content-language" content ="ru">

<Meta http-equiv="Content-Script-Type" content="text/javascript">

<Meta http-equiv="Content-Type" content="text/html; charset=windows-1251">

<Meta http-equiv="Expires" content="Wed, 26 Feb 1999 08:21:57 GMT">

<Meta http-equiv="Refresh" content ="3; URL=http://www.site.ru/">

У цілому не можна гарантувати, що мета-теги будуть використані засобами, для яких вони призначені, але шкоди від їх правильного використання не буде. Тому доцільно включати їх у заголовки сторінок.

У будь-якому місці HTML-документа, у тому числі в заголовку, може міститися коментар – ігнорований браузером текст. Коментар може бути поміщений у спеціальний парний тег:

<COMMENT>текст коментаря </COMMENT>

або таку конструкцію:

<!–текст коментаря->.

У цілому заголовок документа може виглядати таким чином:

```
<!–довільний коментар->
<!DOCTYPE HTML">
<COMMENT>ще один коментар</comment>
<HTML>
<HEAD>
<TITLE>Структура Web-сторінки</title>
< LINK REL=STYLESHEET HREF="http://www.myserver.com/mysheet.css" TYPE= "text/css>
<Meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<META NAME="Author" CONTENT="Molchanov">
<META NAME="Keywords" CONTENT="WWW, HTML, document, element">
</head>
```

Тіло документа. У документі дозволений тільки один тег BODY. Його атрибути діють на весь документ. За їх відсутності браузер використовує значення за замовчуванням. Тег BODY є контейнером для інших елементів.

Колір може задаватися шістнадцятирічним числом, що містить яскравість відповідних колірних компонентів (червоний, зелений, блакитний), наприклад #FFAA99, можна також використовувати ім'я кольору, наприклад: red, blue і тому подібне.

За необхідності задавання одного з атрибутів BGCOLOR, TEXT, LINK, VLINK або ALINK рекомендовано вказувати значення всіх. Інакше, наприклад, специфікований фоновий колір може збігтися з призначеним для користувача кольором для тексту за замовчуванням і текст стане невидимим.

Основні атрибути цього тега, їх призначення та значення приведені в табл. 1.3.

Таблиця 1.3

Призначення атрибутів тега BODY

Ім'я атрибута	Значення	Призначення
Background	"Шлях до файлу малюнка"	Створення фонового малюнка
Bgcolor	#RRGGBB	Колір фону
Text	#RRGGBB	Колір тексту
Link	#RRGGBB	Колір посилань
Vlink	#RRGGBB	Колір використовуваних посилань
Alink	#RRGGBB	Колір останнього посилання

Атрибути BGCOLOR і BACKGROUND можна використовувати одночасно. У цьому випадку зазвичай браузер віддає перевагу BACKGROUND, але, якщо зображення фону не можна завантажити або користувач відмовився від малюнків, буде використано значення BGCOLOR.

1.2.2. Розмітка тексту

Основними елементами, що містять текст, є абзаци (параграфи), заголовки, списки та таблиці. Відповідні теги приведені в табл. 1.4.

Таблиця 1.4

Теги для створення текстових елементів

Ім'я тега	Вміст тега
<P>.</P>	Абзац тексту
<H1>.</H1> ... <H6>.</H6>	Заголовки
<UL>.</UL> <OL>.</OL> <DL>.</DL>	Списки
<TABLE>.</TABLE>	Таблиця

Абзаци. Абзаци виділяються парним тегом <P> </P>. За замовчанням браузери, відображаючи абзаци, зазвичай відокремлюють їх один від одного додатковим інтервалом, відступ першого рядка відсутній.

Заголовки. Заголовки позначаються парними тегами H. Існує шість рівнів заголовків, які позначаються H1...H6. Заголовок рівня 1 найбільший, а кожен подальший – все менший і менший. Для абзаців і заголовків можна використовувати атрибут *align*, що забезпечує вирівнювання (left, center, right або justify).

Призначення заголовків – підкреслити структуру документа. Підсилити це можна вставкою горизонтальних ліній. Горизонтальна лінія (*horizontal rule*) задається тегом <HR> з атрибутами align=left, center, right або justify, size=товщина в пікселях, width=длина в пікселях або відсотках, color=колір. Наприклад:

```
<HR align="center" width=400 size=4 color="red">
```

По суті, *лінія* – це зафарбований прямокутник, і, якщо задати відповідну висоту та ширину її можна зробити вертикальною та використовувати в декоративних цілях. Правда, слід мати на увазі, що вона займе багато місця, а в деяких браузерах перетвориться на коло. Для позиціонування з текстом її можна помістити в таблицю.

Використання заголовків і абзаців багато в чому схоже. Для примусового переходу на новий рядок можна використовувати теги
 (примусовий перехід на новий рядок) і <WBR> (м'яке перенесення, тобто перехід на новий рядок, якщо не вдається розмістити в одному).

Тег <NOBR> </NOBR> забезпечує виведення вмісту в один рядок. Якщо рядок не поміститься на екрані, для його перегляду доведеться використовувати горизонтальну смугу прокрутки.

Для збереження авторського розміщення рядками (пропуски, табуляція, новий рядок) використовується тег <PRE> </PRE>. Крім того, є група тегів, що дозволяє локально змінювати форматування фрагментів тексту.

Всі ці засоби надають досить широкі можливості щодо структурування тексту. При цьому необхідно підкреслити, що мета розмітки – створення структури тексту, а не його оформлення.

1.2.2.1. Списки

Список відрізняється від звичайного тексту тим, що під час виводу у вікні браузера виконується автоматична нумерація або маркування його елементів. Під час внесення змін нумерація автоматично коректується. Існує декілька різновидів списків: нумеровані, ненумеровані та з визначеннями.

Ненумерований список:

```
<UL TYPE=. COMPACT>
<LH>Заголовок
<LI>пункт 1 списку
<LI>пункт 2 списки
<LI>пункт 3 списки
</UL>
```

Атрибут TYPE визначає вид маркера (disc, square, circle), COMPACT робить список компактнішим.

Нумерований список:

```
<OL TYPE= . START=. COMPACT>
<LH>Заголовок
<LI>пункт 1
<LI>пункт 2
<LI>пункт 3
</OL>
```

Атрибут TYPE визначає стиль нумерації (1 – арабські цифри, a – рядкові букви, A – прописні букви, I – великі римські цифри, i – малі римські цифри), значення атрибуту START визначає початкове значення послідовності.

Список з визначеннями:

```
<DL COMPACT>
<DT>текст пункту 1
<DD>визначення 1(виводиться зі зсувом)
<DT>текст пункту 2
<DD>визначення 2(виводиться зі зсувом)
</DL>
```

Елементами списку можуть бути інші теги. Списки можуть бути вкладеними.

У мові HTML є ще два елементи, за структурою схожі на списки <DIR><LH>.<LI>.</DIR> і <MENU><LH>.<LI>.</MENU>. Але використовуються вони вкрай рідко.

1.2.2.2. Таблиці

Таблиці зручні для структуризації даних, а багато Web-дизайнерів використовують їх також для точного розміщення елементів Web-сторінок або форм.

Таблиця в мові HTML задається за допомогою парного тега <TABLE>. Вона може містити заголовок таблиці, визначуваний парним тегом <CAPTION>, і рядки таблиці, що задаються за допомогою парних тегів <TR>. Закривальні теги </TR> можна опускати.

Кожен рядок таблиці містить комірки таблиці, які можуть відноситися до двох різних типів. Комірки в заголовках стовпців і рядків задають парним тегом <TH> (текст у них буде виділений), а звичайні комірки – парним тегом <TD>. Закривальні теги </TH> і </TD>, можна опускати. Наприклад, "порожня" таблиця з двома рядками та двома стовпцями може бути задана таким чином:

```
<TABLE>
<Caption>пуста таблиця</CAPTION>
<TR><TH><TH>
<TR><TD><TD>
</TABLE>
```

Кожна комірка може містити довільний текст, а також будь-який інший контейнер, який є допустимим в HTML-документі. Зокрема, елемент таблиці може містити вкладену таблицю або зображення.

Під час відображення таблиці у вікні браузера відбувається її автоматичне форматування з підбором розмірів комірок відповідно до обсягу розміщеної інформації та заданих атрибутів. Атрибути елементів дозволяють задавати оформлення таблиці.

Атрибути тега TABLE не обов'язкові (табл. 1.5). За замовчуванням, таблиця виводиться без рамки. Як правило, розмір елементів таблиці встановлюється автоматично, щоб щонайкраще розмістити вміст, проте можна встановити ширину таблиці за допомогою атрибуту WIDTH (у пікселях або відсотках). Атрибути BORDER, CELLSPACING і CELLPADDING надають додаткові можливості для контролю за зовнішнім виглядом таблиці.

Заголовок розміщується над або під таблицею залежно від значення атрибуту ALIGN=(TOP або BOTTOM).

Як фон можна використовувати малюнок, задавши атрибут BACK-GROUND="шлях до файла".

Таблиця 1.5

Призначення атрибутів, що визначають вид таблиці

Атрибут	Елемент	Призначення
ALIGN	Таблиця, заголовок, рядок, комірка	Вирівнювання таблиці по горизонталі; вирівнювання даних по горизонталі; розміщення заголовка над або під таблицею
VALIGN	Рядок, комірка	Вирівнювання по вертикалі
WIDTH	Таблиця, комірка	Ширина
HEIGHT	Комірка	Висота
COLSPAN	Комірка	Протяжність у декілька стовпців
ROWSPAN	Комірка	Протяжність у декілька рядків
BGCOLOR	Таблиця, комірка	Колір фону
BACKGROUND	Таблиця, комірка	Фоновий рисунок
CELLSPACING	Таблиця	Зазор між комірками
CELLPADDING	Таблиця	Зазор між вмістом комірки за її межею
BORDER	Таблиця, комірка	Відображення меж комірок і зовнішньої рамки таблиці

Шляхом задання атрибутів COLSPAN і ROWSPAN можна об'єднувати розташовані послідовно комірки (ці атрибути задаються в початковій комірці й указують, яку кількість комірок потрібно об'єднати).

Таблиці, як правило, показуються на екрані "піднесеними" над поверхнею сторінки, а комірки – "втиснуті" в тіло таблиці. Комірки виділяються окантовкою тільки якщо в них є вміст. Якщо вміст комірки складається тільки з пропусків, комірка вважається порожньою, за винятком випадків, коли в ній є навіть один об'єкт   (пропуск).

У таблиці за допомогою спеціальних тегів можна виділяти групи рядків і стовпців для їх окремого форматування. Такими тегами для рядків є <THEAD>, <TFOOT>, <TBODY>. Закривальний тег можна опускати. У таблиці можна указувати по одному елементу THEAD і TFOOT, але декілька елементів TBODY. Послідовність завдання елементів тільки така: THEAD, TFOOT, TBODY, але в таблиці TFOOT виявиться найнижчим.

<COL> і <COLGROUP> дозволяють призначати властивості окремим колонкам і групам колонок. Число колонок, на які розповсюджується властивість, задається атрибутом SPAN. Елементи COL розміщуються як самостійно, так і усередині COLGROUP. Якщо атрибут SPAN використаний і в COL і в COLGROUP, то значення в COLGROUP ігнорується.

Елемент таблиці може містити інші контейнери, включаючи форми й інші таблиці.

Окрім самостійного призначення – подання даних у табличній формі, таблиці використовуються для управління розміщенням матеріалу на сторінці. Такі таблиці найчастіше створюються без рамок і заливки.

1.2.2.3. Теги форматування тексту

Теги форматування тексту можна умовно розподілити на дві групи: теги логічного (табл. 1.6) і фізичного (табл. 1.7) форматування.

За допомогою перших вказується роль текстового фрагмента, наприклад, велика значущість порівняно із звичайним текстом або те, що даний фрагмент є цитатою. Друга група забезпечує задання конкретних параметрів, наприклад, вид шрифту або вид спеціального підкреслення. Перевага повинна віддаватися логічному форматуванню.

Елементи розмітки не повинні перекриватися, але вони можуть бути вкладеними. Слід підкреслити, що вид відформатованих цими тегами елементів в різних браузерах буде відрізнятися.

Таблиця 1.6

Теги логічного форматування тексту

Теги	Значення
EM	Виділення
STRONG	Ще сильніше виділення
DFN	Означає, що цей термін має визначення
VAR	Частина тексту (звичайне слово) є змінною
CITE	Назва цитованої книги або статті
BLOCKQUOTE	Цитування
CODE	Код програми або його еквівалент (наприклад, HTML)
SAMP	Службові повідомлення комп'ютера
KBD	Текст, який повинен друкуватися на клавіатурі користувача

 Крім цих тегів є ще два (div і span). Тег використовується для групування таких елементів, як заголовок, декілька абзаців, списків в один блок, який надалі можна позиціонувати або форматувати як єдине ціле. Тег застосовується для виділення внутрішніх (inline) елементів, таких як окремі слова і фрази, які перебувають в межах абзацу тексту або заголовка. Ці теги не несуть конкретного форматування і були введені для подальшого використання з таблицями стилів.

Таблиця 1.7

Теги фізичного форматування тексту

Теги	Значення
TT	"Телетайпний" текст, тобто текст одного розміру
I	Курсив
B	Жирний
U	Підкреслення
STRIKE	Закреслений текст
BIG	Великий шрифт
SMALL	Малий шрифт
SUB	Підрядковий текст
SUP	Надрядковий текст
FONT	Установка розміру та кольору шрифту
BASEFONT	Базовий розмір шрифту

Усі теги, окрім BASEFONT, парні. <BASEFONT SIZE=n>де n=1-7, задає в точці його використання розмір шрифту, щодо якого далі можна змінювати розміри. Тег FONT може містити три атрибути SIZE (розмір), COLOR (колір) і FACE (вигляд). Розмір може задаватися як абсолютний (число без знаку, 1 – 7), так і щодо базового (число зі знаком). Елементи FONT і BASEFONT доцільні для задання розмірів шрифту. Проте не варто без особливої необхідності задавати розмір шрифту, краще використовувати виділення фраз, інші структурні елементи. Це дозволить користувачам, якщо їм не подобаються розміри шрифту, визначати шрифти на свій розсуд. У цілому слід уникати надмірного використання фізичної розмітки.

Приклад 1. Сторінка, яка демонструє структуру документа, використання мета тегів і розміщення тексту в різних елементах. Відображення сторінки у браузері надано на рис. 1.1.

```
<HTML>
<HEAD>
        <TITLE>Структура Web-сторінки</TITLE>
        <LINK REL=STYLESHEET HREF="http://www.myserv.com/mysheet.css" TYPE=
"text/css"/>
        <META NAME="Author" CONTENT="Molchanov"/>
        <META NAME="Keywords" CONTENT="WWW, HTML, document, element"/>
        <META http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</HEAD>
<BODY>
<HR width=100% size=4 color=black>
<table border=0 width=100%>
        <TR><TD width=30%>абзаци <TD width=40%>списки<TD width=30% >фрейм
        <TR><TD bgcolor=red><P> Приклад тексту</P><H1> Приклад тексту </H1>
                <TD BACKGROUND="fon_1.jpg"><H3>Список маркований</h3>
                    <UL TYPE=disc >
                        <LH>Заголовок
                        <LI>пункт 1 списку
                        <LI>пункт 2 списку
                        <LI>пункт 3 списку
                    </UL>
                 <TD bgcolor=yellow><IFRAME SRC="p1.htm"
                                        scrolling="no" frameborder=1
                                        width=100%>
                                        </IFRAME>
</table>
</BODY>
</HTML>
```

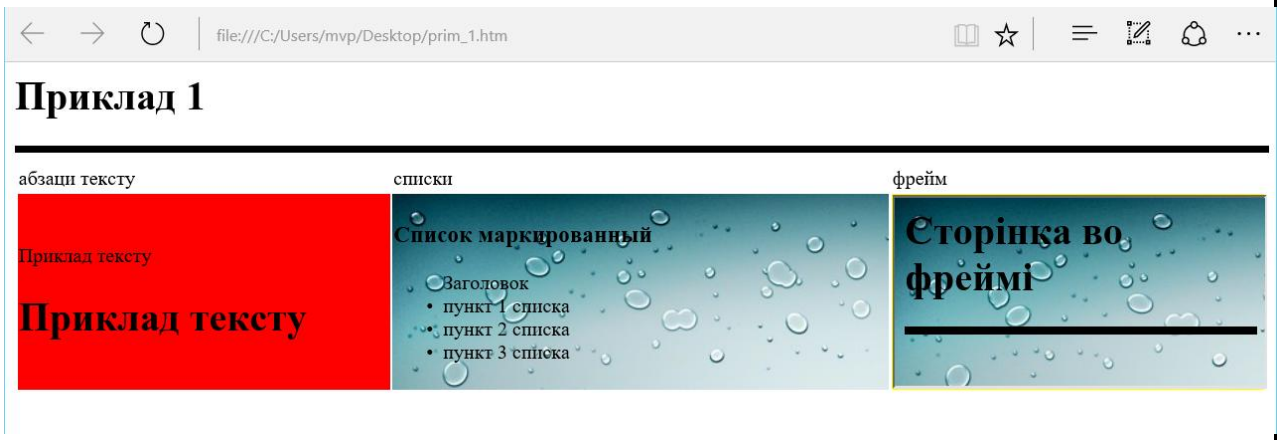

Рис. 1.1. **Вид сторінки у вікні браузера**

Теги семантичної розмітки. У мові HTML 5 на перевагу до попередніх версій була введена ціла група тегів, що дозволяють групувати елементи WEB-сторінок у блоки, на зразок тегів DIV. Це ARTICLE, ASIDE, FOOTER, HEADER і багато інших. Візуально використання цих тегів без зв'язку з таблицями стилів ніяк не проявляється, як і для DIV. Їх основне призначення – повідомляти засобу (програмі), що виконує аналіз розміченого тексту, про смисловий (семантичний) зміст відповідного контейнера. Завдяки використанню цих тегів поліпшується читаність коду та стає більш детальною його структуризація з метою подальшого форматування.

1.2.2.4. Текст і рисунки на WEB-сторінках

Графіка в більшості випадків є невід'ємною частиною Web-документів. Сьогодні для графічних елементів Web-сторінок використовують декілька форматів: GIF, JPEG, PNG, SVG. Три перших є растровими, останній – векторний.

Усі браузери останніх версій здатні розпізнавати та відображати файли цих форматів. Для підготовки зображень можна використовувати відповідний графічний редактор, а також створювати зображення за допомогою фотоапарата та сканера.

Зображення зберігаються в окремих файлах, але відображаються як елементи Web-сторінок. Для вставки зображення використовується тег <IMG>, який повинен містити обов'язковий атрибут SRC, що задає адресу (URL) файла у відносній або абсолютній формі:

```
<IMG SRC="picture1.gif">
<IMG SRC="http://www.wwroot.picture1.gif">
```

Для відображення рисунка браузер за замовчуванням використовує його реальні розміри. Якщо рисунок необхідно зменшити, застосовують атрибути WIDTH і HEIGHT і задають ширину та висоту (у пікселях), відповідно. Якщо ці параметри задані, то браузер може визначити, яке місце треба виділити для зображення ще до того, як воно буде завантажене. Це дещо прискорює відображення сторінки, тому доцільно завжди задавати ці атрибути. Наприклад, <IMG SRC="picture2.jpg" WIDTH="100" HEIGHT="40">.

Зовнішній вигляд Web-сторінки залежить від того, як саме рисунок розташовується на ній. Рисунок може задаватися як деякий нестандарт-

ний символ, що знаходиться всередині якогось абзацу. Щоб зображення відображалося автономно, його включають в окремий абзац.

Для зображення, яке включене в рядок, можна задати режим взаємодії з текстом за допомогою атрибута ALIGN. Атрибут може набувати три значення: BOTTOM (нижня межа зображення поєднується з підставою текстового рядка), MIDDLE (середина зображення поєднується з серединою текстового рядка) і TOP (верхня межа зображення вирівнюється за верхнім обрізом текстового рядка). Наприклад: <IMG SRC= "picture3.gif" ALIGN="BOTTOM">.

Проте переважніше використання "плаваючого" зображення, обтічного текстом, що також досягається використанням таких значень атрибуту ALIGN: LEFT (зображення розміщується з лівого краю сторінки, а подальший текст – праворуч від нього), RIGHT (зображення розміщується з правого краю сторінки, а подальший текст – зліва від нього). У цьому випадку рекомендовано поміщати тег <IMG> в самий початок відповідного абзацу.

Нормальний режим обтікання вимагає, щоб між текстом і зображенням залишався деякий проміжок. Задати величину цього проміжку можна за допомогою атрибутів HSPACE (по горизонталі) і VSPACE (по вертикалі). Розміри задаються в пікселях. Для точнішого позиціонування зображення можна помістити в елемент таблиці.

Створюючи ілюстровані сторінки, не слід забувати, що не всі зможуть побачити ці ілюстрації. Користувачів, які не мають адекватного засобу перегляду, можна ознайомити зі змістом ілюстрацій за допомогою альтернативного тексту. Альтернативний текст задається як значення атрибута ALT і відображається замість картинки, якщо вона за якихнебудь причин не може бути виведена.

Як вже зазначалося, зображення можуть виводиться не тільки з допомогою тега img, а також в якості фону сторінки, таблиці або осередка. Шлях до файла задається за допомогою атрибута BACKGROUND (BACKGROUND = *шлях до файла*).

Існує ряд елементів, працездатність яких у масових браузерах не гарантується. Причин цьому декілька. По-перше, такі елементи з'являються у результаті змін, що вносяться до мови. З'являються нові, ефективніші засоби, а старі – ігноруються. По-друге, є ряд елементів, що створюють малопомітні невиразні ефекти. Про них просто починають забувати. І, по-третє, є елементи, які не включені у стандарт, але

підтримуються окремими виробниками браузерів. Їх використання не виправдане, оскільки не всі їх побачать. Прикладами можуть слугувати теги BLINK, DIR, MENU, BANNER і ряд інших.

1.2.3. Створення елементів навігації

Що таке інтерфейс, якщо говорити про цей термін стосовно Web-сторінок? Сюди слід віднести все, що використовується для взаємодії, включаючи елементи управління браузера, елементи навігації для переходів на інші сторінки, а також всілякі кнопки, перемикачі й інші елементи, з якими можуть бути зв'язані обробники подій. У цілому, майже будь-який елемент сторінки можна зробити керівним. Ми розглянемо засоби створення посилань і форм.

1.2.3.1. Відносні й абсолютні посилання

Механізм посилань забезпечує основну властивість, що відрізняє гіпертекстовий документ від звичайного. Посилання дозволяють клацанням завантажувати у вікно як абсолютно інший документ, так і іншу частину поточного документа. Для цього використовується парний тег <A> </A>. Усередині нього розміщують елемент, на зображенні якого повинне виконуватися клацання для переходу. Тег містить обов'язковий атрибут HREF, значенням якого є URL або адреса ресурсу іншого типу (електронної пошти, FTP та ін.).

```
<A HREF="http://www.site.com/index.htm"> клацни Тут</a>
<A HREF="mailto:victor@molchanov.eu.org"> пошта</A>
<A HREF="file:///D:/root/tur.gif"> подивитися рисунок</A>
```

За першим тегом браузер створить у вікні текст "клацни Тут", що є гіперпосиланням. Після клацання мишею відбудеться перехід на нову сторінку. Після клацання на другому посиланні відкриється діалог для створення та відправки пошти.

Видима частина гіперпосилання може бути рисунком:

```
<A HREF="http://www.site.com/index.htm"> <IMG SRC="picture1.gif"></A>.
```

Посилання не можуть знаходитися одне всередині іншого.

URL може бути заданий в абсолютній формі, тобто починатися зі вказівки протоколу й адреси Web-вузла (http://www.meta.com/page1.htm). З використанням відносної форми пошук ресурсу відбудеться відносно папки, в якій розташована сторінка, або значення тегу BASE. Наприклад, якщо мова йде про сторінку http://www.meta.com/page1.htm і тегу BASE немає, то базовою буде адреса – "http://www.meta.com/", а як відносну адресу сторінки можна використовувати page1.htm. Для переходу буде сформована адреса http://www.meta.com/page1.htm.

Для задавання переходів за ієрархією папок можна використовувати спеціальне ім'я .. (перехід на рівень вгору), наприклад: ..\root\f1.htm.

Внутрішнє посилання зберігає свою працездатність у разі зміни місця зберігання сторінок сайта як цілого (наприклад, у результаті перенесення папки з файлами на інший сервер). Тому за потенційної нагоди такої події слід відмовлятися від повного задавання адреси в гіперпосиланнях.

Для зміни базової адреси допускається використання тільки одного елемента BASE. Тут базовою адресою вважається послідовність символів до останнього "слеша". Таким чином, за задавання <BASE HREF= "http://www.hut.fi/3dr"> за базову адресу вважатиметься http://www.hut.fi/.

Тег <A> може містити декілька необов'язкових атрибутів (найчастіше використовуються TARGET і TITLE). Перший указує вікно, в якому буде відкрита сторінка. Значенням атрибуту є ім'я вікна (для фрейма те, яке указувалося в атрибуті NAME тега FRAME) або спеціальне значення:

_self – відкрити у поточному вікні;

_parent, _top – відкрити у вікні без фреймів;

_blank, new – відкрити в новому вікні.

Значенням атрибуту TITLE є текст, який буде виведений наведенням курсора на посилання.

Посилання й якорі. Посилання може бути виконане і на певне місце всередині сторінки. Для цього відповідне місце позначають за допомогою того ж тега <A> </A>, але з атрибутом NAME. Значення цього атрибута – довільна послідовність латинських літер і цифр без пропусків (ім'я), що розглядається як метка. Такий тег у літературі називають *анкер* або *якір*. Для посилання на мітку її ім'я указується як значення параметра HREF, але перед ім'ям має бути символ #. Наприклад, можна створити електронну книжку із змістом на початку, використовуючи такий шаблон:

```
<UL>
<LI><A HREF="#gl1">ГЛАВА 1</A>
<LI><A HREF="#gl2">ГЛАВА 2</A>
<LI><A HREF="#gl3">ГЛАВА 3</A>
</UL>
.
<A NANE="gl1"></A>
.
<A NANE="gl2"></A>
.
<A NANE="gl3"></A>
```

Така конструкція дає можливість зручно переміщатися по сторінці з великою кількістю тексту.

1.2.3.2. Карти посилань

Під час інтерпретації тегів IMG (створення зображень на Web-сторінках) браузер підтримує для кожного елемента систему координат з можливістю визначення координат курсора в межах зображення. Завдяки цьому є можливість відправки координат курсору на сервер або пов'язання з ними гіперпосилань. Характер використання задається атрибутами ISMAP (відправити серверу), USEMAP (зв'язати посилання).

Для реалізації можливості скріплення посилань з різними частинами зображення спільно з тегом IMG використовується тег MAP, який задає конкретні посилання. Ім'я цього тега і є значенням атрибуту USEMAP:

```
<IMG SRC="URL" USEMAP="#им'я_карти"
.
<MAP NAME="им'я_карти">
.
</MAP>
```

Відмітимо, що на сервері координати теж можуть використовуватися для виконання переходу, залежно від того, на якій частині зображення виконано клацання.

Усередині тега MAP знаходиться група тегів AREA (значення та сенс атрибутів наведені в табл. 1.8), кожен з яких визначає одну область зображення та пов'язане з нею посилання.

Таблиця 1.8

Атрибути тегу AREA

Ім'я атрибута	Можливі значення	Сенс	Примітки
SHAPE	RECT, CIRCLE, POLY	Контури області	За замовчуванням RECT
COORDS	Рядок	Координати області	Обов'язковий
HREF	URL	Адреса документа	Діє як гіпертекстовий зв'язок
NOHREF	NOHREF	Область не використовується	За необхідністю заборонити перехід
ALT	Рядок	Текстовий опис області	Необов'язковий

Область може мати форму прямокутника (RECT), кола (CIRCLE) або багатокутника довільної конфігурації (POLY).

Значення координат (x і y) задаються в пікселях від верхнього лівого кута зображення (значення y зростає вниз). Координати (або одна з них) можуть бути також задані у відсотках, зі знаком відсотків, що додається до числа. Відсотки беруться відповідно до ширини або висоти зображення.

Коло задається координатами центра та радіусом, прямокутник – двох кутів, багатокутник – всіх вершин. Наприклад:

```
<HTML>
<HEAD>
<TITLE>Карта посилань</TITLE>
</HEAD>
<BODY BGCOLOR=Silver TEXT="#848484" LINK=RED VLINK=PURPLE
ALINK=GREEN>
<H1 ALIGN="center"> Приклад використання карт посилань</H1>
<IMG SRC="map.gif" BORDER=0 USEMAP="#map1">
<MAP NAME="map1">
<AREA HREF="primer1.htm" ALT="Текст перший" SHAPE=CIRCLE
COORDS="X,Y,R">
<AREA HREF="primer2.htm" ALT="Текст другий" SHAPE=POLY
COORDS="X1,Y1,X2,Y2,X3,Y3">
</map>
 </BODY>
</HTML>
```

Зрозуміло, що замість x1, x2 і так далі повинні стояти конкретні значення.

Із зображеннями карт зручно працювати у графічному редакторі, що підтримує координатну сітку (у простішому випадку це Paint). Курсором виділяються потрібні крапки, і в рядку стану прочитуються значення координат.

Карти можна використовувати для створення оригінальних меню, наприклад, у вигляді смуг уздовж вікна або з довільною геометрією, з написами, виконаними власними шрифтами.

1.2.3.3. Форми

Не дивлячись на те, що керівним можна зробити практично будь-який елемент сторінки, традиційно такими елементами є кнопки, поля і т. д. Контейнером для таких елементів слугує тег FORM.

На Web-сторінці форма є набором елементів управління, зокрема декількома видами полів: однорядкові та багаторядкові вікна для введення тексту; групи радіокнопок; квадрати, що позначаються; меню та багато інших. Користувач у процесі роботи з Web-сторінкою заповнює форму й активізує деяку процедуру обробки.

Сама обробка може полягати у відправці даних на один з серверів мережі або за поштовою адресою. На сервері дані, відправлені з форми, приймаються й обробляються спеціальними програмами. Дані можуть бути оброблені і за допомогою програм, вбудованих у Web-сторінку.

Тег FORM має зміст, усередині якого містяться теги, що створюють елементи (INPUT, SELECT і TEXTAREA). Можливі атрибути тегу FORM наведені в табл. 1.9.

Таблиця 1.9

Атрибути тегу FORM

Ім'я атрибута	Значення	Призначення
action	URL або адреса E-mail	Указує на ресурс, який оброблює форму
method	GET або POST, за замовчуванням – GET	Метод пересилання вмісту
enctype	За замовчуванням – "application/x-www-form-urlencoded"	Механізм кодування форми
name	Рядок символів	Ім'я елемента

Обов'язковим є тільки action. Наприклад, для відсилання вмісту форми електронною поштою слід задати action="mailto:foo@bar.com", для відсилання на сервер – action="http://www.acme.com/cgi-bin/register.pl".

Теги fieldset, legend, label забезпечують можливості щодо додаткового оформлення форм (угрупування елементів, назва для групи та назви елементів, відповідно).

Тег INPUT можна використовувати для створення безлічі полів форм: однорядкових текстових полів; полів для введення паролів; квадратів, що позначаються; радіокнопок; кнопок запуску і скидання; прихованих полів; полів завантаження файлів і зображень-кнопок. Елементи, що створюються тегом INPUT, не можуть містити в собі інших елементів. Призначення основних атрибутів тега приведені в табл. 1.10.

Таблиця 1.10

Атрибути тегу INPUT

Ім'я атрибута	Значення	Призначення
Type	text	Однорядкове текстове поле (за замовчуванням)
	password	Поле введення пароля
	checkbox	Перемикач (квадрат, що позначається)
	radio	Перемикач (радіокнопка)
	submit	Кнопка для відправки даних
	button	Кнопка для генерації події
	reset	Кнопка для приведення полів у початковий стан
	color	Створює середовище для вибору кольору
	date	Створює середовище введення дати (календар)
	email	Створює поле для введення e-mail
	range	Створює повзунок
	url	Створює поле для введення url
	file	Створює середовище для вибору файла
size	число	Розмір поля в символах
name	рядок символів	Ім'я елемента
value	рядок символів	Напис, виведений за початковим завантаженням
autocomplete	on або off	Включає / відключає автозаповнення
pattern	регулярний вираз	Шаблон для перевірки правильності введення
placeholder	рядок символів	Рядок, який міститься в текстовому полі
required	required	Поле, яке обов'язково має бути заповненим

Атрибут name використовується для визначення імені, яке буде привласнене вмісту поля під час передавання обробникові. Атрибут має бути присутнім у всіх елементах, дані з яких відправлятимуться.

Значення type=text (використовується за замовчуванням) визначає однорядкове текстове поле. Видимий розмір може встановлюватися атрибутом size, наприклад, size=40 для поля шириною 40 символів. Користувачі можуть вводити і довші рядки. Верхню межу числа символів в рядку, що вводиться, можна встановити атрибутом maxlength. Атрибут value можна використовувати для задавання рядка, який буде показаний в полі за початкового завантаження документа. Наприклад:

```
<input type=text size=40 name=user value="your name">
```

Значення type=password аналогічне type=text, проте замість символів, що вводяться з клавіатури, показує на екрані символ-маску (наприклад, *), що дозволяє приховати текст від сторонніх очей шляхом введення пароля. Як і у звичайних текстових полях, можна використовувати атрибути size і maxlength для управління видимою та максимально допустимою довжиною. Наприклад:

```
<input type=password size=12 name=pw>
```

Значення type=checkbox використовується для створення логічних елементів, передавання значення яких визначається станом: відмічений – не відмічений (квадрати, що позначаються). Таких квадратів може бути декілька. У останньому випадку використовуються декілька пар атрибутів *name – value* з однаковим атрибутом name та різними атрибутами value. Кожен відмічений квадрат генерує власну пару ім'я/значення у складі переданих даних, навіть якщо це приводить до появи імен-двійників. Атрибут checked використовується для ініціалізації квадрата з позначкою. Наприклад:

```
<input type=checkbox checked name=uscitizen value=R1>
<input type=checkbox name=uscitizen value=R2>
```

Значення type=radio використовується для створення елементів, які задають тільки одне значення з фіксованого набору альтернатив (радіокнопки). Кожна радіокнопка у групі повинна мати один і той же атрибут name та різні value. Пара ім'я/значення генерується тільки відміченою радіокнопкою. Одна радіокнопка в кожній групі повинна ініціалізуватися

відміченою радіокнопкою шляхом використання атрибута checked. Наприклад:

```
<input type=radio checked name=age value="16-21">
<input type=radio name=age value="21-35">
```

Значення type=submit визначає кнопку, на яку користувач натискає, щоб передати вміст форми на обробку. Текст на кнопці встановлюється з атрибуту value. Якщо вказаний атрибут name, то пара ім'я/значення, створена кнопкою, буде включена в передаванні дані. У одну форму можна включати декілька кнопок запуску.

Значення type=reset створює кнопку для приведення всіх полів форми в початковий стан. Наприклад: <input type=reset value="СБРОС">.

Для позиціонування елементів у вікні можна використовувати теги <P>, <TABLE>, <PRE>. Наприклад;

```
<FORM ACTION="mailto:victor@molchanov.eu.org" METHOD=POST>
<P>введіть ім'я
<input type=text size=20 name=user value="your name">
<p>введіть пароль
<input type=password size=12 name=pw>
<TABLE CELLSPACING=40>
<TR><TD><P>задайте режим доступу
<P><input type=checkbox name=uscitizen value=r1>режим 1
<P><input type=checkbox checked name=uscitizen value=r2 >режим 2
<TD><P>задайте діапазон
<P><input type=radio checked name=age value="16-21">16-21
<P> <input type=radio name=age value="21-35">21-35
</TABLE>
<P><input type=reset value="СБРОС">
<input type=submit value="ОТПРАВКА">
</FORM>
```

Тег SELECT використовується для створення меню, з яких можна вибрати один або декілька елементів. Елементи SELECT містять один або декілька тегів OPTION, що визначають пункти меню. Меню з вибором одного елемента зазвичай показуються на екрані як випадні, наприклад:

```
<SELECT name="evaluation">
<OPTION value=1>Дуже убого
<OPTION value=2>Бідно
<OPTION value=3>Середньо
<OPTION value=4>Досить добре
<OPTION value=5>Прекрасно
</SELECT>
```

Кожен вибраний пункт передається обробникові у вигляді пари ім'я/значення. Тег OPTION може мати атрибут selected, присутність цього атрибута пункт меню виявляється відміченим за початкового завантаження документа. Включати цей атрибут більш ніж в один пункт меню, що допускає вибір тільки одного пункту, – помилка. Атрибут value встановлює значення, яке буде передане обробникові з ім'ям, певним атрибутом name елемента SELECT.

Тег TEXTAREA використовується для створення багаторядкових полів введення тексту. Для елементів TEXTAREA обов'язкові відкривальні та закривальні мітки. Використовується для введення та відсилання довгих текстів. Текст, поміщений після відкривального тега, показується у вікні за первинного завантаження документа. Число символів у рядку та кількість рядків задаються атрибутами cols і rows. Наприклад:

```
<TEXTAREA name=Comments rows=5 COLS=72>
Тут можна ввести/запроваджувати докладніший коментар:
</TEXTAREA>
```

Нижче приведений фрагмент HTML-документа, що містить меню:

```
<FORM ACTION="mailto:victor@molchanov.eu.org" METHOD=POST>
Висловіть свою думку про якість цього документа:
<SELECT name="evaluation">
<OPTION value=1>Убого
<OPTION value=2>Бідно
<OPTION value=3>Середньо
<OPTION value=4>Добре
<OPTION value=5 selected>Відмінно
</SELECT>
<TEXTAREA name=Comments rows=5 COLS=72>
Тут можете ввести докладніший коментар
</TEXTAREA>
<P>Введіть своє прізвище
<INPUT TYPE=TEXT NAME=fio size=30>
<INPUT TYPE=SUBMIT VALUE=Send>
</FORM>
```

Інші атрибути дозволяють зробити форму більш зручною для користувача.

При цьому можуть створюватися рамки, пояснюючі написи, групуватися елементи і виконуватися інше оформлення. Можуть виводиться підказки і дані про невірне заповненні полів.

Приклад 2. Сторінка, яка демонструє можливості щодо створення форм. Відображення сторінки у браузері надано на рис. 1.2.

```
<HTML>
<HEAD>
        <TITLE>Створення форм</TITLE>
        <META NAME="Author" CONTENT="Molchanov"/>
        <META NAME="Keywords" CONTENT="WWW, HTML, document, element"/>
        <META http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</HEAD>

<BODY>
<form name="F1" ACTION="mailto:victor@molchanov.eu.org" METHOD="POST">
<fieldset>
<legend>Висловіть свою думку про якість цього документа:</legend>
 <label>виберіть варіант відповіді
<SELECT name="evaluation">
  <OPTION value=0>
  <OPTION value=1>Мені не сподобалося
  <OPTION value=2>Середньо
  <OPTION value=3>Досить добре
  <OPTION value=4>Відмінно
</SELECT>
</label><br>
<TEXTAREA name="Comments" rows=5 COLS=72  placeholder="Тут можете ввести більш докладний коментар">
</TEXTAREA><br>
 <label>введіть прізвище
<input type="text" required="required" name="TX1" size="35">
</label><br>
<label>введіть дату
<input type="date" name="TX1" size="35">
</label>
   <input type="reset" value="скидання">
   <input type="submit" value="відправити">
</fieldset>
</form>
</BODY>
</HTML>
```

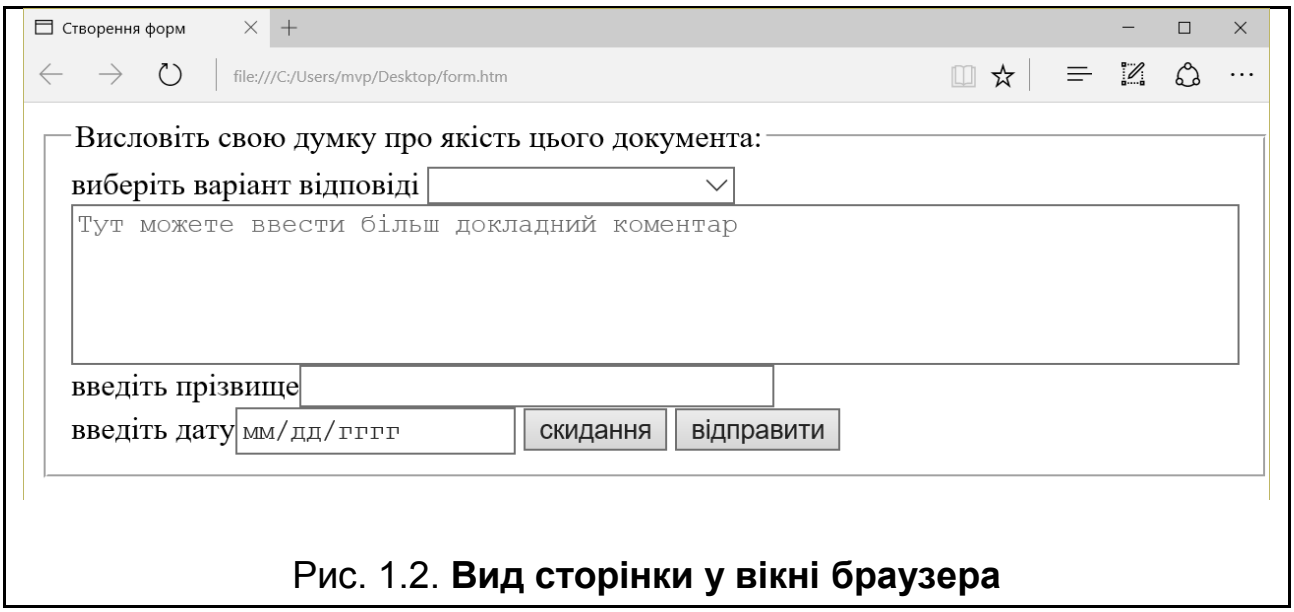

Рис. 1.2. **Вид сторінки у вікні браузера**

Наприкінці ще раз підкреслимо, що для взаємодії з формою потрібна програма на сервері або клієнтові. Крім того, для забезпечення додаткових дій з формами (різні ефекти і застосування) вони доповнюються скриптами.

1.2.4. Розміщення об'єктів

У процесі виконання розмітки тексту засобами HTML під об'єктами розуміються елементи, які знаходяться в окремих файлах у власному форматі. За такого підходу об'єктами будуть рисунки, різні мультимедійні елементи (аудіо та відео), програми.

Рисунки та карти посилань уже були розглянуті. Зважаючи на масове розповсюдження та простоту вбудовування, їх об'єктні особливості не підкреслювалися. А для вбудовування інших об'єктів є спеціальні теги.

Об'єкти, що підлягають відображенню на Web-сторінках, створюються окремо та зберігаються у файлах власних форматів. Формат визначається або за розширенням .gif, .jpg, .mpg, .class і так далі, або за спеціальним кодом content type="тип вмісту", що вставляються у початковий код документа, де тип вмісту може набувати значень: text/html, image/gif, video/quicktime і т. д.

Частина об'єктів інтерпретується безпосередньо більшістю браузерів. Це рисунки, програмні коди (скрипти та аплети). Інша частина, а це більшість мультимедійних файлів, вбудованими засобами відображені бути не можуть і вимагають додаткових модулів розширення або звернення до відповідних зовнішніх програм.

Додаткові модулі розширення, які називають плагінами (plug-ins), розробляються сторонніми фірмами та можуть бути підключені до браузера. Підключення виконується один раз.

Якщо ж браузер не може обробити отримані дані сам або за допомогою вбудованих модулів, то може бути запущений зовнішній додаток, що зазвичай указується в налаштуваннях. Зовнішні додатки, наприклад RealAudio, обробляють вміст вже у своєму власному вікні.

Теги та передавання параметрів. До тегів, орієнтованих на вбудовування об'єктів, можна віднести: OBJECT, EMBED, APPLET, SCRIPT, BGSOUND, AUDIO, VIDEO.

Тег OBJECT є універсальним і дозволяє вставляти в документ графічне зображення, відеокліп, аплет JAVA або елемент управління ACTIVEX, може створювати вікно. Хоча найчастіше він використовується для вбудовування елементів ACTIVEX і аплетів.


Основні атрибути тега OBJECT:

Title – текст спливаючої підказки;

Declare – об'єкт завантажується, але не активізується;

Classid – ідентифікатор класу або екземпляр об'єкта;

Codebase – посилання на об'єкт або базовий URL;

Data – URL для завантаження необхідних даних або об'єкта;

Type – тип вмісту (MIME-об'єкта) відповідно до data (може набувати значень image/gif, image/jpeg, video/mpeg, video/mp4, audio/mpeg, audio/mp4, text/css, text/html, text/xml і т. д.);

Codetype – тип даних відповідно до атрибута CLASSID;

Standby – текст, що виводиться в процесі завантаження об'єкта;

Align – режим вирівнювання об'єкта щодо рядка поточного тексту або як окремого елемента (text-top, middle, textmiddle, left і так далі);

width, height – ширина і висота об'єкта у вікні браузера;

alt – альтернативний вміст, який буде візуалізовано у випадку, якщо користувач не може або не хоче відобразити об'єкт.

Усередині тега <OBJECT> можуть знаходиться інші елементи (наприклад, текст або малюнок, інші об'єкти), але вони інтерпретуються тільки за неможливості активізувати об'єкт. Завдяки цьому можна задавати декілька альтернативних елементів для виводу в одному місці.

У ряді випадків для відображення об'єкта потрібно передати йому які-небудь параметри. Для цього використовується спеціальний тег <PARAM>, що розміщується усередині <OBJECT>. Тег містить три атрибути:

name – ім'я параметра;
value – значення параметра;
valuetype – тип параметра (data, object, ref).
Значеннями параметра name можуть бути:
"FileName" – URL;
"AutoStart" – {true, false};
"ShowDisplay" – {true, false};
"AnimationAtStart" – {true, false};
"TransparentAtStart" – {true, false};
"ShowStatusBar" – {true, false};
"src" – URL;
"hidden" – {true, false}.

Під час використання тега необхідно враховувати особливості вбудовуваних об'єктів. Наприклад, можна вбудувати малюнок (альтернатива тегу IMG):

```
<OBJECT data="http://www.arcus.lv/dimas/roma.jpg"
type="image/jpeg" width=150 height=150></OBJECT>
```

або відкрити вікно з іншою сторінкою (альтернатива фреймам):

```
<OBJECT data="http://www.arcus.lv/dimas/index.html"
type="text/html" width=200 height=150></OBJECT>
```

Конструкція з параметрами може виглядати так:

```
<OBJECT TYPE="audio/mpeg">
 <param name="FileName" value="all\dam.mp3">
  . . .
</object>
```

Ще одним тегом для вбудовування об'єктів, орієнтованим в основному на мультимедійні об'єкти, є EMBED.


Тег EMBED може містити такі атрибути:
height – висота;
width – ширина;
autostart – можливість запуску під час завантаження (true, false);
hidden – приховати елемент управління;
loop – повторення (true, false);
hidden – приховати панель управління (true, false);

src – URL мультимедійного файла;
pluginspage – URL плагіна;
bgcolor – фон об'єкта;
type – тип мультимедійного файла;
quality – якість мультимедійного файла;
alt – альтернативний зміст.

Наприклад, для вбудовування флеш-файла тег може виглядати так:

```
<EMBED src="file.swf" menu=true quality=high bgcolor=#000066
WIDTH=760
HEIGHT=410
TYPE="application/x-shockwave-flash"
PLUGINSPAGE="http://www.macromedia.com/shockwave/download/index.cgi?
P1_Prod_Version=ShockwaveFlash">
</EMBED>
```

Тег BGSOUND слугує для створення фонового звуку, але вважається застарілим.

Наприкінці необхідно зробити одне зауваження: більшість мультимедійних файлів будуть відкриті браузером після клацання на посиланні, яке містіть URL мультимедійного файла. Браузер запустить зовнішній додаток, відповідний формату файла. У деяких варіантах можлива видача запиту: відкрити або зберегти. Це буде як перехід на іншу сторінку, а приведені вище теги дозволяють саме вбудувати, тобто пов'язати об'єкт із змістом відображуваної сторінки в межах вікна браузера.

Теги audio і video були введені в стандарт HTML 5 з метою спростити вбудовування відповідних об'єктів (звук і відео) в WEB-сторінки. Скорочено число параметрів у тегах і поліпшений інтерфейс API. Фрагменти коду з використанням цих тегів демонструють їх переваги.

Слід навести два варіанта вбудовування звуку. Атрибут controls вказує на необхідність відобразити панель управління.

```
<audio src="v_a/dam.mp3" controls="controls">
Ваш браузер не підтримує теги audio!
</audio>

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

У першому варіанті за неможливості обробити об'єкт буде виведений напис, який міститься в тегу. У другому варіанті буде зроблено дві спроби відтворення звуку з різних джерел.

Використання тега video здійснюється аналогічно.

```
<video src="pr6.webm" width="320" height="240" controls></video>

<video width="320" height="240" controls>
<source src="pr6.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
<source src="pr6.ogv" type='video/ogg; codecs="theora, vorbis"'>
</video>
```

Приклад 3. Сторінка, яка демонструє можливості щодо відображення різних об'єктів. Вона уможливлює тестування браузера. Відображення сторінки у браузері надано на рис. 1.3.

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<meta http-equiv=Content-Type content="text/html; charset="utf-8">
<TITLE>Приклади вбудовування об'єктів</TITLE>
</HEAD>
<BODY>
<h1>Приклад 3</h1>
<table border=1>
<tr><td><h2>використовуваний тег</h2>
     <td><h2>EMBED</h2>
      <td><h2>OBJECT</h2>
      <td><h2>audio / video</h2>
<tr><td><h2>вбудовування звуку</h2>
      <td><EMBED SRC="all/dam.mp3" WIDTH=170 HEIGHT=60> </EMBED>
      <td><OBJECT TYPE="audio/mpeg">
                <param name="FileName" value="all\dam.mp3">
                <param name="AutoStart"  value="false">
                 Нічого не виходить
           </object>
      <td><audio controls="controls">
            <source src="all/pv.ogg" >
            <source src="all/dam.mp3">
         </audio>
 <tr><td><h2>вбудовування відео</h2>
     <td><EMBED SRC="all/kino.wmv"  width="300" height="200"> </EMBED>
     <td><OBJECT TYPE="video/x-ms-wmv"  width="300" height="200">
```

```
                <param name="FileName" value="all\kino.wmv">
                <param name="AutoStart"  value="false">
                Нічого не виходить
            </object>
      <td><video src="all/Subaru.mp4" width="320" height="240"
controls="controls"> </video>
</table>
          </BODY>
         </HTML>
```

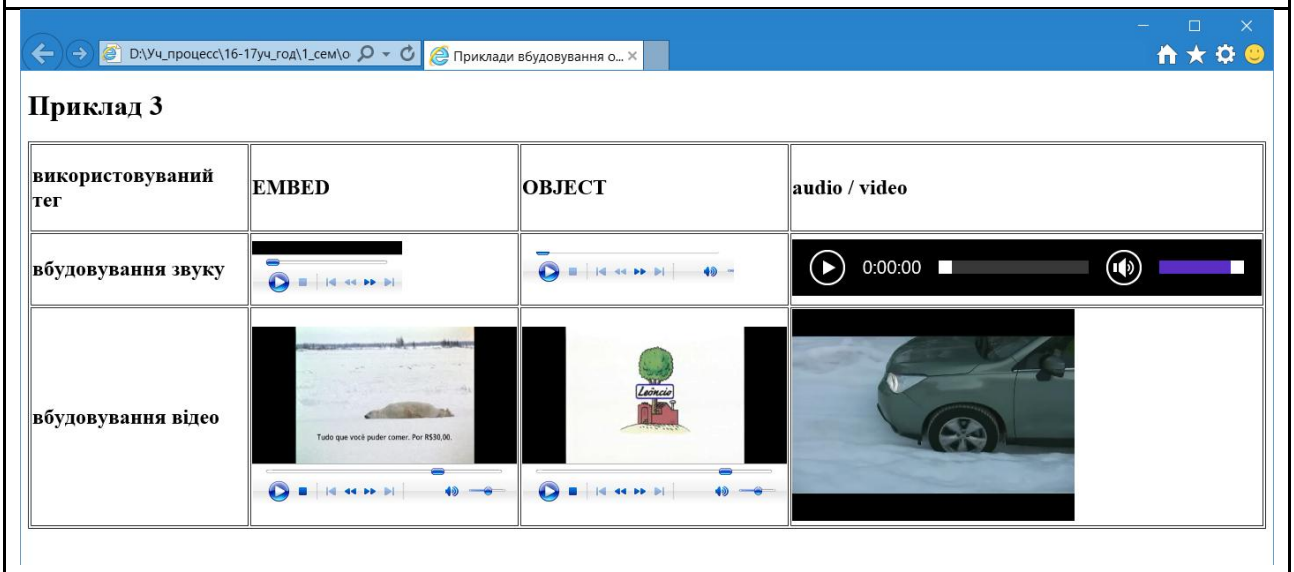

Рис. 1.3. **Вид сторінки у вікні браузера**

Можливість програвання звукових і відеофайлів на сторінках багато в чому залежить від наявності плагінів і кодаків (програм для розкодування). Велике число використовуваних кодувань і форматів створює певні проблеми під час відтворення. Ці проблеми легко виявити навіть перевіривши відображення сторінки з прикладу 3 у різних браузерах. У зв'язку з цим для підвищення ймовірності того, що користувач побачить ваше відео або почує звук, вбудовують ці об'єкти, попередньо опублікувавши їх на спеціальних сервісах, наприклад, **www.youtube.com**. У цьому випадку мультимедійні об'єкти перекодуються та поміщаються в контейнер flash (відповідні плагіни є в усіх сучасних браузерах). Після цього стає доступним посилання на ресурс, що забезпечує гарантоване відтворення. Для вбудовування зазвичай використовують плаваючий фрейм, наприклад:

```
<iframe width="560" height="315" src="//www.youtube.com/embed/OUS7jJdPDJM"
frameborder="0" allowfullscreen></iframe>
```

У будь-якому випадку у процесі вбудовування об'єктів слід ретельно тестувати розроблювані сторінки за різних умов їх перегляду.

Висновки й узагальнення

Розмітка тексту є відповідальним етапом у розробленні WEB-ресурсів. Шляхом розмітки створюється основне інформаційне наповнення сторінок. Водночас реалізується концепція розподілу змісту й оформлення. Для послідовної реалізації цієї концепції під час розмітки необхідно основну увагу приділяти змісту елементів, а не їх зовнішньому вигляду.

Відповідно до сформованої тенденції в розробленні та використанні ресурсів сформувалися три *основні вимоги* до розмітки сторінок: семантичність, логічність і валідність.

Вимога *семантичності* передбачає використання елементів (тегів) згідно з їх змістом (заголовки, абзаци і т. д.). Так, для різних за змістом частин сторінки слід використовувати відповідні контейнери FOOTER, HEADER тощо. Для виділення тексту рекомендується використовувати теги логічного форматування, а не b, font та ін.

Логічність передбачає природну структуру вмісту документа. Текст повинен бути легко зрозумілий користувачу. Слід використовувати конструкції мови за їх прямим призначенням, а не з метою отримання несподіваних ефектів. Наприклад, таблиці – це засіб структурувати текст, а не управляти розміщенням елементів. Для управління розміщенням (верстання) слугують таблиці стилів.

Під *валідністю* розуміють відповідність вимогам стандартів. Більша частина невідповідностей не призводять до катастрофічних наслідків. Однак необхідно пам'ятати, що і браузери, і засоби автоматичного аналізу сторінок скрупульозно фіксують усі ці випадки. Число і значимість таких невідповідностей можуть бути покладені в основу оцінки якості сторінки. Перевірка валідності може бути виконана з використанням спеціальних програм (валідаторів), вбудованих у різні засоби створення сторінок, э також відповідні плагіни для браузерів. Перевірку можна виконати і на спеціальних ресурсах в мережі Інтернет (наприклад, http: //validator.w3.org/).

Теоретичні запитання

1. Сформулюйте напрям розширення сфери використання мережі Інтернет.
2. Які технології використовують для додання сторінкам динамічних властивостей?
3. Що таке RSS?
4. Знання яких програмних засобів і для чого можуть знадобитися розробнику WEB-документів?
5. Опишіть процес взаємодії сервера та браузера.
6. Яку структуру може мати сайт?
7. Що таке юзабіліті?
8. Що таке бриф?
9. Які питання відображаються в концепції сайта?
10. Сформулюйте зміст етапів реалізації проекту створення WEB-сайта.
11. Які нові можливості надає HTML5 для дизайна?
12. Сформулюйте правила вживання тегів.
13. У чому полягає відмінність у вбудовуванні мультимедійних файлів різними тегами?
14. Як задати фоновий малюнок на сторінці?
15. У якій якості можуть використовуватися таблиці на WEB-сторінках?
16. Що таке гіперпосилання? Надайте приклад відповідного тегу.
17. Записати приклад використання тегів, які задають посилання на місце всередині сторінки.
18. Як розмістити на сторінці посилання у вигляді кнопки?
19. Дайте порівняльну характеристику різних форматів графіки для WEB.
20. Що нового надає HTML5 для створення форм?
21. Поясніть термін "семантична розмітка".

Контрольні завдання

1. Опишіть зовнішній вигляд сторінки у вікні браузера:

```
<html><head> <title>Контрольне завдання</title> </head>
<body>
  <form name=F1>
   <input type="text" name="TX1" size="35" value="обрати">
   <input type="reset" value="сброс">
  </form>
```

```
<HR>
<form>
   <select name="menu">
     <option selected value="primer31.htm">Поради
     <option value="primer32.htm">1
     <option value="primer36.htm">2
     <option value="primer35.htm">3
   </select>
   <input type=button value="Ok">
   <input type=password size=12 name=pw>
   <input type=checkbox checked name=uscitizen value=R1>
   <input type=checkbox name=uscitizen value=R2>
<TABLE CELLSPACING=40>
<TR><TD><P>задайте режим доступу
       <P><input type=checkbox name=uscitizen value=r1>режим 1
       <P><input type=checkbox checked name=uscitizen value=r2 >режим 2
   <TD><P>задайте вік
       <P><input type=radio checked name=age value="16-21">16-21
       <P> <input type=radio name=age value="21-35">21-35
</TABLE>
<HR>
<P><input type=reset value="СБРОС">
<input type=submit value="ВІДПРАВЛЕННЯ">
</FORM>
</FORM>
</body>
</html>
```

2. Опишіть зовнішній вигляд сторінки у вікні браузера:

```
<HTML>
  <HEAD>
   <TITLE>OBJECT</TITLE>
  </HEAD>
  <BODY>
<H1 align=midl>XXXXXX</H1>
<OBJECT data="all\pic1.jpg" type="image/jpeg" width=150 height=150>
 <OBJECT data="all\im1.gif" type="image/gif" width=150 height=150>
  <P> yyyy</P>
 </OBJECT>
</OBJECT>
<HR>
   <input type=radio checked name=age value="16-21">
   <input type=radio name=age value="21-35"
```

```
<FORM ACTION="mailto:victor@molchanov.eu.org" METHOD=POST>
<P>введите имя
<input type=text size=20 name=user value="your name">
<p>введите пароль
<input type=password size=12 name=pw>
<HR>
 </BODY>
</HTML>
```

3. Запишіть код сторінки, зовнішній вигляд якої наведено на рис. 1.4.

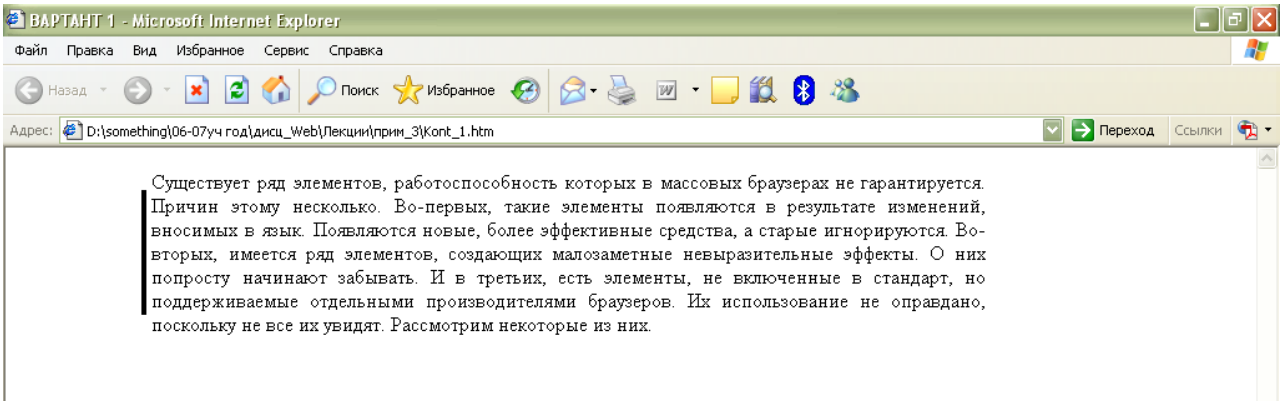

Рис. 1.4. **Вигляд сторінки у вікні браузера**

4. Запишіть код сторінки, зовнішній вигляд якої наведено на рис. 1.5.

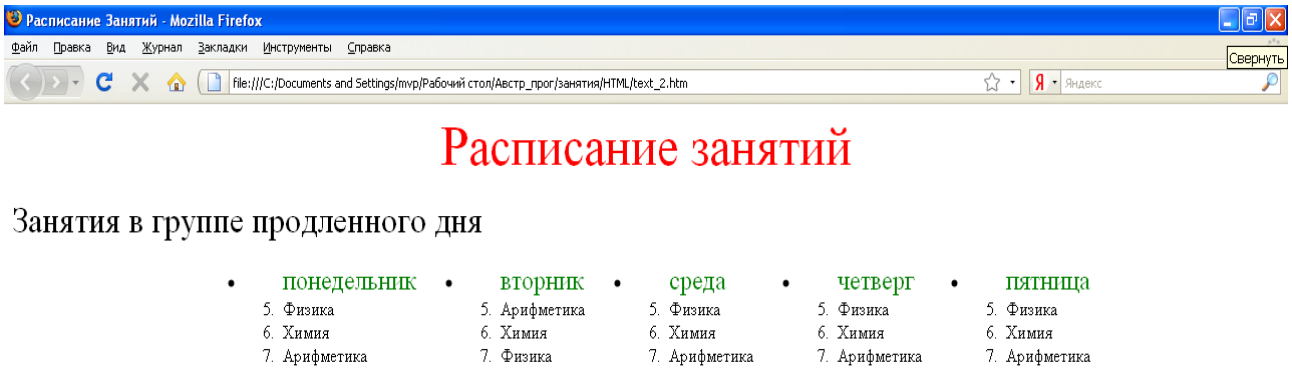

Рис. 1.5. **Вигляд сторінки у вікні браузера**

5. Запишіть код сторінки, що містить область для заголовка, область для елементів навігації, область основного контенту, що складається з трьох статей, і область з даними розробника і контактною інформацією (підвал). Кожна з областей повинна містити власний фон і елементи заповнення за власним задумом.

Розділ 2. Форматування WEB-документів

Основна ідея розділу

Розділ присвячений форматуванню WEB-сторінок за допомогою таблиць стилів CSS, верстання й адаптації сторінок до умов перегляду.

Ключові поняття розділу: правила CSS, селектори CSS, властивості елементів, верстання WEB-сторінок, CSS-фреймворки, Bootstrap, засоби адаптації до умов перегляду сторінок.

Питання розділу:

2.1. Використання стильових специфікацій.
- 2.1.1. Специфікація CSS.
- 2.1.2. Властивості елементів.

2.2. Форматування за допомогою CSS.
- 2.2.1. Використання таблиць стилів.
- 2.2.2. Створення ефектів.

2.3. Верстання сторінок.
- 2.3.1. Верстання за допомогою CSS.
- 2.3.2. Адаптивна верстання.

Цілі вивчення розділу

Інформація, подана у розділі, надає студентові можливість сформувати такі **компетентності**:

знання:

властивостей основних елементів та їх значень;

основних селекторів і прийомів їх використання;

впливу особливостей CSS на дизайн сторінок;

основних засобів і прийомів верстання WEB-сторінок;

уміння:

виконувати форматування WEB-сторінок з використанням таблиць стилів;

користуватися різними селекторами для форматування елементів;

ураховувати можливості CSS у ході створення дизайна сторінок;

виконувати верстання WEB-сторінок;

комунікації:

приймати зважене, з урахуванням думки інших розробників рішення про зміст і місце використання стильових специфікацій;

аргументувати прийняті дизайнерські рішення;

обґрунтувати обрання макету сторінок;

автономність і відповідальність:

здатність верстати сторінки з урахуванням особливостей різних браузерів, режимів відображення та поведінки користувачів;

здатність знаходити компромісні рішення.

Вступ

Основна концепція, яка покладена у основу сучасної WEB-технології, – це розподіл змісту та оформлення сторінок. Тому зміст описується за допомогою HTML у відповідних файлах (з розширенням htm або html), а оформлення здійснюється за допомогою правил CSS (таблиць стилів), як правило, в окремих файлах (з розширенням css).

За допомогою CSS керують не тільки виглядом елементів (власне оформлення, форматування), а також взаємним розташуванням елементів і їх поведінкою зі зміною умов перегляду (верстання).

2.1. Використання стильових специфікацій

Для більшої гнучкості процесу оформлення сторінок до складу WEB-технологій введений засіб задавання стильових специфікацій – CSS (каскадні таблиці стилів). Сенс терміну "каскадні" полягає в тому, що таблиць, які визначають вид документа, може бути декілька й існують правила взаємодії між ними.

Поняття стилів у CSS подібне до поняття стиля в текстових редакторах – це інструмент форматування елементів, розміщених на сторінках. Таблиці стилів містять формалізований опис основних параметрів форматування сторінки (властивостей елементів). Параметри форматування зв'язуються з тегами. Таким чином, з кожним елементом сторінки може бути зв'язаний ряд властивостей, які визначають його зовнішній вигляд і задаються в таблицях стилів.

2.1.1. Специфікація CSS

Правила створення та використання таблиць стилів визначені в стандартах консорціуму W3C. Сьогодні активно впроваджується версія стандарту CSS 3. Таблиця стилів дозволяє формально описати зовнішній вигляд і розташування елемента, його взаємодію з іншими елементами, а також зв'язати цій опис з одним або декількома елементами WEB-сторінки.

Таблиця стилів є набором правил. Кожне правило записується з нового рядка та складається з селектора та набору декларацій, який укладений у фігурні дужки. *Декларація* – це ім'я властивості та його значення, розділене двокрапкою. Одна декларація відділяється від іншої крапкою з комою. Наприклад, правило, яке задає вид тексту в тегі H1 може виглядати так: H1 {color:red; font-size: 14pt; text-align: center;}. Тут, H1 – селектор, color:red – декларація, color – властивість тексту (колір), red – значення кольору.

2.1.1.1. Приєднання таблиць стилів

Таблиця стилів може розміщуватися або в окремому файлі (з розширенням css), або в тексті сторінки (всередині тега style). Декларації CSS можуть також розміщуватися в тегах у вигляді значень атрибута style.

У ході розміщення в файлі правила послідовно розміщуються одне за іншим без жодної додаткової інформації та роздільників. Наприклад:

```
H1 {color: red; font-size: 14pt; text-align: center;}
H4 {font-weight:normal;font-style:italic;font-size:40pt}
```

Розміщення таблиці в окремому файлі дозволяє використовувати її для декількох сторінок одночасно, забезпечити єдність стильового оформлення. Таблиця розміщується у файлі з розширенням .css. Файл текстовий, його можна створити за допомогою будь-якого текстового редактора. Для встановлення зв'язку з таблицею використовується тег LINK, наприклад:

```
<LINK REL=STYLESHEET TYPE="text/css"
HREF="http://www.myserver.com/mysheet.css">.
```

Значення перших двох атрибутів цього тега є зарезервованими іменами, потрібними для того, щоб повідомити браузеру, що на цій сторінці використовуватиметься CSS.

TYPE визначає синтаксис таблиць (**text/css** – більшість браузерів підтримують тільки його, хоча може бути й інший, наприклад: **text/javascript**) і дозволяє браузерам, які не підтримують CSS, ігнорувати це посилання. Третій атрибут – HREF указує на файл, який містить описи стилів. Цей параметр повинен містити або відносний шлях до файла (у випадку, якщо він знаходиться на тому ж сервері, що і документ, з якого до нього звертаються), або повний (у випадку, якщо файл стилів знаходиться на іншому сервері).

Включення таблиці стилів у текст сторінки за допомогою тега style:

```
<!DOCTYPE html>
<html>
 <head>
  …
 <style >
        H1 {color: red; font-size: 14pt; text-align: center;}
        H4 {font-weight:normal;font-style:italic;font-size:40pt}
 </style>
</head>
 <body>
```

Таблиця стилів створюється для кожної сторінки. Тег може містити атрибут TYPE (text/css або text/javascript). Розміщується цей тег зазвичай усередині заголовка (HEAD).

Тег style може містити директиву @import. У цьому випадку відбувається імпорт таблиць, тобто читання із файла та розміщення в документі.

```
<STYLE TYPE="text/css">
@ import url(http://www.myserver.com/mystyle.css);
</STYLE>
```

Відмінність імпорту від зв'язування полягає в тому, що у зв'язуванні таблиця використовується як альтернативна, а в імпорті вона додається до документа та має більший пріоритет у використанні.

Включення таблиці стилів в тег:

```
<H1 STYLE="color: red; font-size: 14pt; text-align: center">ПРОБА</H1>
```

Цей спосіб використовується для оперативного перевизначення стилю, визначеного в таблицях. Параметри стиля, задані в атрибуті, перекривають параметри, які визначені поза тегом.

Слід зазначити, що надмірне захоплення визначенням стилю в тегах зводить нанівець основну ідею застосування каскадних таблиць стилів.

Однією з фундаментальних властивостей CSS є те, що можливе одночасне використання декількох таблиць стилів. Таким чином, тегів LINK і директив @import може бути по декілька екземплярів. Отже, на зовнішній вигляд документа може впливати не тільки одна таблиця стилів, а відразу декілька одночасно. Як наслідок, можуть виникати конфлікти між таблицями стилів, які впливають на документ.

Вирішення конфліктів ґрунтується на тому, що кожне правило має вагу. Правила з імпортованих таблиць перевизначають зовнішні (мають більшу вагу), а найбільшу вагу мають правила, вбудовані в теги. Проте цей спосіб змішує стиль зі вмістом і тому втрачає відповідні переваги таблиць стилів.

За необхідності автор може збільшувати вагу визначень, використовуючи спеціальну властивість !important, воно саме є значенням, і можна просто вставляти це ім'я у правила. Ця властивість не успадковується.

```
H1 { color: black ! important; background:white ! important }
P { font-size: 12pt ! important; font-style: italic }
```

У поданому прикладі перші три визначення мають підвищену вагу, а останнє – нормальну.

Отже, можна рекомендувати таку загальну схему використання таблиць стилів. Для створення невеликого проекту вибір способу включення таблиць автор здійснює сам, виходячи зі своїх цілей.

У ході створення значного проекту доцільно розробити велику оригінальну таблицю, помістити її в окремий файл і використовувати як альтернативу. На тих сторінках, де за задумом необхідно використовувати оригінальне оформлення, розмістити вбудовану таблицю. Для створення разових ефектів можна використовувати задання стилів у тегах. А для заборони перевизначення стильового оформлення додавати властивість-значення !important.

2.1.1.2. Селектори

Селектор – це формальна конструкція, яка пов'язує правило (параметри форматування) з конкретним елементом WEB-сторінки. У найпростішому випадку в якості селектора виступає ім'я тега.

Можна об'єднати декілька селекторів, згруповуючи їх, тоді відповідні елементи отримають однакові властивості, наприклад: P, LI {font-size: 12pt;}.

Окрім імен тегів, можна використовувати і багато інших селекторів. Найбільш використовувані з них:

селектори на основі значення атрибута class (класові селектори, Class Selectors);

селектори на основі значення атрибута ID (ID-селектори, ID Selectors);

селектори на основі значення інших атрибутів;

контекстні селектори (Contextual Selectors);

псевдокласи та псевдоелементи (Pseudo Classes, Pseudo element).

Використання різних селекторів додає цій технології додаткову гнучкість.

Класові селектори. CLASS – це атрибут, який може бути присутнім у будь-якому тегу HTML. Його значенням є ім'я, яке і використовується для задання правил. Завдяки цьому імені всередині однакових елементів можуть виділятися відособлені групи зі своїми параметрами форматування, і можна описати різні стилі тих самих елементів.

Класовий селектор складається з імені тега й імені класу, сполучених крапкою. Наприклад, .bl {color:blue; size:20pt}. Тут для тегів визначений клас bl з шрифтом 20 пунктів блакитного кольору. Тепер текст з тега <H1 CLASS="bl">TEXT</H1> буде блакитного кольору незалежно від інших.

ID селектори. Серед атрибутів різних тегів є ще один, значення якого може використовуватися як селектор, це атрибут ID="имя". У CSS цей селектор починається з символу #.

Після визначення правила #rd {color:red} усі елементи з тегами, що містять атрибут ID="rd", будуть відтворюватись червоним кольором. У тега обидва атрибути (CLASS і ID) можуть використовуватися одночасно, довизначаючи стиль елемента.

Селектори на основі значення інших атрибутів. Можна використовувати селектори на основі імен атрибутів і їх значень. Вони створюються з використанням імен тегів і квадратних дужок "ім'я-тега"["ім'я-атрибута"]. Наприклад:

H1[title] – теги H1, які мають атрибут title (незалежно від його значення);

a[href="foo"] – теги a, які мають атрибут href зі значенням "foo".

Можна вказати на зміст частині значення (наприкінці або на початку):

a[href^="http"] – літери http на початку значення (наприклад, http: //vt.com);

a[href$="JPG"] – літери JPG наприкінці (наприклад, img10.JPG).

Контекстні селектори. Контекстні селектори складаються з простих селекторів, розділених пропуском (усі описувані до цього селектори були простими). Вони застосовуються до елементів, зв'язаних спадковими стосунками (тобто до вкладених елементів) і задають властивості тільки конкретного дочірнього елемента. Завдяки цьому можна визначити різні стилі для різних послідовностей тегів, наприклад:

```
OL LI {list-style-type : decimal }
UL LI { list-style-type : square }
```

У цьому випадку елементи в нумерованому списку (OL) матимуть один стиль, а в ненумерованому (UL) – інший.

Контекстні селектори можуть містити тип елемента, атрибути CLASS, атрибути ID або їх комбінацію:

```
DIV P  { font : small sans-serif }
.reddish H1 { color : red }
#x78y CODE  { background : blue }
 DIV .sidenote H1 { font-size : large }
```

Перший селектор відповідає елементам P, які поміщені в DIV. Другий селектор відповідає всім елементам H1, які мають предка класу 'reddish'. Третій селектор відповідає всім елементам CODE, які є спадкоємцями елемента з ID=x78y. Четвертий селектор відповідає всім елементам H1, які мають предка DIV з класом 'sidenote' (знаходяться всередині відповідного розділу).

Можна групувати декілька контекстних селекторів:

```
H1 B, H2 B, H1 EM , H2 EM { color : red }
```

Псевдокласи та псевдоелементи. Стиль зазвичай застосовується до елемента відповідно до його позиції у структурі документа. Ця проста модель достатня для широкого спектра застосувань, але вона не покриває декілька типових ефектів (наприклад, ефекти, пов'язані з першим рядком абзацу, оскільки такого елемента немає). Концепція псевдокласів і псевдоелементів розширює механізм адресації в CSS, щоб дозволити інформації, зовнішній за відношенням до документа, робити вплив на процес форматування.

Псевдокласи та псевдоелементи – це особливі класи й елементи, відсутні в тексті HTML-документа, але присутні в CSS і автоматично визначувані підтримувальними CSS браузерами.

Псевдокласи розрізняють типи одного елемента (наприклад, посилання в різних станах: активне, вже відвідуване, ще не відвідуване), створюючи під час визначення власні стилі для кожного з них.

Псевдоелементи є частинами інших елементів, задаючи цим частинам відмінний від елемента в цілому стиль (наприклад, перший рядок в абзаці).

Селектори з псевдокласами та псевдоелементами складаються з будь-якого раніше розглянутого селектора двокрапки й імені псевдокласу або псевдоелемента. Необхідно розглянути деякі з них.

Псевдокласи елемента <a href=" ">, який позначає посилання, їх імена: link (посилання), active (активне посилання), visited (відвіданий раніше URL), hover (псевдоклас, що виникає при наведенні курсору на елемент). Приклади запису правил:

```
a:link,a:visited {color:blue}
a:active {color:red}
a:hover {text-decoration:none}
```

У даному прикладі всі посилання будуть синіми. З натисненням (у активному стані) вони поміняють колір на червоний, і при підведенні курсору мишки зникне підкреслення.

Псевдоелемент first-line – перший рядок елемента (First Line Pseudo-element) – може бути використаний з блоковими елементами (p, h1 і т. д.). Він змінює стиль першого рядка цих елементів.

Псевдоелемент first-letter – перша літера (First Letter Pseudo-element), схожий на first-line – але впливає не на весь рядок, а тільки на перший символ. Наприклад, це правило встановлює колір першої букви абзацу P:first-letter { color: purple }.

Окрім правил, таблиці стилів можуть містити коментарі (пояснювальний текст); коментар розміщується між символами /* */, наприклад:

```
BODY {margin-left: 1in}  /* Відступ на 1 дюйм*/
H1 {margin-left: -1in}       /* Зрушення вліво на 1 дюйм*/
```

2.1.2. Властивості елементів

CSS містять досить багато властивостей. Серед них є властивості, які можуть відноситься до багатьох елементів, а є специфічні – для конкретних тегів. Для зручності ознайомлення з властивостями слід розглянути їх за групами (область розміщення, позиціонування, фон, шрифт, текст, списки і т. д.). Це розбиття умовне, але воно допомагає орієнтуватися в призначенні властивостей.

Перед розглядом конкретних властивостей варто зазначити, що зі задаванням розмірів указується число з позначенням одиниці вимірювання, для цього можна використовувати *px (пікселі), in (дюйми), cm (сантиметри), mm (міліметри), pt (1pt = 1/72in), pc (1pc = 12pt).*

2.1.2.1. Область розміщення

Область розміщення будь-якого елемента – це прямокутник з основним змістом і рядом прилеглих полів (рис. 2.1).

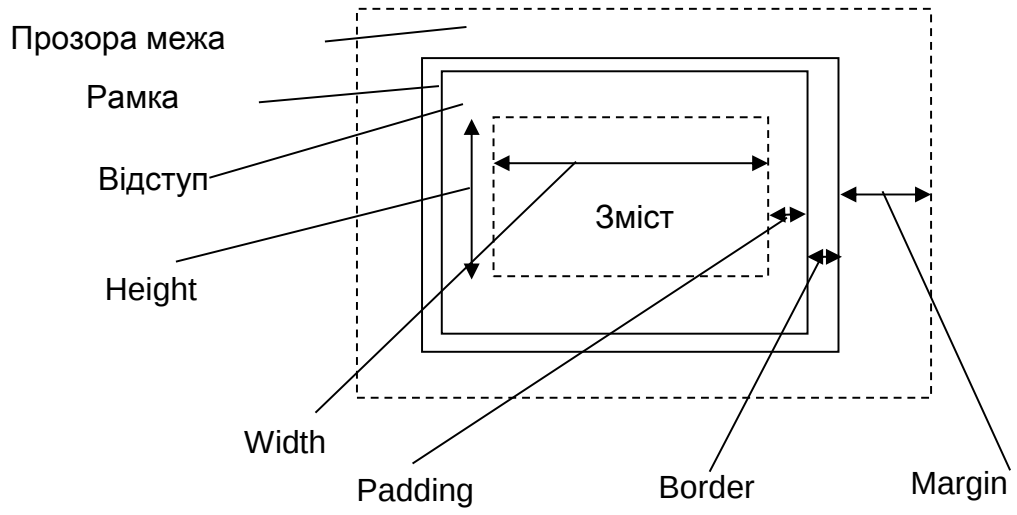

Рис. 2.1. **Схема області розміщення елемента**

Основний зміст – це власне елемент, наприклад малюнок з тега IMG. Його розмір задається властивостями width і height.

Вміст з чотирьох сторін оточений полями відступів. Їх розмір задається властивостями padding (padding-left тощо). Ця область заповнена тим же фоном, який заданий для елемента.

Поля відступів обрамлені рамкою, її товщина задається властивостями border-width (border-left-width і так далі). У рамки є й інші властивості border-color (border-top-color тощо), border-style (border-top-style та ін.).

Усі наведені області видимі, але видимістю елемента в цілому можна управляти за допомогою властивості visibility.

Навколо рамки розміщена прозора межа, розмір якої задається властивістю margin (margin-left і т. д.). Ця область прозора для фону документа або належного елемента, вона приймається за розміщення елементів. Зазвичай між двома сусідніми елементами цей розмір відповідає більшому зі значень.

У табл. 2.1 наведені імена основних властивостей цієї групи та значення, що набуваються ними.

Таблиця 2.1

Властивості області розміщення елемента

Ім'я	Значення	Примітка
width	<число><розмірність> або auto	Розмірність – px, in, cm, mm, %, em, …
height		
margin		
padding		
border-width		
border-color	black, coral, orange, … або rgb(r,g,b), rgb(r%,g%,b%) або #rrggbb	наприклад, #00cc00 або rgb(0,215,67)
border-style	none, dotted, dashed, solid, double, groove, ridge, inset, outset	Різні варіанти рамок: відсутня, пунктирна, штрихова і так далі

Для скорочення запису можна використовувати так звану узагальнену властивість. Для цього значення окремих властивостей, що входять в узагальнені, розділяються пропусками.

Наприклад, якщо необхідно вказати border-width:"10", border-color :"red", і border-style:"double", то можна записати значення узагальненої властивості border:"10 double red". Порядок переліку значень не істотний, браузер розпізнає їх за контекстом. Це дозволяє скоротити запис.

2.1.2.2. Позиціонування елементів

Властивості управління позиціонуванням забезпечують можливості управління розташуванням елементів усередині вікна або батьківського контейнера. Основні з них приведені в табл. 2.2.

Таблиця 2.2

Властивості, використовувані для позиціонування

Ім'я	Значення	Примітка
Display	Block, inline, inline-block, list-item, flex, none	Визначає тип елементу
Position	Absolute, fixed, relative, static	Задає систему позиціонування
Top, right, left, bottom	<число>	Визначають положення елемента
Float	Right, left, none	Обтікання
Z-index	<число>	Порядок накладення елементів
Visibility	Visible, hidden, collapse	Управління видимістю елемента
Overflow	Auto, hidden, scroll, visible	Режим відображення вмісту

Властивість display визначає, як елемент має бути показаний в документі відносно до сусідніх у потоці. Її можна застосувати до всіх елементів, але вона не успадковується. Специфікація CSS визначає багато можливих значень для цієї властивості (block, inline, inline-table, list-item, none, run-in, table, flex і так далі), проте більшість браузерів підтримують такі:

block – елемент показується як блоковий, відбувається перенесення рядків на початку та в кінці вмісту, тобто в його рядках навіть за наявності місця інші елементи не розміщуються;

inline – елемент відображається як вбудований в рядок, тому вміст елементів починається з того місця, де закінчився попередній;

inline-block – створюється блоковий елемент: його внутрішня частина форматується як блоковий елемент, а сам елемент – як вбудований;

list-item – елемент виводиться як блоковий і додається маркер списку;

flex – блоковий елемент, який автоматично заповнюється дочірніми елементами;

none – тимчасово видаляє елемент з документа, займане ним місце не резервується, і Web-сторінка формується так, немов елемента і немає.

Кожен елемент має своє значення за замовчуванням, наприклад, img – inline, div – block і т. п.

Властивість position визначає спосіб позиціонування елемента щодо вікна браузера або батьківських об'єктів на WEB-сторінці. Вона може набувати таких значень:

absolute – елемент видаляється зі звичайного потоку елементів документа, і його положення задається атрибутами left, top, right і bottom відносно краю вікна браузера;

fixed – це значення схоже на absolute, прив'язується до вказаної параметрами left, top, right і bottom крапки на екрані, але не міняє свого положення навіть з перегортуванням сторінки; різні браузери реагують на цю властивість дуже по-різному;

relative – положення елемента встановлюється відносно його початкового місця в потоці; додавання атрибутів left, top, right і bottom змінює позицію елемента та зрушує його в той або інший бік від первинного розташування – залежно від вживаного параметра;

static – елемент відображається звичайним чином, використання параметрів left, top, right і bottom не змінює місцеположення.

За замовчуванням використовується static.

Властивість visibility призначена для відображення або приховування елемента, включаючи рамку навколо нього та фон. У разі приховування елемента, хоча він і стає невидимім, місце, яке елемент займає, залишається за ним. Якщо передбачається виведення різних елементів в одне і те ж місце екрана, для обходу цієї особливості можна використовувати абсолютне позиціонування або скористатися властивістю display.

Значення для властивості visibility:

visible – елемент відображається як видимий;

hidden – елемент стає невидимим або, правильніше сказати, повністю прозорим, оскільки він продовжує брати участь у форматуванні сторінки;

collapse – застосовується для вмісту комірок таблиць. Вони реагують так, немов до них було додано стильову властивість display зі значенням none (рядки та колонки забираються, а таблиця перебудовується). Значення підтримується не всіма браузерами.

Властивість overflow управляє відображенням змісту блокового елемента, якщо воно цілком не поміщається, виходячи за область заданих розмірів. Сенс значень такий:

auto – відображається весь зміст елемента, навіть за межами встановленої висоти та ширини;

hidden – відображається тільки область усередині елемента, останнє обрізатиметься;

scroll – завжди додаються смуги прокрутки;

visible – смуги прокрутки додаються тільки за необхідністю.

З кожним елементом пов'язаний фон (таким чином, фон може бути не тільки в усієї сторінки, але й у окремих елементів). Управління фоном забезпечує група властивостей background, вони застосовуються до всіх елементів і не успадковуються. Властивості подані в табл. 2.3.

Таблиця 2.3

Властивості для управління фоном

Ім'я	Значення	Примітка
Background	<набір значень>	Узагальнена властивість
Background-attachment	Fixed, scroll	Режим прокрутки
Background-color	<колір>, transparent	Колір фону
Background-image	Url (шлях до файла), none	Фонове зображення
Background-size	<число> <число> або cover, або contain	Розмір зображення
Background-position	<число>%, left, center, right, top, center, bottom	Розташування зображення
Background-repeat	Repeat-x, repeat-y, repeat, no-repeat	Повторюваність зображення

Залежно від значення властивості background-attachment фонове зображення може бути зафіксоване та залишатися нерухомим або переміщатися спільно з документом.

Колір фону у властивості background-color задається так само, як у тегах: або ім'ям, наприклад, "red"), або шістнадцятковим значенням, наприклад, #6669FF, або за допомогою функції rgb, наприклад, rgb (100,100,150) або rgb(50%,60%,50%). За значенням transparent (використовується за замовчуванням) установлюється прозорий фон, через який видно фон батьківського елемента.

Як фон може використовуватися зображення, шлях до файла якого вказується у значенні властивості background-image з використанням конструкції url ("шлях"). Коли фонове зображення непотрібне, аргумент може набувати значення none. Якщо одночасно з малюнком для елемента

заданий колір фону, то кольоровий фон буде показаний, поки фонова картинка не завантажиться повністю. Те саме відбудеться, якщо зображення недоступне або відключений їх показ у браузері. У разі наявності в малюнку прозорих областей, через них просвічуватиме фоновий колір.

Властивість background-size визначає розмір зображення у пікселях або відсотках. Вона може також примати значення cover (розтягує фонове зображення на весь блок) або contain (розтягує фонове зображення зі збереженням пропорцій, поки його ширина або висота не упреться в край блоку).

Властивість background-position визначає початкове положення фонового зображення, встановленого за допомогою параметра background-image. У цієї властивості є два значення: положення по горизонталі (можливо – left, center, right або у відсотках, пікселях та інших одиницях) і вертикалі (можливо – top, center, bottom або у відсотках, пікселях та інших одиницях). Якщо застосовуються ключові слова, то порядок їх проходження не має значення, якщо – відсотки, то спочатку задається положення малюнка по горизонталі, а потім, через пропуск, положення по вертикалі. Наприклад, "top left " відповідає "0% 0%" (у лівому верхньому куті), а "top center " – "50% 0%" (по центру вгорі).

Властивість background-repeat визначає, як повторюватиметься фонове зображення, встановлене за допомогою параметра background-image, і по якій осі. Можна встановити повторення малюнка тільки по горизонталі (repeat-x), по вертикалі (repeat-y), в обидва боки (repeat) або без повторення (no-repeat).

Узагальнена властивість background у якості значення може містити набір значень складених властивостей (background-color, background-image і так далі). Значення записуються через кому або пробіл і будуть співвіднесені з відповідними властивостями за форматом запису. Завдяки цьому опис елемента буде коротшим. Наприклад:

background:"fixed url (bg .jpg) 50% 50% no-repeat " відповідає набору background-attachment:"fixed", background-image:"url(bg .jpg)", background-position:"50% 50%", background-repeat:"no-repeat".

Слід зазначити, що фонових зображень може бути кілька, шляхи до їх файлів та інші властивості задаються через кому. У ході формування сторінки фонові зображення накладаються одне на одне.

2.1.2.3. Властивості текстових елементів

Для управління зображенням текстових елементів використовують дві групи властивостей – font і text.

Група властивостей font. Властивості описують шрифти, за допомогою яких зображуються текстові елементи в абзацах, списках, заголовках. Це найчастіше використовувані властивості. Вони забезпечують дуже гнучкий і ефективний механізм форматування. Властивості групи font приведені в табл. 2.4.

Таблиця 2.4

Властивості шрифтів

Ім'я	Значення	Примітки
Font	<набір значень>	Узагальнена властивість
Font-family	<список імен>	Список імен шрифтів або сімейств
Font-size	xx-small, x-small, small ...	Розмір шрифту
Font-style	Normal, italic, oblique	Зображення шрифту
Font-variant	Normal , small-caps	Подання рядкових літер
Tont-weight	Normal, bold, bolder, lighter, 100, 200 ...	Насиченість шрифту

Властивість font-family встановлює список імен шрифтів або сімейств шрифтів, які використовуватимуться для оформлення тексту вмісту. Список може включати одну або декілька назв, розділених комою. Якщо в імені шрифту містяться пропуски, воно повинне розміщуватися в лапках.

У ході відображення послідовно перевіряється наявність на комп'ютері користувача відповідних шрифтів. Якщо такого шрифту немає, береться наступне ім'я зі списку й аналізується його наявність. Тому декілька шрифтів збільшують вірогідність, що хоч би один з них буде виявлений на клієнтському комп'ютері. Закінчують список зазвичай ключовим словом, яке описує тип шрифту, – serif (шрифти із зарубками), sans-serif (рубані), cursive (курсивні), fantasy (декоративні) або monospace (моноширинні). Розумно починати послідовність шрифтів з екзотичних типів і закінчувати узагальненим ім'ям, за якими буде підібраний найбільш відповідний з наявних.

За необхідності використання оригінального шрифту, його можна імпортувати за допомогою спеціальної директиві таким чином:

```
<style>
  @font-face {
   font-family: Pompadur;        /* Ім'я шрифта */
   src: url(fonts/pompadur.ttf); /* Шлях до файла зі шрифтом */
  }
</style>
```

Після цього він може використовуватися для форматування:

```
P {font-family: Pompadur;}
```

Розмір шрифту (**властивість font-size**) може бути встановлений декількома способами: абсолютно з використанням символьних констант, абсолютно зі вказівкою чисельного розміру та відносно батьківського.

Символьна константа (xx-small, x-small, small, medium, large, x-large, xx-large) для абсолютного розміру задає індекс у таблиці розмірів. Зазвичай розмір відрізняється від сусіднього в 1,5 рази.

Розмір можна задати з використанням будь-яких допустимих одиниць: пункти (pt), пікселі (px) і так далі, і у відсотках відносно до розміру шрифту батьківського елемента. Негативні значення недопустимі.

Ще один набір констант (larger , smaller) встановлює відносні розміри шрифту, збільшуючи або зменшуючи батьківський розмір.

Властивість font-style визначає зображення шрифту – звичайне (normal), курсивне (italic) або похиле (oblique). Коли для тексту встановлено курсивне або похиле зображення, браузер звертається до системи для пошуку відповідного шрифту. Якщо заданий шрифт не знайдений, браузер використовує спеціальний алгоритм для імітації потрібного виду тексту.

Властивість font-variant визначає, як потрібно подавати рядкові літери – робити їх всі прописними зменшеного розміру (small-caps) або залишити без змін (normal). Такий спосіб зміни символів називається капітеллю.

Властивість font-weight установлює насиченість шрифту. Значення встановлюється від 100 до 900 з кроком 100. Допустиме також використання ключових слів (normal, bold, bolder, lighter), проте доцільніше використовувати чисельне позначення, щоб задіяні слова не тлумачилися як назви гарнітур.

Окрім властивостей font, що визначають вид символів, є група властивостей, які стосуються тексту в цілому (відстань між символами, словами і тому подібне). Вони належать до групи text (табл. 2.5).

Таблиця 2.5

Властивості тексту

Ім'я	Значення	Примітки
letter-spacing	<число>, normal	Інтервал між символами
line-height	<число>, <число>%, normal	Міжрядковий інтервал
text-align	left, right, center, justify	Горизонтальне вирівнювання
text-decoration	none, underline, overline, line-through, blink	Варіант оформлення
text-indent	<число>, <число>%	Відступ першого рядка
vertical-align	baseline, sub, super, top-text, top, middle, bottom, bottom-text, <число>%	Вирівнювання по вертикалі щодо батьківського елемента або оточуючого тексту
word-spacing	<число>, normal	Інтервал між словами

Властивість text-decoration додає оформлення тексту у вигляді його підкреслення (underline), перекреслювання (overline), лінії над текстом (line-through) і миготіння (blink). Одночасно можна застосувати більше одного стиля, подаючи значення через пропуск. За замовчуванням встановлюється none.

Властивість word-spacing установлює інтервал між словами. Якщо встановлений параметр вирівнювання justify, то атрибут word-spacing не діє, оскільки інтервал між словами буде встановлений примусово.

Існує окрема група властивостей для управління видом списків (табл. 2.6). Властивості цієї групи застосовні до елементів зі значенням list-item для властивості display. Усі вони успадковуються.

Таблиця 2.6

Властивості списків

Ім'я	Значення	Примітки
list-style	<набір значень>	Узагальнена властивість
list-style-image	Url ("шлях"), none	Зображення маркера
list-style-position	Outside, inside	Розміщення маркера
list-style-type	Disc, circle, square ...	Вид маркера

Властивість list-style-image має значення: url ("шлях") або none (встановлює адресу зображення, яке слугує маркером списку). Ця властивість успадковується, тому за необхідністю відновлення маркера для окремих елементів списку використовується значення none.

Властивість list-style-position визначає, як розміщуватиметься маркер відносно тексту. Є два значення: outside – маркер винесений за межу елемента списку і inside – маркер обтікається текстом.

Властивість list-style-type змінює вид маркера для кожного елемента списку. Цей атрибут використовується тільки у разі, коли значення властивості ist-style-image встановлене як none. Маркери розрізняються для маркованого списку (тег UL) і нумерованого (тег OL). Значення: disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha, none. За замовчуванням – disc.

Узагальнена **властивість list-style** дозволяє одночасно задати стиль маркера, його положення, а також зображення, яке використовуватиметься як маркер.

На закінчення – ще одна цікава властивість. Це властивість cursor, використовуючи яку, можна змінювати форму курсора, коли він знаходиться в межах елемента. Вид курсора залежить від операційної системи та встановлених параметрів.

Перш ніж скористатися цією можливістю слід задуматися, чи буде це доречно. Когось з користувачів подібні зміни можуть ввести в оману, коли, наприклад, замість традиційної руки, що з'являється з наведенням на посилання, виникає щось інше.

Ця властивість застосовна до всіх елементів, вона успадковується та може набувати таких значень: auto (вид курсора за замовчуванням для поточного елемента), crosshair, default, e-resize, help, move, n-resize, ne-resize, nw-resize, pointer, progress, s-resize, se-resize, sw-resize, text, w-resize, wait, url ("шлях до курсора"). Останнє значення дозволяє встановити свій власний курсор. Для цього потрібно вказати шлях до файла, в якому зберігається зображення курсора в форматі CUR або ANI. Властивість підтримується не всіма браузерами.

2.2. Форматування за допомогою CSS

Найбільш складним завданням у застосуванні таблиць стилів є забезпечення однотипності відображення сторінок у різних браузерах і різних

пристроях. Іноді для програм кожного типу створюються окремі таблиці, які підвантажуються скриптом після відповідної перевірки. Тому слід ретельно тестувати сайт і корегувати правила таблиць.

2.2.1. Використання таблиць стилів

Більшість браузерів відображають абзаци без червоного рядка, вирівнюють текст по лівому краю та відокремлюють абзаци додатковим відступом. За необхідності можна надати тексту вигляду звичного для текстових редакторів, додавши форматування. Також можна створити заголовки зі застосуванням різних ефектів. Багато WEB-дизайнерів створюють власне оформлення для тексту, приділяючи велику увагу деталям. Наступний приклад демонструє можливості CSS з форматування тексту.

Приклад 4. Сторінка, яка демонструє можливості з форматування тексту. Вид сторінки у вікні браузера поданий на рис. 2.2.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Форматування тексту</title>
  <style >
body { background-image: url("pic1.jpg"),url("bg.jpg");
          background-repeat:no-repeat,repeat;
          background-position: center 300px;}
 div {  text-align:center}
 article {font-family: Times New Roman;
           margin: 0px auto;
           background-color: rgba(0, 120, 201, 0.7);
           width: 50%; height: 200px;
           color: #fff;
          }
P {   font-family: Times New Roman;
       font-size:14pt;
       text-indent: 1.25cm;
       text-align:justify;
       margin-bottom:0;margin-top:0}
 @font-face {font-family: NS; /* */
   src: url(american_textc.ttf);}
  .str {
```

```
   font: bold 8em NS;
   color: #0d3967;
   text-shadow: #cad5e2 5px 5px 0;
        }
.str_1 {
   font: bold 8em NS;
   color: #f0f0f0;
   text-shadow: #fff -1px -1px 0,  #333 1px 1px 0;
        }
 </style>
</head>
<body>
 <div class="str_1">Рельєфний заголовок </div>
 <div class="str">Заголовок з тінню  </div>
<article>
<p>...</p>
<p>...</p>
</article>
 </body>
</html>
```

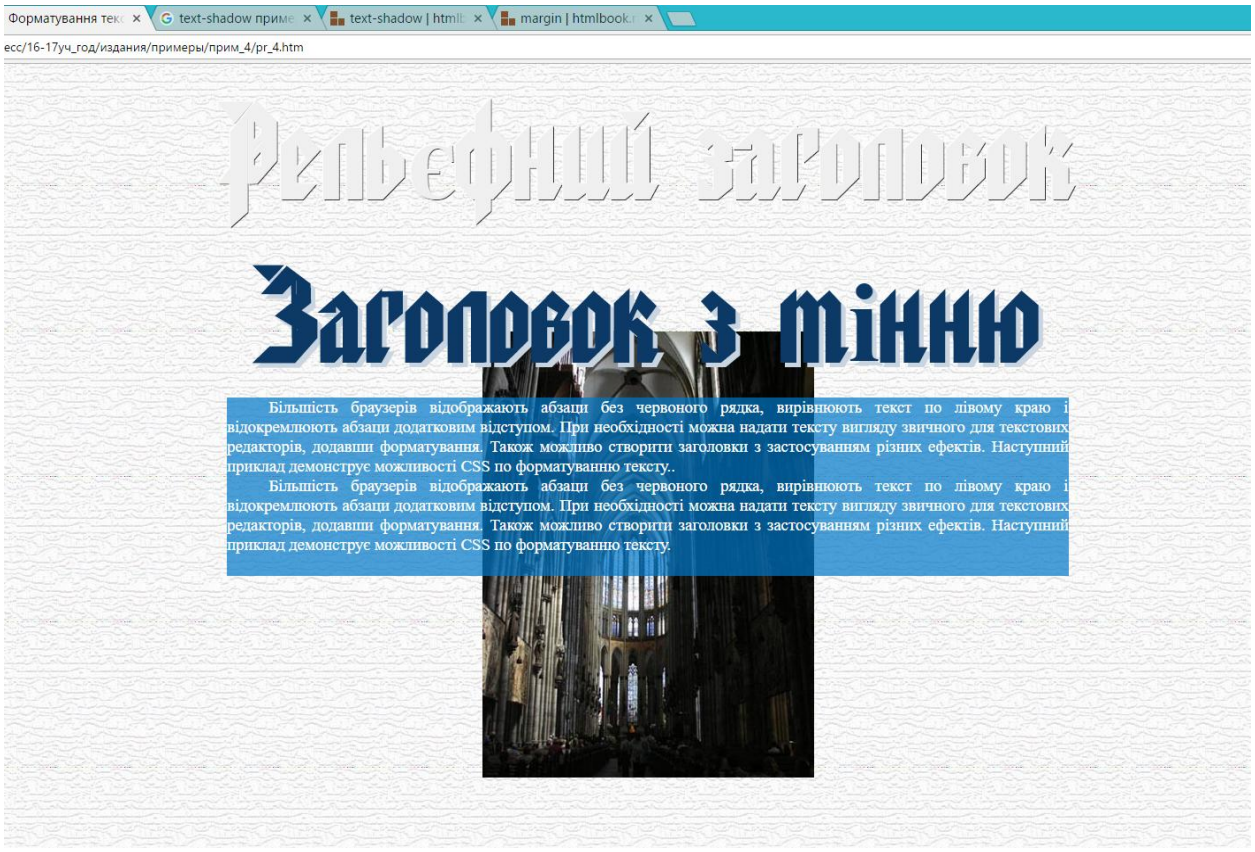

Рис. 2.2. **Вид сторінки у вікні браузера**

У цьому прикладі на сторінці використані два фонові малюнки: темний малюнок по центру і світлий, якій розмножується та заповнює все вікно.

Заголовки створені текстом, виведеним нестандартним шрифтом з тінями. У першому заголовку колір символів майже збігається з кольором фону, і він виглядає рельєфним. Для другого заголовка колір символів обраний контрастним, а тінь трохи світліше.

Основний розділ (поміщений в контейнер **article**) містить кілька абзаців тексту, відформатованих з червоним рядком, без додаткових відступів. Фон блоку напівпрозорий (видна фонова картинка). Для центрування блоку у вікні задана властивість **margin: 0px auto** (часто використовуваний прийом).

За допомогою таблиць стилів можна створювати навіть нові елементи. Наприклад, список посилань може стати горизонтальним меню і навіть багаторівневим.

Приклад 5. Сторінка, яка демонструє можливості зі створення меню, навіть з динамічним розкриттям другого рівня. Відображення сторінки у браузері надано на рис. 2.3, 2.4.

```
<!DOCTYPE html>
<html>
<head>
<title>МЕНЮ</title>
<style type="text/css">
#menu-outer {height: 60px;text-align:center; background: url(bg.jpg) repeat-x;}
#menu-outer ul li {display: inline;}
#menu-outer ul {padding-top: 10px;}
#menu-outer ul li button {border-radius:10px;}
button img {vertical-align: middle}

nav {display:block;width:30%;min-width:600px;margin: 0px auto;overflow:visible;}
ul.navig {height:30px;list-style: none;}
ul.navig li {line-height: 30px;background-color: #9DA4B7; height: 30px;float:left;
width:auto;border-radius:10px}
ul.navig li a {display:block;color: #fff;text-decoration:none;
        padding: 0 10px;border-right: 1px solid #f4f4f4;
        transition: background-color 0.5s linear;border-radius:10px}
ul.navig li a:hover{background-color: #99BCCC;}
ul.navig li ul {display:none;list-style: none;margin-left:-40px}
```

```
ul.navig li ul li{float:none;background-color: #3DA4B7;}
ul.navig li:hover ul{display:block;}
</style>
</head>
<body>
<div id="menu-outer">
   <ul>
      <li><a href="#"><button><img src="uba.gif"> Головна </button></a></li>
      <li><a href="#"><button><img src="uba.gif">Сторінка 1</button></a></li>
      <li><a href="#"><button><img src="uba.gif">Сторінка 2</button></a></li>
      <li><a href="#"><button><img src="uba.gif">Сторінка 3</button></a></li>
   </ul>
</div>
<nav>
   <ul class="navig">
      <li><a href="#">Головна   </a></li>
      <li><a href="#">Сторінка 1</a></li>
      <li><a href="#">Сторінка 2</a></li>
      <li><a href="#">Сторінка 3</a>
       <ul>
            <li><a href="#">Сторінка 3-1</a></li>
            <li><a href="#">Сторінка 3-2</a></li>
            <li><a href="#">Сторінка 3-3</a></li>
       </ul>
        </li>
    </ul>
</nav>
</body>
</html>
```

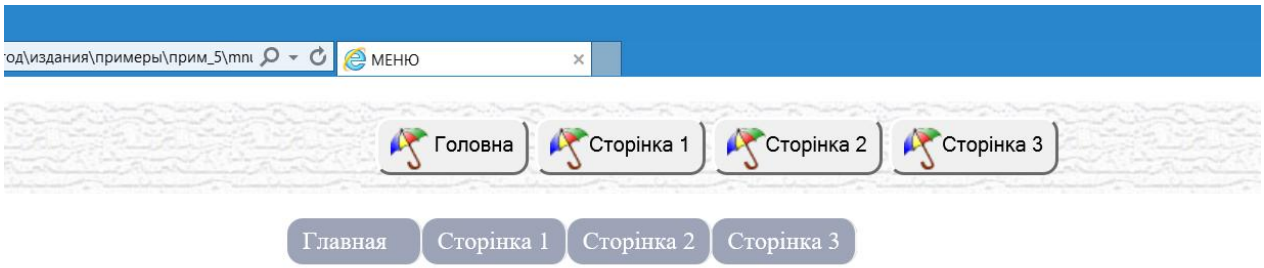

Рис. 2.3. **Вид меню у вікні браузера**

Рис. 2.4. **Вид меню з відкритим другім рівнем**

У цьому прикладі створено два меню. Для першого список посилань (блок **div** з **id="menu-outer"**) перетворений в горизонтальне меню. Для цього в посилання поміщені кнопки зі зображенням і написами. Елементи списку оголошені **inline**, у кнопок закруглені кути, а сам блок меню забезпечений фоном і фіксованою висотою.

На друге меню перетворені вкладені списки посилань (блок **nav**). Блок обмежений у ширині та вирівняний по центру вікна. У списків прибрані маркери. Елементи верхнього списку оголошені обтічними (це забезпечує їх горизонтальне розміщення). Посилання оголошені блоковими елементами із закругленими кутами, для них задана трансформація кольору за наведенням курсора. Показ меню другого рівня забезпечується зміною значення властивості **display:none** на **display:block** з наведенням курсора.

2.2.2. Створення ефектів

Таблиці стилів містять значну кількість властивостей, які забезпечують застосування до елементів різних ефектів (округлені кути блоків, тіні, обертання і т. п.). Більшість цих властивостей були включені в останню версію специфікації, вони підтримуються браузерами неоднаково. Крім того, багато браузерів підтримують власні властивості, відмінні від стандарту. У зв'язку з цим для забезпечення платформ запропоновано використовувати в іменах властивостей так звані вендорні префікси, які вказують на браузер, що підтримує дане властивість.

Значення префіксів для різних платформ наведені в табл. 2.7.

Таблиця 2.7

Вендорні префікси

Префікс	Виробник	Браузер	Браузерний движок
-o-, -op-, -xv-	Opera Software	Opera	Presto
-moz-	Проект Mozilla	Firefox, Camino и др.	Gecko
-ms-	Microsoft	Internet Explorer 8	Trident
-khtml-	Проект KDE	Safari, Konqueror.	KHTML
-webkit-	Apple	Google Chrome и др.	WebKit

Приклад властивостей з префіксами:

```
-ms-filter:"progid:DXImageTransform.Microsoft.Alpha(Opacity=50)";/*для IE 8*/
-moz-opacity:0.5;          /* для Mozilla 1.6 */
-khtml-opacity:0.5;        /* для Konqueror 3.1, Safari 1.1 */
```

Використання префіксів – тимчасовий захід, розрахований на перехідний період, поки різні браузери будуть в повній мірі підтримувати стандарт.

Серед числених ефектів слід розглянути основні:

закруглені рамки;

створення тіні до елементів;

можливість задати прозорий колір;

трансформації елементів;

анімація.

Рамки всіх елементів мають прямокутну форму з гострими кутами. За допомогою властивості **border-radius** можна вказати радіус заокруглення для кутів. Радіус задається в будь-яких допустимих одиницях. Для кута можна задати два значення (наприклад, 10px/20px), тоді кут буде округлений еліптично. Властивість може мати від одного до чотирьох значень, розділених пробілами:

1 – однаковий радіус для всіх чотирьох кутів;

2 – перше значення задає радіус верхнього лівого та нижнього правого кута, друге значення – верхнього правого та нижнього лівого;

3 – перше значення задає радіус для верхнього лівого кута, друге – одночасно для верхнього правого та нижнього лівого, а третє – для нижнього правого;

4 – для кожного кута задається свій радіус.

Наприклад:

border-radius: 13em 0.5em / 1em 0.5em 20em.

У цьому випадку кожен кут закруглений по-своєму, причому верхній правий – еліптично.

Іншим цікавим ефектом є створення тіні. Тінь може бути у блоку й у символів тексту. Тінь може відкидатися як назовні, так і всередину. Можна управляти розмірами, розмиванням і кольором тіні. Тіней може бути кілька. Імена властивостей **box-shadow** (для блоку) та **text-shadow** (для тексту). Значення окремих параметрів задаються в такій послідовності: **inset** (тінь усередині, за відсутності – зовні), 2 числа (зміщення по x і по y, в будь-яких одиницях), 1 число (розмиття, в будь-яких одиницях), 1 число (розтягнення, в будь-яких одиницях), колір.

Наприклад:

box-shadow: inset 0px 3px 20px 3px # f00;

Параметри другої та наступних тіней задаються через кому.

Ще одна властивість (**opacity**) дозволяє зробити елемент прозорим. Значенням властивості слугує число від 0 (повністю прозорий) до 1 (непрозорий). Наприклад, **opacity: 0.05**.

Ці ефекти були використані в прикладах 4,5.

Трансформації – це зміна елемента за деякий час. Для цього може змінюватися місце розташування, орієнтація або розміри, а також інші властивості, що задаються числами (наприклад, колір). Завдяки таким ефектам з'явилася можливість створення динамічних елементів без написання програмного коду. Слід зазначити, що така тенденція (заміна програмування на декларування) простежується в розвитку багатьох нових засобів.

У CSS 3 для цього включені три нових властивості: transform, transition і animation. Для кросплатформеного відображення – властивості необхідно застосовувати з вендорними префіксами. Але з метою спрощення розгляд властивостей буде виконаний відповідно до стандарту, без урахування особливостей різних браузерів. Трансформації найчастіше зв'язуються з псевдокласом hover, який відповідає елементу з наведеним курсором.

Властивість transform забезпечує зміну зовнішнього вигляду елемента відповідно до деякої функції (поворот на заданий кут, нахили, стиснення/розтягування зміщення та ін.). Значенням властивості є відповідна функція (rotate, skewY, skewX, scale, scaleX, scaleY, translate тощо).

Ще один вид трансформації пов'язаний з властивістю transition, яка забезпечує зміну властивостей елементів за заданий час від поточного значення до того, яке декларується в правилі. Характер зміни задається чотирма властивостями: transition-property (ім'я змінюваної властивості), transition-duration (час зміни в секундах), transition-timing-function (функція зміни часу), transition-delay (затримка перед початком зміни). Вони можуть бути записані у вигляді узагальненої властивості transition. Наприклад, колір фону буде змінений наведенням курсора до значення # 808991 за 3 сек, з уповільненням до кінця та затримкою перед початком 1 сек:

.trans:hover {background: #808991;transition:background 3s ease-out 1s}

Інші значення для transition-timing-function: ease (на початку повільно, потім швидше, до кінця знову сповільнюється), ease-in (на початку повільно, до кінця прискорюється), ease-out (починається швидко, до кінця сповільнюється), linear (однакова швидкість від початку і до кінця).

Змінюваних властивостей в декларації transition може бути кілька.

Крім ефектів transform і transition, у специфікацію включена властивість animation, яка забезпечує ланцюжок змін властивостей елементів. Ця властивість містить значення, що визначають характер анімації в цілому: назва (animation-name), тривалість (animation-duration), функція зміни часу (animation-timing-function), стан елемента до і після відтворення анімації (animation-fill-mode) та ін. Можна використовувати узагальнену властивість, наприклад, анімація з ім'ям move (animation-name), тривалістю 10 сек (animation-duration), час змінюється лінійно (animation-timing-function):

animation: move 10s linear;

З анімацією зв'язується ланцюжок значень властивостей (кроків анімації), тобто ключовими кадрами (властивість @keyframes). Для кожного ключового кадру задаються початкове та кінцеве значення змінюваної властивості, його проміжні значення та час змін (у відсотках). У ключові кадри можна також включати властивість animation-timing-function. Наприклад, такі ключові кадри забезпечать переміщення елемента від лівого краю вікна до правого та назад (елемент повинен мати властивість position: absolute):

```
@keyframes move {
  from {left: 0%;}
  50% {left: 100%;}
  to {left: 0%;}
}
```

Таким чином можна створювати досить складні динамічні ефекти без використання програмування. Наостанок слід зазначити, що підтримка анімації в браузерах поки неоднакова, тому необхідно використовувати вендорні префікси.

Приклад 6. Сторінка демонструє можливості зі створення динамічних ефектів. Оскільки передати динаміку на малюнку неможливо, наведено початковий і фінальний вигляд елементів (рис. 2.5, 2.6).

```
<!DOCTYPE html>
<html>
<head>
<title>анімація</title>
<style type="text/css">
.turn:hover {transform: rotate(30deg); }
 .pos {position:absolute;left:200px;top:200px;}
.trans {position:absolute;width:140px;height:140px}

.trans:hover {left:0px;background: #808991; transition:background 3s ease-out 1s, left 2s linear 1s;}
.trans {top:150px;left:-100px; background: #D0D941; transition:background 3s ease-out 1s, left 2s}

#sun {position:absolute; left:0;bottom:0;width:200px;height:200px;background-color:yellow; border-radius:100px;
                -webkit-animation-name: 'move';
                -webkit-animation-timing-function:linear;
                -webkit-animation-duration: 10s; }
        @-webkit-keyframes 'move' {
           from {bottom: 0%; left: 0%; }
           20%  {lbottom:20%;}
                 40% {bottom: 50%;}
                 50% {bottom: 55%;}
                 60% {bottom: 50%;}
                 80% {bottom: 20%;}
                 to     {bottom:  0%;  left: 80%;}
```

```
        }
</style>
</head>
<body>
<img src="pic.gif" alt="pic" class="turn">
<div class="turn pos">ggggggggnnnnnnnnn</div>
<div id="sun"></div>
<div class="trans"></div>
</body>
</html>
```

Рис. 2.5. **Початковий вигляд сторінки**

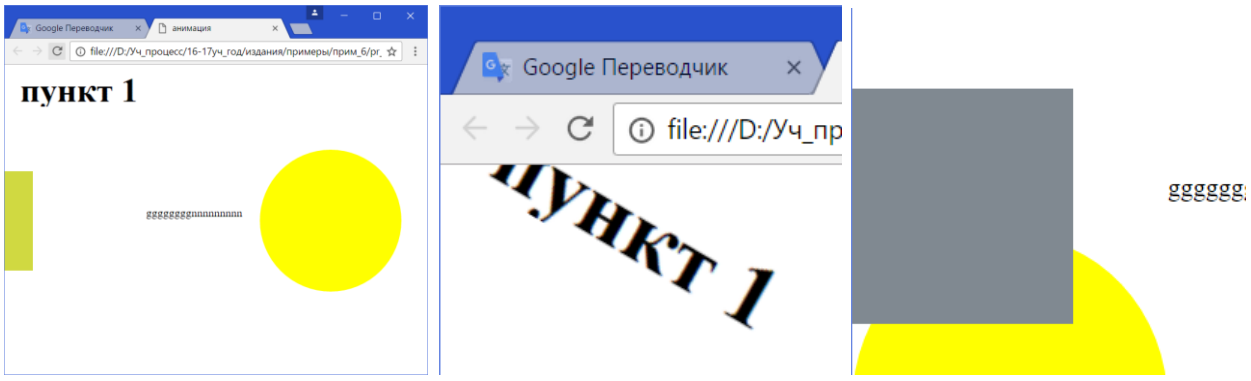

Рис. 2.6. **Динамічні ефекти з елементами**

Приклад містить три розглянутих види трансформацій. Перший ефект забезпечує поворот елемента на 30 градусів за наведенням курсора. Другий – плавно висуває елемент з лівого краю вікна та змінює колір фону, роблячи його повністю видимим після наведення курсора на корінець. Третій – переміщує зафарбоване коло по криволінійній траєкторії.

Має сенс звернути увагу на ім'я класу у елемента . Таке ім'я дозволяє зв'язати його з двома правилами (селектори .turn і .pos).

Властивості для анімації використані з векторним префіксом webkit, тому приклад слід перевіряти в браузері Google Chrome.

2.3. Верстання сторінок

Приведені підходи до форматування елементів дозволяють достатньо просто створювати макети сторінок. Проте за допомогою ретельного розгляду та тестування у різних браузерах будуть помітні різноманітні дефекти (смуги між областями, накладання областей, незадовільний вигляд за зміни розмірів вікна і т. п.). У зв'язку з цим розробникам доводиться корегувати розміщення елементів та їх властивості, щоб добитися бажаного результату. Цей процес називається верстанням сторінок.

Верстання – це розташування складових елементів (тексту, заголовків, зображень, таблиць) на сторінці документа. Верстання виконується для друкованих видань і для WEB-сторінок. Верстання WEB-сторінок – це створення HTML і CSS коду, який забезпечує зображення елементів WEB-сторінки в потрібних місцях вікна браузера відповідно до розробленого макету. У процесі верстання враховуються обмеження, які накладає HTML і CSS, особливості браузерів і вимоги до верстання.

2.3.1. Верстання за допомогою CSS

Вимоги до верстання розподіляють на загальні та спеціальні.

Загальні вимоги: кросплатформеність, адаптивність і логічність.

Під *кросплатформеністю* розуміють однаковий вигляд сторінки в різних браузерах. Звісно, допускаються відмінності, але основні дизайнерські рішення мають зберігатися.

Адаптивність передбачає прийнятне відображення сторінки на пристроях з різними характеристиками (розміри екрану і т. п.).

Логічність передбачає зрозумілу структуру коду, використання зрозумілих імен, семантичну розмітку.

Спеціальні вимоги формулюються для конкретного проекту. Це можуть бути вимоги до поведінки елементів у разі зміни параметрів перегляду, вимоги до окремих елементів тощо.

У ході верстання послідовно вирішуються такі питання:

визначення вигляду та складу таблиць стилів;

визначення та розміщення областей на сторінці;

поведінка областей за зміни параметрів вікон;

колірні та шрифтові рішення.

Перш за все необхідно визначитися з вибором таблиць стилів, які будуть використані (зовнішні, імпортовані, вбудовані). Слід одразу відмовитись від вбудованих, як найменш гнучких. Якщо передбачається, що однотипних сторінок буде декілька, то доцільне зберігання таблиць в окремому файлі або декількох файлах, якщо сторінка одна, то особливої різниці в місці розміщення таблиці немає.

Ще одне міркування стосується внутрішньої організації самої таблиці. Якщо її розмір буде значним, то доцільно згрупувати правила за деякими ознаками, розділяючи їх коментарями. Такою ознакою може бути приналежність до певного інформаційного блоку (поля на сторінці) або за призначеннями правил (позиціонування блоків, колірні характеристики, шрифти і т. п.).

Основою для верстання слугує макет сторінки, наприклад, у вигляді ескізу. Попри всю різноманітність дизайнерських рішень існують три підходи до розміщення областей (шарів): смугами, колонками та комбінація двох перших. Необхідно розглянути реалізацію цих підходів різними засобами.

2.3.1.1. Верстання смугами

Верстання смугами найбільш просте в реалізації. Контейнерами виступають блокові елементи (div, header, і т. д.). Основна увага повинна приділятися вирівнюванню, відступам і поведінці зі зміною розмірів вікна.

Шаблон розмітки та відповідна таблиця стилів для сторінки з чотирьох смуг (заголовок, панель навігації, область основного контенту та "підвал") можуть виглядати так:

```
<!DOCTYPE html>
<HTML>
<HEAD>
<TITLE>ЗАГОЛОВОК</title>
<style>
 body {background: #f0f0f0;}
 header, nav, section, footer {margin: 0 auto;}
 header {height: 80px;}
```

```
 nav {height: 60px;text-align:center;}
 nav ul li {display: inline;}
 nav ul {padding-top: 10px;}
 nav ul li button {border-radius:10px;}
 section {width: 90%; text-align:justify; }
footer {height:60px; width:90%;text-align:right;}
</style>
</head>
<BODY>
<header>
Заголовок (логотип, назва, форма пошуку по сайта, ...)
</header>
<nav>
   <ul>
      <li><a href="#"><button>посилання 1</button></a></li>
      <li><a href="#"><button>посилання 2</button></a></li>
      <li><a href="#"><button>посилання 3</button></a></li>
      <li><a href="#"><button>посилання 4</button></a></li>
  </ul>
</nav>
<section>
Основний зміст …
</section>
<footer>
  Розміщення контактної інформації
</footer>
</body>
<html>
```

Правило {margin: 0 auto;}, яке застосоване до всіх чотирьох елементів, забезпечує їх розміщення по центру вікна. Група правил, яка пов'язана з елементом nav, перетворює список посилань у горизонтальне меню. Елемент footer ("підвал") знаходиться в нижній частині сторінки. Властивості можуть бути доповнені для додання необхідного вигляду або поведінки.

2.3.1.2. Верстання колонками

Розміщення контейнерів з вмістом у кілька колонок може бути здійснено з використанням різних засобів. Найчастіше використовується властивість обтікання та блоки flex.

Дві колонки створюються гранично просто: два розділи float:left і float:right або один статичний, а один з float.

Перший варіант:

```
#main {float:right; width:65%;}
#sections {float:left; width:35%;}
```

Другий варіант:

```
#main {width:65%;}
#sections {float:left; width:35%;}
```

У використанні статичного розділу важливо, щоб розділ з float ішов у потоці першим:

```
<div id="sections">
...
</div>
<div id="main">
...
</div>
```

Третю колонку можна розмістити, якщо задати їй відповідне обтікання та розмістити в потоці після розділу, до якого вона примикає (main після sections):

```
#main {float:left;width:55%;}
#sections {float:left;width:20%;}
#news {float:right; width:25%;}
```

Позиціонувати колонки можна також, використовуючи збільшені відступи ліворуч і праворуч. Наприклад, задаючи margin-left можна поміняти main і sections місцями:

```
#main {float:left;width:55%;margin-left:-75%;}
#sections {float:left;width:20%;margin-left:55%;}
#news {float:right; width:25%;}
```

За допомогою властивості clear можна керувати взаємодією розділів, які обтікаються, з іншими. Властивість скасовує обтікання. Це означає, що елемент буде розміщений строго нижче попереднього, якій обтикається. Властивість може приймати значення left, right, both, скасовуючи обтікання ліве, праве та обидва. Наступній приклад демонструє ці можливості.

Приклад 7. Сторінка демонструє можливості верстання з використанням властивості float. Створюються три колонки. Оскільки розмір вказаний у відсотках, зі змінами розмірів вікна розмір колонок змінюється пропорційно. Такий підхід отримав назву "гумової" верстки, а подібні макети називаються "гумовими"

```
<!DOCTYPE html>
<HTML>
<HEAD>
<meta charset="UTF-8" />
<title>Верстання float</title>
<style>
 body {background-image:url(bg.jpg);}
 header, nav, section, footer {margin: 0 auto;}
 header {height: 60px;text-align:center;font-size:2em}
 nav {height: 60px;text-align:center;}
 nav ul li {display: inline;}
 nav ul {padding-top: 10px;}
 nav ul li button {border-radius:10px;}
 section {width: 90%; text-align:justify; }
footer {height:60px; width:90%;text-align:justify;clear:both}
#main {float:left;width:50%;padding:1%}
#sections {float:left;width:20%;padding:1%}
#news {float:right; width:25%;1%}
</style>
</head>
<BODY>
<header>
Заголовок (логотип, назва, форма пошуку по сайта, ...)
</header>
<nav>
   <ul>
       <li><a href="#"><button>посилання 1</button></a></li>
       <li><a href="#"><button>посилання 2</button></a></li>
       <li><a href="#"><button>посилання 3</button></a></li>
       <li><a href="#"><button>посилання 4</button></a></li>
  </ul>
</nav>
<section>
  <div id="sections">
  <H1>секція 1</h1> <BR>
<P>Бокова панель … панельБокова панель</p>
```

```
  </div>
<div id="main">
  <H1>Основний контент</h1><BR>
                <p>Основний контент …</p>
  </div>
<div id="news">
  <H1>секція 2</h1> <BR>
  <p>Новини  …</p>
 </div>
</section>
<footer>  Розміщення контактної інформації …
</footer>
</body>
<html>
```

На сторінці використане меню з прикладу 5. Вигляд сторінки наведено на рис. 2.7.

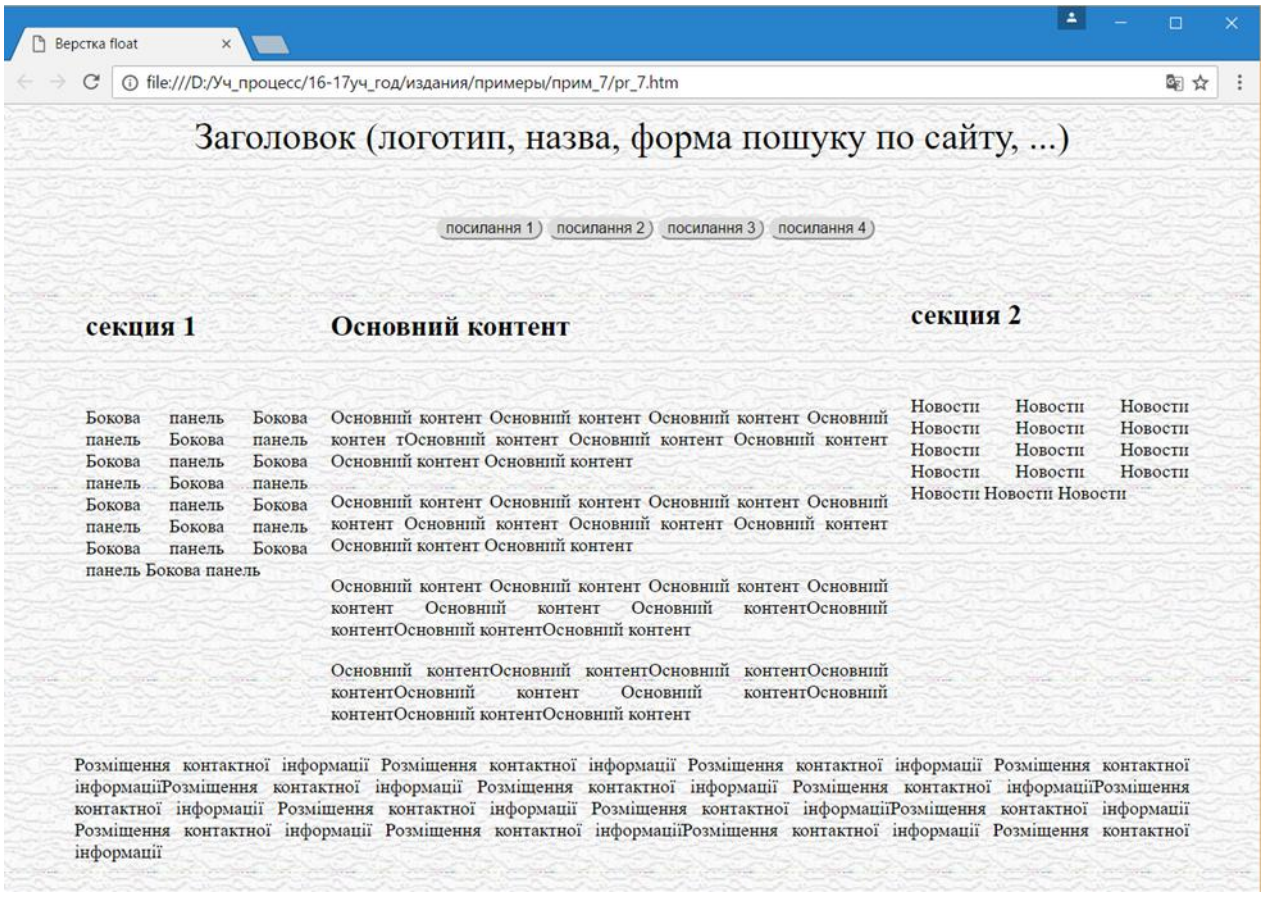

Рис. 2.7. **Вигляд сторінки у вікні**

Гнучкість технології забезпечується безліччю засобів для досягнення подання контенту в необхідному вигляді.

 Текстовий вміст контейнера можна розмістити в кілька колонок. Для цього запропонована властивість columns (column-width і column-count): column-count задає кількість колонок, а column-width – ширину колонки. Браузер сам розбиває контент на потрібну кількість колонок, щоб заповнити зовнішній контейнер. За спільного використання властивостей column-count інтерпретується як максимальна кількість колонок. Додаткові властивості column-gap і column-rule задають відступ між колонками та розділову лінію (column-rule-color, column-rule-style, column-rule-width – приймають значення аналогічні властивості border). Є властивості й для управління розміщенням контенту на колонках: column-span, та ін.

Більшість браузерів підтримують ці властивості з вендорними префіксами.

Блоки на екрані у вигляді колонок можуть бути створені також з використанням властивості flex. З цією метою створюється блок, який автоматично заповнюється дочірніми елементами з корегуванням їх розмірів. Структура контенту для цього повинна бути подана у вигляді блоку зі вкладеним довільним числом дочірніх елементів (блоків).

Приклад структури контенту:

```
<div id="cont">
   <div class="c1"> …</div>
   <div class="c2"> …</div>
   <div class="c3"> …</div>
</div>
```

Для батьківського блоку вказується display:flex і параметри заповнення: flex-direction (row – по горизонталі, column – по вертикалі), flex-wrap (wrap – кілька смуг, nowrap – в одну смугу), justify-content, align-items, align-content (різні варіанти вирівнювання).

Для дочірніх блоків указується порядок знаходження (order), ступінь можливого збільшення (***flex-grow***) і зменшення розмірів (***flex-shrink***), а також початковий (базовий) розмір (***flex-basis***). Можна використовувати узагальнену властивість, наприклад, ***flex: 2 2 450px*** (цей елемент можна зменшувати та збільшувати щодо початкового розміру в 450 пікселів, але ступінь зміни розмірів залежить від властивостей інших блоків).

Наприклад, для наданої структури можна задати стилі:

```
#cont {display: flex; flex-direction: row;}
.c1 {flex:2 1 300px }
.c2 {flex:1 1 300px }
.c3 {flex:1 1 300px }
```

У цьому випадку через нестачу місця (вікно менше 900 пікселів) усі три блоки будуть зменшені, але їх розмір буде однаковий. Якщо місця більше 900 пікселів, то перший блок буде більший за інші.

З використанням властивості flex для верстання розробник позбавлений обліку безлічі дрібних деталей і нюансів поведінки елементів. Як ілюстрацію цих можливостей треба зверстати сторінку, вигляд і поведінка якої подібні до поданої у прикладі 7.

Приклад 8. Сторінка, для верстання якої використані блоки з властивістю display:flex. Вигляд сторінки наведений на рис. 2.8.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8" />
<title>Верстання flex</title>
<style>
 body {background-image:url(bg.jpg);}
.box {display: flex; flex-direction: column;width: 90%;margin: 0 auto;}
footer, header, nav {flex: 0 0 100%; padding: 12px; text-align:justify;}
header {height: 60px;text-align:center;font-size:2em}
nav {height: 60px;text-align:center;}
nav ul {padding-top: 10px;}
nav ul li {display: inline;}
 nav ul li button {border-radius:10px;}
section {display: flex;flex-direction: row;}
.sidebar_first { order: 3; flex: 0 0 150px;}
.sidebar_second { order: 1;flex: 0 0 210px;}
.content_ {order: 2;flex: 1 1 0;}
</style>
</head>
<body>
 <div class="box">
       <header>Заголовок (логотип, назва, форма пошуку по сайта, ...)</header>
       <nav class="nav">
         <ul>
          <li><a href="#"><button>посилання 1</button></a></li>
          <li><a href="#"><button>посилання 2</button></a></li>
          <li><a href="#"><button>посилання 3</button></a></li>
          <li><a href="#"><button>посилання 4</button></a></li>
        </ul>
       </nav>
       <section>
```

```
            <div class="content_">
              <H1>Основний контент</h1><BR>
              <p>Основний контент …</p>
            </div>
           <div class="sidebar_first">
               <H1>секція 2</h1> <BR>
              <p>Новини  …</p>
            </div>
            <div class="sidebar_ sidebar_second">
              <H1>секція 1</h1> <BR>
               <P>Бокова панель …</p>
             </div>
     </section>
    <footer>Розміщення контактної інформації ….  </footer>
  </div>
</body>
</html>
```

У цьому прикладі використані два блоки з властивістю display:flex, один (тег div з атрибутом class="box">) забезпечує розміщення у вікні чотирьох смуг, а другий (тег section) – трьох стовпців в третій смузі.

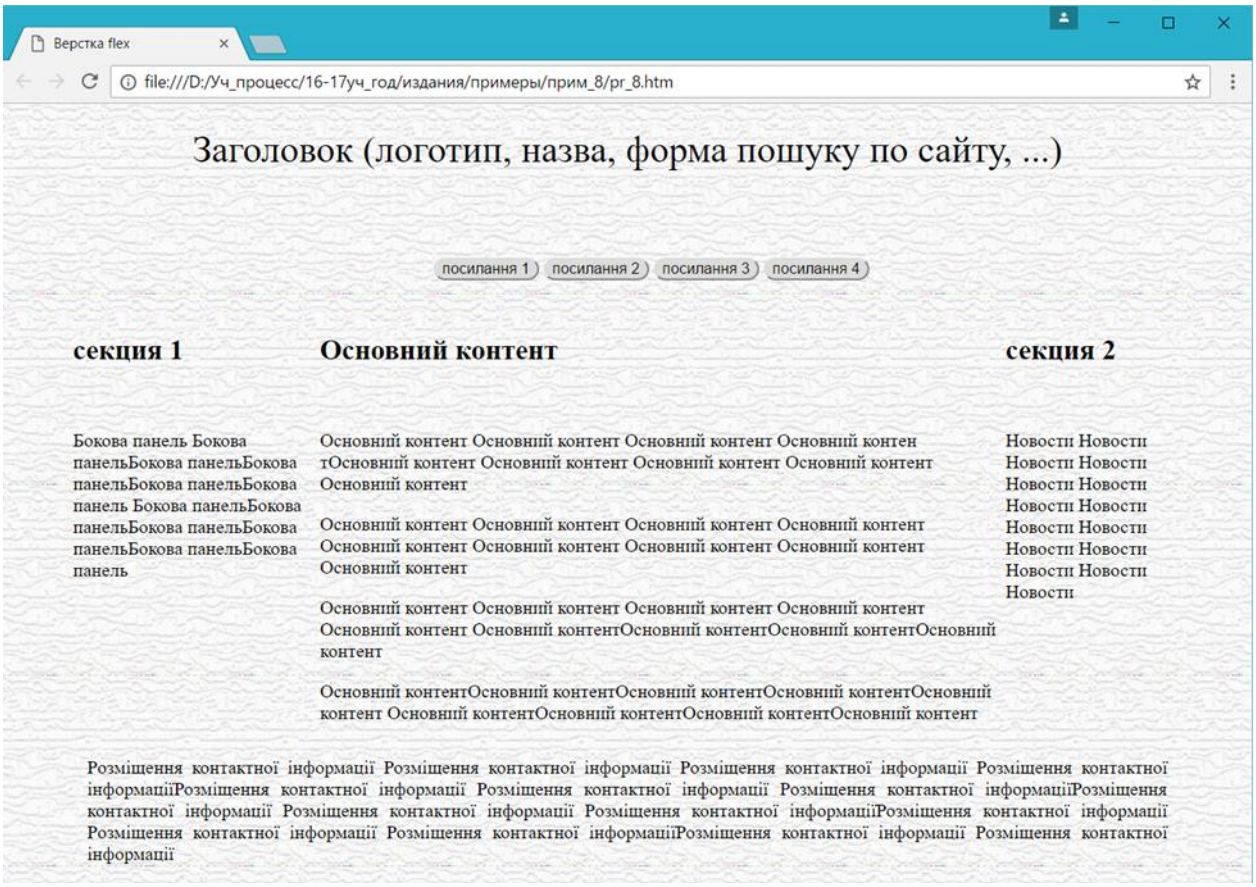

Рис. 2.8. **Вигляд сторінки у вікні**

Неважко переконатися, що вигляд у вікні не відрізняється від наведеного у прикладі 7.

2.3.2. Адаптивне верстання

Серед основних вимог до верстання часто називають кросплатформеність, що означає однакове відображення на різних платформах (різних пристроях, різних ОС, різних браузерах). Однак, зважаючи на значні різноманітності пристроїв, з яких користувачі виходять в Інтернет, виконання цієї вимоги стає неможливим. Дійсно, чи можна однаково відобразити сторінку на екрані з діагоналлю 5 і 25 дюймів? Для забезпечення комфортної роботи чимось доводиться жертвувати, змінювати зовнішній вигляд сторінки. Підходи до вирішення цієї проблеми постійно розвиваються й удосконалюються. Нині сформувалося кілька напрямів: створення окремих версій сайта з різними URL для окремих типів пристроїв (наприклад, окремо для мобільних і окремо – для всіх інших); генерація різних версій сайта на боці сервера після аналізу типу пристрою, з якого прийшов запит; використання так званих медіазапитів, описаних у модулі Media Queries специфікації CSS3. Найбільш перспективними є останні два. З урахуванням того, що другий підхід вимагає знання програмування на боці сервера, необхідно розглянути використання медіазапитів.

2.3.2.1. Медіазапити

Медіазапити – це умовні конструкції, що використовують значення характеристик пристрою користувача для вибору варіанту правил CSS. Вони можуть застосовуватися як у тегу LINK для вибору файла з таблицею стилів, так і всередині таблиць для вибору окремих правил.

Умовна конструкція є логічним вираженням, побудованим за допомогою логічних операторів (and, not, ",") з імен типів пристроїв (all – будь-яке, screen – екран монітора і т. п.) і медіафункцій (max-width: 600px означає, що екран пристрою не більше 600 пікселів тощо) і правила (або файла), яке буде вибрано, якщо вираз істинний. Наприклад, логічне вираження, яке приймає значення істинного для смартфона з екраном, видима область якого менше 600 пікселів: handheid and (max-width: 600px). Його можна використовувати і в тегу LINK, і всередині таблиці стилів:

<link rel="stylesheet" media="handheid and (max-width: 600px)" href="sml.css" />
@media handheid and (max-width: 600px) {div {width: 100%;}}

Імена типів пристроїв наведені в табл. 2.8, медіа функції – в табл. 2.9.

Таблиця 2.8

Імена типів пристроїв

Ім'я типу пристрою	Опис
all	всі типи
braille	пристрої, засновані на системі Брайля
embossed	принтери, що використовують для друкування систему Брайля
handheld	смартфони і аналогічні їм апарати
print	принтери та інші друкувальні устрої
projection	проектори
screen	екран монітора
speech	мовні синтезатори
tty	пристрої з фіксованим розміром символів
tv	телевізори

Таблиця 2.9

Основні медіафункції

Ім'я функції	Опис
max-width	ширина не більше, ніж
min-width	ширина не менше, ніж
max-height	висота не більше, ніж
min-height	висота не менше, ніж
max-color	глибина кольору не більше, ніж
min-color	глибина кольору не менше, ніж
max-resolution	дозвіл не більше, ніж
min-resolution	дозвіл не менше, ніж
orientation	орієнтація (landscape, portrait)

За допомогою медіазапитів можна легко створювати адаптивні сторінки, тобто сторінки, які будуть по-різному відформатовані у відображенні на різних за характеристиками пристроях. Наприклад, наступний фрагмент таблиці стилів забезпечить використання шрифтів різного розміру в заголовках на різних за розмірами екранах.

```
@media all and (max-width: 760px) {h1 {font-size: 12pt}}
@media all and (min-width: 760px) and (max-width: 1200px) {h1 {font-size: 16pt}}
@media all and (min-width: 1200px) {h1 {font-size: 20pt}}
```

Таким же чином можна адаптувати сторінку, змінюючи за необхідності ширину, кількість колонок, розміри зображень, тексту й інші властивості. Наступний приклад це демонструє.

Приклад 9. За основу взята сторінка з прикладу 8. За допомогою медіа-запитів вона адаптована до зміни розміру екрана: на великих – це основний розділ у три колонки, на планшетах – у дві, а на смартфонах – одна.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8" />
<title>Адаптивна верстання flex</title>
<style>
body {background-image:url(bg.jpg);}
footer, header, nav {width:100%; text-align:justify;}
header {height: 60px;text-align:center;font-size:2em}
nav {height: 60px;text-align:center;}
nav ul {padding-top: 10px;}
nav ul li {display: inline;}
 nav ul li button {border-radius:10px;}
.main {display: flex;flex-direction:row;text-align:center;}
.sidebar_first { order: 1; flex: 0 0 250px;}
.sidebar_second { order: 3;flex: 0 0 210px;}
.content {order: 2;flex: 1 1 0;}
@media all and (max-width: 1080px) {
                                        .main {flex-direction: row;flex-wrap:wrap;}
                                        .content {flex: 1 1 70%;}
                                        .sidebar_first {flex: 0 1 30%;}
                                        .sidebar_second { flex: 0 0 100%;}
                                        }
@media all and (max-width: 768px) {
                                       .main {flex-direction:column;}
                                       .sidebar_first {flex: 0 0 auto;}
                                       .sidebar_second {flex: 0 0 auto;}
                                       .content {flex: 0 0 auto;}
                                        }
</style>
</head>
<body>
        <header>
        Заголовок (логотип, назва, форма пошуку по сайта, ...)
       </header>
       <nav class="nav">
         <ul>
          <li><a href="#"><button>посилання 1</button></a></li>
```

```
        <li><a href="#"><button>посилання 2</button></a></li>
        <li><a href="#"><button>посилання 3</button></a></li>
        <li><a href="#"><button>посилання 4</button></a></li>
      </ul>
    </nav>
    <div class="main">
        <div class="content">
          <H1>Основний контент</h1>
          …
        </div>
       <div class="sidebar_first">
          <H1>секція 1</h1> <BR>
          …
        </div>
        <div class="sidebar_second">
          <H1>секція 2</h1> <BR>
          …
        </div>
   </div>
   <footer>
      <h1>Підвал</h1>
        …
   </footer>
</body>
</html>
```

На великих екранах (ширина більш, ніж 1080 пікселів) основний розділ (class="main") матимуть три колонки (рис. 2.9).

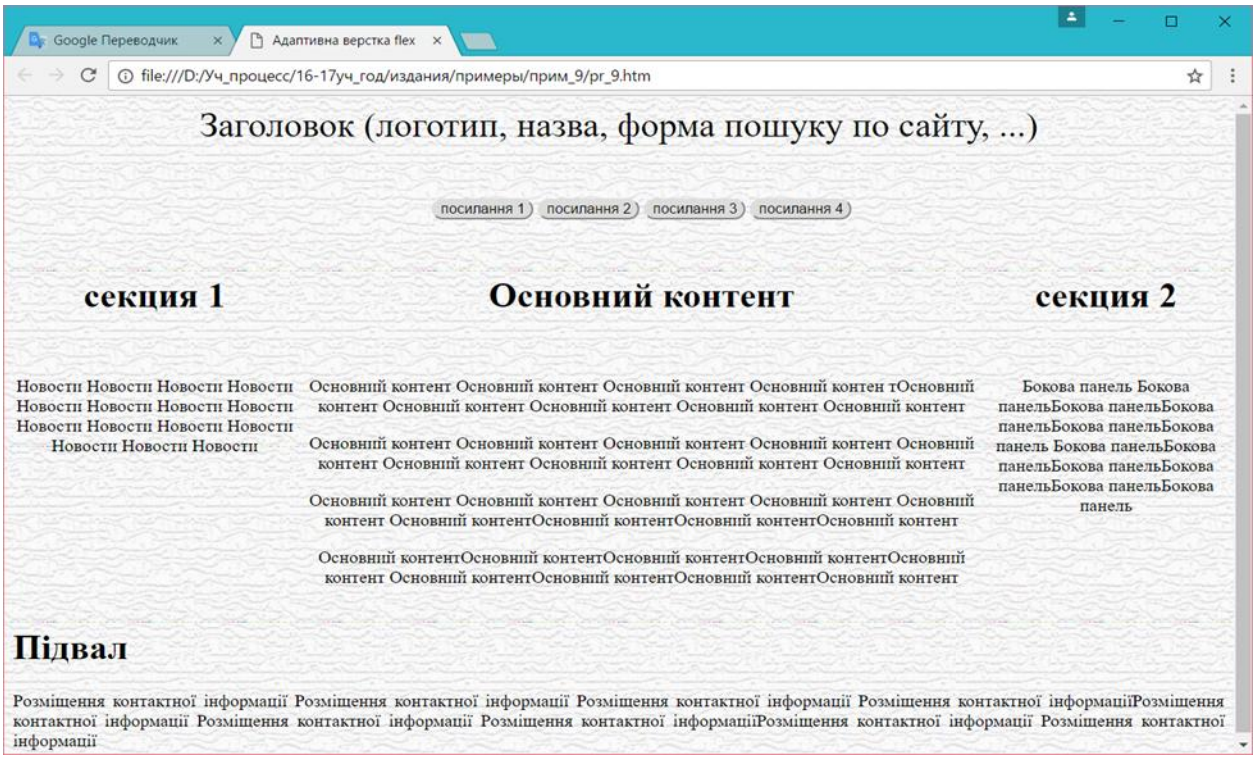

Рис. 2.9. **Вигляд сторінки на великих екранах**

Верстання виконане за допомогою блоків flex, бокові панелі мають фіксовану ширину, не змінюючись зі змінами розмірів вікна до 1080 пікселів.

На екранах розмірів планшетів (ширина більше 768 пікселів, однак менша 1080 пікселів) основний розділ має дві колонки, а права панель розташована знизу на всю ширину вікна (рис. 2.10). Оскільки ширина задана у відсотках, то блоки поводяться як "гумові", смуга прокрутки не виникає.

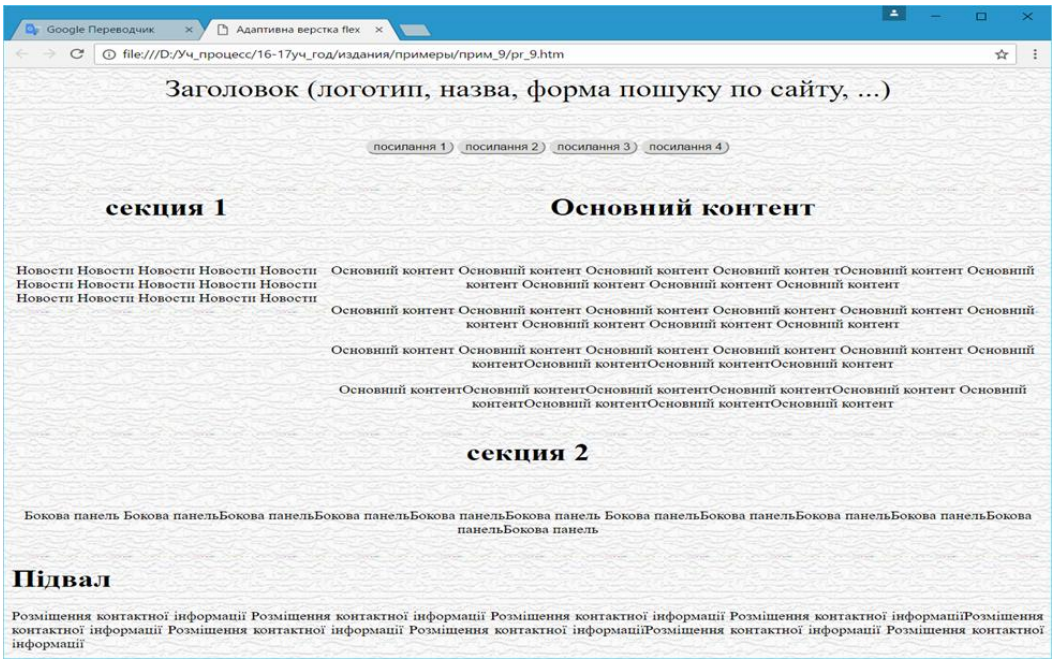

Рис. 2.10. **Вигляд сторінки на екранах планшетів**

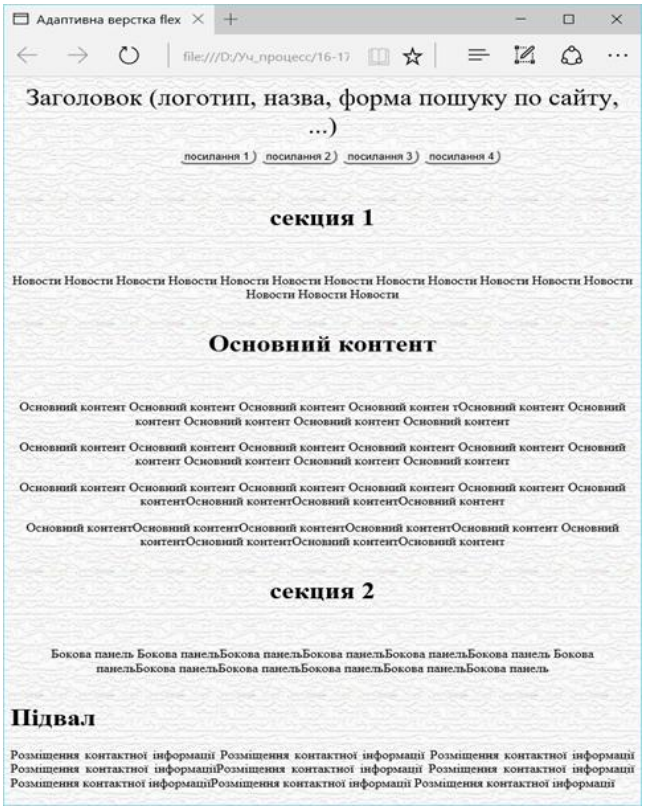

Рис. 2.11. **Вигляд сторінки на екранах смартфонів**

На смартфонах (видима вікна ширина менше, ніж 768 пікселів) усі блоки розташовані в одну колонку (див. рис. 2.11) і заповнюють усю ширину вікна.

2.3.2.2. Фреймворки

Фреймворк (англ. Framework – каркас, структура) – програмне забезпечення, що полегшує розроблення й об'єднання різних компонентів великого програмного проекту. Їх використання є загальною тенденцією в розробленні програмних продуктів.

CSS-фреймворки створюються для спрощення роботи верстальника. Завдяки їх використанню скорочуються терміни розроблення, зменшується кількість помилок верстання, полегшується облік відмінностей відображення в різних браузерах. Вони дозволяють створювати сторінки, які адаптуються до характеристик пристроїв. Приклади CSS-фреймворків: Skeleton, Kube, Blueprint, 960 Grid System, Bootstrap, Topcoat.

Як приклад слід розглянути Bootstrap – один з найвідоміших на сьогоднішній день. Він містить сітку, шаблони для відтворення кнопок, бічних панелей, навігаційних панелей, форм та інших елементів сайта. Bootstrap сумісний з усіма основними сучасними браузерами та підтримує адаптивність.

Ефект від використання досягається за рахунок готових таблиць, що містять різні правила для елементів сторінки. Розробнику достатньо створити код розмітки та вжити необхідні імена класів (значення атрибуту class).

У використанні фреймворка значення атрибута class у тегах є набором імен, розділених пробілами. Будь-яке з них, використане в класовому селекторі, зв'яже правило з цим елементом. Такий підхід дозволяє легко додавати власне форматування. Достатньо написати правило та додати через пробіл ім'я в значення атрибуту.

Для використання фреймворка необхідно підключити таблицю стилів, завантажену з сайта розробника, та дві бібліотеки:

```
<link href="css/bootstrap.min.css" rel="stylesheet">
 <script src="js/jquery.min.js"></script>
 <script src="js/bootstrap.min.js"></script>
```

Вміст, який форматуватиметься цими правилами має знаходитися в контейнері з класом container або container-fluid (для гумового макета).

Макет верстання (сітка) є послідовністю смуг (контейнер з класом row), всередині яких знаходяться колонки (контейнери з префіксом імені класу col-). Кожна смуга поділена на 12 частин, колонці може призначатися від 1 до 12 цих частин, призначення задається в імені класу, наприклад: col-xs-4 (призначено 4 частини). Адаптивність забезпечується системою імен класів: xs (смартфони, ширина менше 768 пікселів), sm (планшети, ширина більше 768 пікселів), md (монітори, ширина більше 992 пікселя) і lg (монітори з високою роздільною здатністю, ширина понад 1200 пікселів).

Таким чином, код сторінки, що складається зі смуги з декількома колонками, може виглядати так:

```
<!doctype html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Макет</title>
  <link href="css/bootstrap.min.css" rel="stylesheet">
 </head>
 <body>
  <div class="container">   /*container-fluid   для резинового макета*/
   <div class="row">
    <div class="col-xs-12 col-sm-6 col-md-3">контент 1 колонки</div>
    <div class="col-xs-12 col-sm-6 col-md-3"> контент 2 колонок</div>
    <div class="col-xs-12 col-sm-6 col-md-3"> контент 3 колонки</div>
    <div class="col-xs-12 col-sm-6 col-md-3"> контент 4 колонки</div>
   </div>
  </div>
  <script src="js/jquery.min.js"></script>
  <script src="js/bootstrap.min.js"></script>
 </body>
</html>
```

На моніторах це буде одна смуга з чотирма колонками, на планшетах – дві смуги по дві колонки, на смартфонах – чотири смуги в одну колонку.

Існує також можливість управління відображенням елементів. Наприклад, можна зробити несуттєві елементи невидимими на малих екранах. Для цього призначені два класи visible та hidden з відповідними суфіксами (xs, sm, md, lg), наприклад: hidden-xs (на смартфонах пункт не відображається), visible-lg (відображається тільки на великих екранах).

Надалі в контейнерах, що відповідають різним областям макета сторінки, розміщуються елементи контента, яким надається необхідний зовнішній вигляд за допомогою присвоєння атрибутам class спеціальних імен. Перелік таких елементів досить великий, від оформлення текстових фрагментів до каруселей слайдів. Імена для часто використовуваних елементів: кнопки; групи кнопок; навігаційні панелі.

Стилі кнопок можуть застосовуватися до елементів <button>, <input type = "button">, а також до посилань <a>. Атрибут class повинен містити ім'я btn і одне з імен, яке визначають вид кнопки (btn-default, btn-primary, btn-success, btn-info, btn-warning, btn-danger, btn-link), наприклад:

```
<a class="btn btn-default" href="#" role="button"> Кнопка </a>
<Button class = "btn btn-default" type = "submit"> Кнопка </ button>
<Input class = "btn btn-default" type = "button" value = "Кнопка">
```

За допомогою імен btn-lg, btn-md, btn-sm, btn-xs можна управляти розміром кнопки.

Для створення групи кнопок необхідно помістити їх в контейнер (наприклад,) і задати ім'я класу btn-group (горизонтальне розташування) або btn-group-vertical (вертикальне розташування):

```
<div class = "btn-group-vertical">
  <button type = "button" class = "btn btn-primary"> Верхня </ button>
  <button type = "button" class = "btn btn-primary"> Нижня </ button>
</div>
```

Навігаційна панель, як правило, є групою елементів, з якими пов'язане гортання сторінок. Найчастіше це набір майданчиків, кнопок, вкладок, які ведуть себе як посилання. Варіантів оформлення досить багато. У найпростішому випадку створюється список, що містить необхідну кількість посилань, для якого вказується ім'я класу nav та ім'я, яке визначає вид панелі (nav-pills – набір кнопок, nav-tabs – набір вкладок і т. д.). Наприклад:

```
<ul class = "nav nav-pills">
  <li class = "active"> <a href="#"> Головна </a> </li>
  <li> <a href="#"> Питання </a> </li>
  <li> <a href="#"> Статті </a> </li>
</ul>
```

Існують стилі і для безлічі інших елементів. У тому випадку, якщо оформлення елемента не влаштовує розробника, легко можна додати власні правила, включивши їх у тег style або підключивши власний файл з таблицею стилів. Відповідні імена додаються в значення атрибуту class через пробіл, не порушуючи загальний каскад правил форматування.

Приклад 10. За основу взята сторінка з прикладу 8. Відмінність макета полягає в зміщенні до лівого краю області заголовка, відступах і заокруглених кутах основних областей, а також наявність загального фону, завдяки чому основні розділи повинні виглядати як аплікації на загальному тлі. Сторінка повинна на смартфонах виглядати в одну колонку, на всіх інших – триколонковою основною частиною.

```
<!DOCTYPE html>
<html>
<head>
        <title>Приклад 10</title>
        <meta content="text/html; charset=utf-8" http-equiv="Content-Type">
        <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">
       <link rel="stylesheet" type="text/css" href="css/my_styles.css">

</head>
<body>
<div class="container-fluid fon">
<heder class="col-xs-10 col-xs-offset-2"><h1>Заголовок</h1></heder>
<div class="col-xs-12 my_nav">
       <button type="button" class="btn btn-primary my_kn"><span
class="glyphicon glyphicon-star"></span> Пункт 1</button>
       <button type="button" class="btn btn-primary my_kn"><span
class="glyphicon glyphicon-map-marker"></span> Пункт 2</button>
       <button type="button" class="btn btn-primary my_kn"><span
class="glyphicon glyphicon-tint"></span> Пункт 3</button>
       <button type="button" class="btn btn-primary my_kn"><span
class="glyphicon glyphicon-align-left"></span> Пункт 4</button>
    </div>
 <div class="row">

   <div class="col-xs-12 col-sm-2">
     2 колонки<br>
      <p>…</p>
    </div>
```

```
    <div class="col-xs-12 col-sm-7">
     7 колонок<br>
     <p>...</p>
    </div>

     <div class="col-xs-12 col-sm-3">
      3 колонки<br>
      <p>...</p>
     </div>

  </div>
  <footer class="col-xs-12"><h1>"ПОДВАЛ"</h1></footer>
</div>
 <script type="text/javascript" src="js/jquery-1.11.3.min.js"></script>
 <script type="text/javascript" src="js/bootstrap.min.js"></script>
 <script type="text/javascript" src="js/my_js.js"></script>
</body>
</html>
```

На 2.12 зображена сторінка без додаткового форматування.

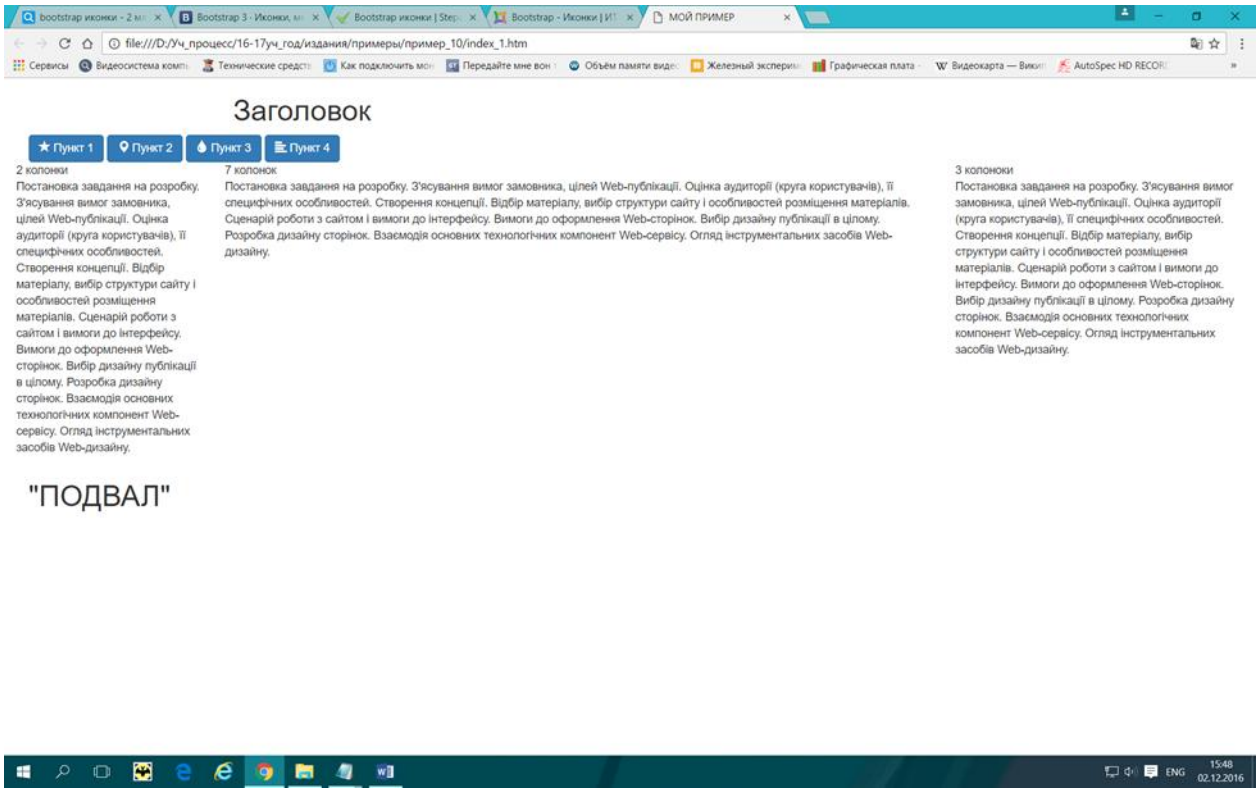

Рис. 2.12. **Вигляд сторінки на екранах без додаткового форматування**

Додаткова таблиця стилів (файл my_styles.css) забезпечує потрібний вигляд (рис. 2.13).

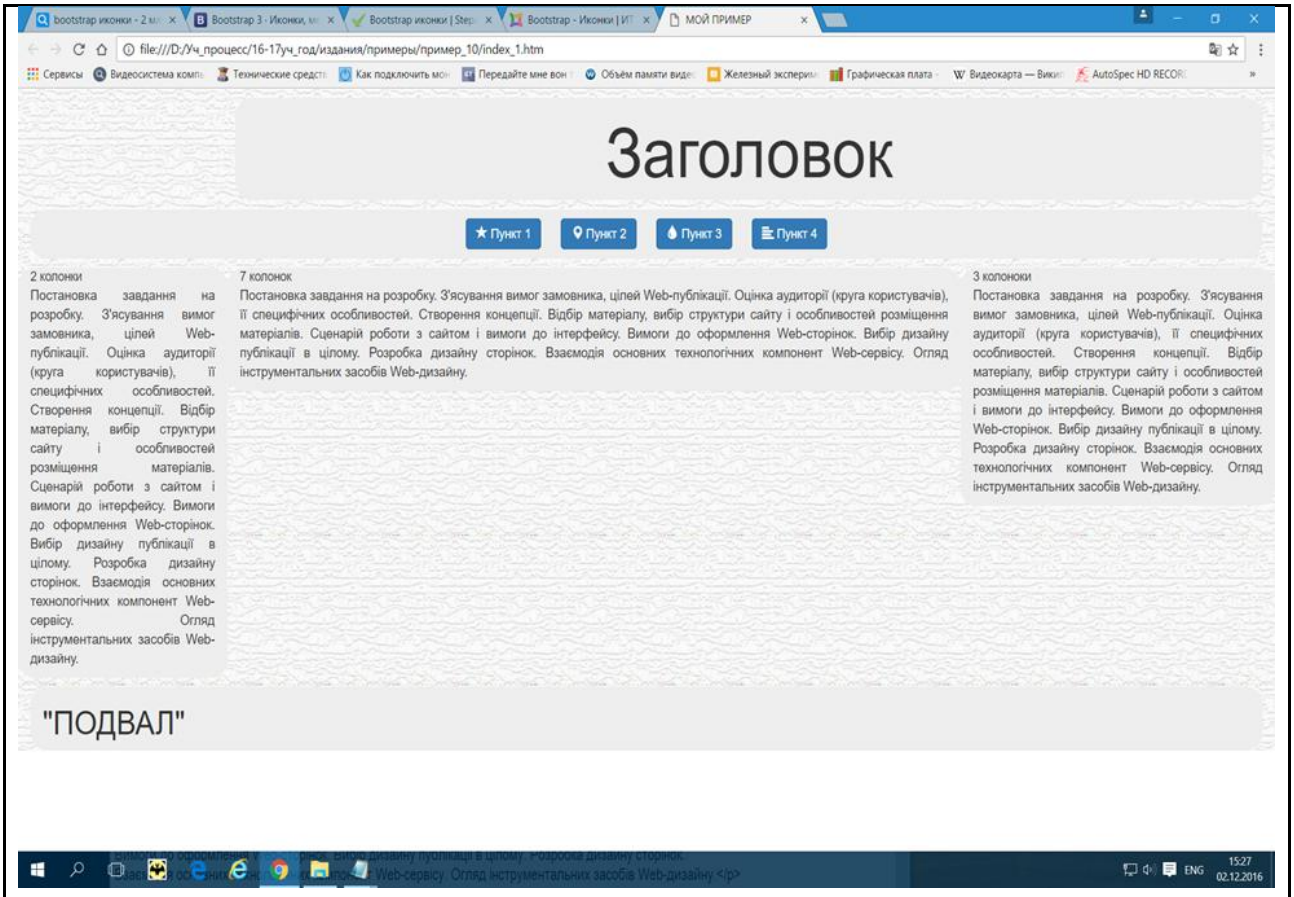

Рис. 2.13. **Вигляд сторінки на екранах після додаткового форматування**

Зміст таблиці стилів:

```
.fon {background-image: url("../bg.jpg")}
[class^="col-"] {background-color: #eee; border-radius: 20px;margin-top:10px;}
heder h1 {text-align:center;font-size:56pt}
p {font-size:15px; text-align:justify}
.my_nav {text-align:center}
.my_kn {margin:10px;display:inline-block;}
```

Для забезпечення додаткового форматування в значення атрибутів class відповідних тегів додані через пробіл імена fon, my_nav, my_kn. Крім того, селектор [class^="col-"] забезпечить форматування всіх елементів, у яких значення атрибута class починається з символів col-.

На закінчення відзначимо, що Bootstrap використовує сучасні напрацювання з CSS і HTML, тому необхідно тестувати його підтримку в різних браузерах, особливо старих версій.

Висновки й узагальнення

Стильові специфікації (CSS) є надзвичайно гнучким і універсальним засобом управління зовнішнім виглядом елементів WEB-сторінки. У ході розроблення зазвичай вирішуються дві взаємно пов'язані завдання: верстання та власне форматування.

Верстання полягає в розміщенні на сторінці досить великих блоків (контейнерів) відповідно до деякого макета та наданні їм необхідних властивостей (наприклад, поведінка у разі зміни розмірів вікна і т. д.). Для вирішення цього завдання можуть використовуватися різні засоби (обтічні блоки, елементи з display:flex, а також різні шаблони та фреймворки). У ході верстання вирішуються й питання адаптації сторінок до умов перегляду.

Для форматування таблиці стилів мають для кожного елемента сторінки багато властивостей, що дозволяють управляти зовнішнім виглядом. На додаток до цих властивостей є можливість створення великої кількості ефектів, включаючи динамічні (такі, як обертання, зміна параметрів – аж до анімації). Усе це дозволяє розробнику створювати оригінально оформлені сторінки.

Теоретичні запитання

1. Що таке селектор?

2. Сформулюйте рекомендації з використання селекторів CLASS і ID. Які ще селектори ви можете назвати для використання на сторінках?

3. Що таке правило?

4. Сформулюйте основні вимоги синтаксису таблиць стилів.

5. Чи можуть декілька тегів мати однакові значення атрибуту ID?

6. У чому полягає основна властивість блокових елементів?

7. Поясніть призначення властивості float. Яких значень воно може набувати?

8. Опишіть можливості, які забезпечує декларація display: flex.

9. Що таке узагальнена властивість? Як записати її значення?

10. За допомогою яких властивостей можна забезпечити подання контенту в кілька колонок?

11. Сформулюйте рекомендації з використання таблиць стилів. У яких випадках їх використовують (зовнішні, в тегах і так далі)?

12. Чи може сторінка містити декілька тегів STYLE? Чи може тег містити декілька атрибутів STYLE?

13. Поясніть, який текст буде отриманий в результаті застосування правила P {font: bold italic large Palatino, serif }? Запишіть його у вигляді набору значень для окремих властивостей.

14. Для чого призначені елементи SPAN і DIV?

15. Чи можна позиціонувати елементи SPAN?

16. Запишіть правило, що забезпечує форматування, еквівалентне тегу B, тегу I.

17. Чи може бути присутнім фон у стрічних елементів?

18. Назвіть декілька застосувань для управління видимістю елементів.

19. Чи викликає зміну видимості елементів зміна в їх взаємному розташуванні?

20. Які властивості мають значення repeat-y? hidden?

21. Як можна задати вид курсору під час наведення його на елемент?

22. Що таке адаптивний дизайн? Які засоби забезпечують адаптивний дизайн?

Контрольні завдання

1. Опишіть зовнішній вигляд сторінки у вікні браузера:

```
<html>
<head>
<title>Завдання</title>
<STYLE>
html {height:100%;}
body {padding:0;width:700px; margin:0 auto;min-height:100%;position:relative;}
#i1 {float:left; width:55%;margin-left:20%;}
#i2 {float:left; width:20%; margin-left:-75%;}
#i3 {position:relative;top:100;overflow:hidden; width:100%;}
#i4 {position:absolute;top:0; right:0;margin:5px;}
</style>
<body>
<div id="i3">
  <div id="i1">
    <h2>Початок</h2>
    <p><img src="pic1.jpg" width=100 height=80 alt="рисунок">
    <p>ТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ
ТЕКСТТЕКСТ ТЕКСТ ТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ
ТЕКСТТЕКСТ
```

```
ТЕКСТТЕКСТ ТЕКСТ  </p>
  </div>
 <div id="i2">
  <h2>Розділи</h2>
  <ul>
   <li><a href="solutions/">Рішення</a>
   <li><a href="partners/">Партнери</a>
   <li><a href="achievments/">Досягнення</a>
   <li><a href="ringtones/">Рингтони!!!</a>
  </ul>
  <form id="i4" action="search/" method="get">
   <p>Пошук по сайта <input type="text"> <button type="submit">Знайти</button>
  </form>
 </div>
</div>
</body>
```

2. Опишіть зовнішній вигляд сторінки у вікні браузера:

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
/* Layout *****************/
@media screen and (min-width: 600px) {#main {float:left;width:55%;margin-
left:20%;}}
@media screen and (min-width: 600px) {#sections {float:left;width:20%;margin-left:-
75%;}}
@media screen and (min-width: 600px) {#news {float:right; width:25%;}}

</style>
</head>
<body>

 <div id="main">
 <H1>ОСНОВНА</h1><BR>
  <P> ТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ
ТЕКСТТЕКСТ
ТЕКСТТЕКСТ ТЕКСТ ТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ
ТЕКСТТЕКСТ </p>
 </div>
 <div id="sections">
 <H1>секція</h1> <BR>
<P> ТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ
ТЕКСТТЕКСТ ТЕКСТ ТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ
```

ТЕКСТТЕКСТ. </p>
** **
** **
**<H1>НОВИНИ</h1>
**
<P> **ТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТ ТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ ТЕКСТТЕКСТ </p>**
** **
</body>
</html>

3. Створіть HTML-документ, що містить розмічений текст, і таблицю стилів, яка забезпечує його уявлення у вигляді, поданому на рис. 2.14.

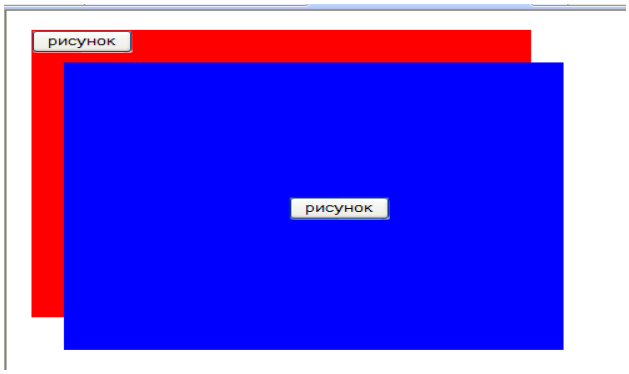

Рис. 2.14. **Вигляд сторінки у вікні браузера**

4. Створіть HTML-документ, що містить розмічений текст, і таблицю стилів, що забезпечує його уявлення у вигляді, поданому на рис. 2.15.

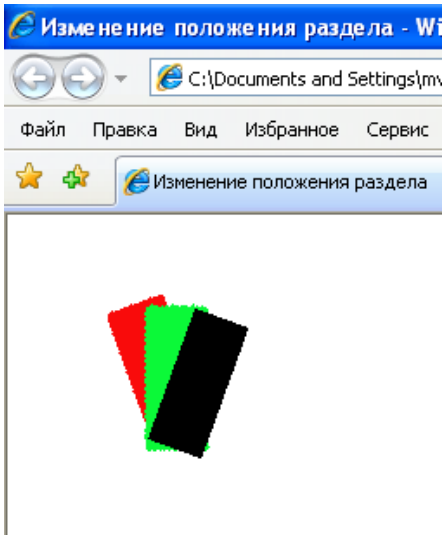

Рис. 2.15. **Вигляд сторінки у вікні браузера**

Розділ 3. Створення динамічних WEB-сторінок

Основна ідея розділу

Розділ присвячений питанням вбудовування програм у WEB-сторінки. Розглядаються особливості синтаксису та виконання сценаріїв мовою JavaScript.

Ключові поняття розділу: сценарій, мова JavaScript, події, обробники подій, DOM, об'єкти, методі об'єктів, властивості об'єктів, засоби доступу до об'єктів.

Питання розділу:

3.1. Основи використання мови JavaScript.

3.1.1. Особливості мови JavaScript.

3.1.2. Виконання сценаріїв.

3.2. Застосування сценаріїв.

3.2.1. Доступ до елементів сторінок.

3.2.2. Доступ до властивостей елементів.

Цілі вивчення розділу

Інформація, подана у розділі, надає студентові можливість сформувати такі **компетентності**:

знання:

основ синтаксису мови JavaScript;

методів включення сценаріїв до сторінок;

об'єктних моделей браузерів і засобів доступу до них;

уміння:

самостійно створювати динамічні сторінки, використовуючи засоби програмування на боці клієнта WWW;

виконувати перевірку та відлагодження створюваних програмних елементів;

комунікації:

обґрунтувати та пояснити основні рішення, пов'язані з розробленням і використанням клієнтських сценаріїв;

автономність і відповідальність:

виконувати самостійний пошук засобів для створення WEB-сторінок і надання їм необхідних властивостей.

Вступ

Загальною тенденцією розвитку ресурсів мережі Інтернет є розширення їх функціональності. Забезпечується це за рахунок розроблення програм, які виконуються на боці сервера або на боці клієнта (в браузері). Для клієнта передбачені два типи таких програм: аплети і сценарії (скрипти). *Аплети* – це відкомпільовані елементи, розміщені в окремих файлах. *Скрипти* – програми, які включаються в сторінку у вигляді вихідних текстів. Їх розробленню присвячений даний розділ.

Як мови для розроблення сценаріїв використовуються JavaScript і Visual Basic Scripting (VBScript). Однак VBScript підтримується тільки в Internet Explorer, тому буде розглянуто тільки JavaScript, як найбільш поширена.

Ця мова є підмножиною мови Java, спрощеною та пристосованою для використання на WEB-сторінках. У міру розвитку технології деякі спрощення почали заважати ефективній розробці із застосуванням сучасних підходів. У зв'язку з цим був розроблений новий стандарт (ECMAScript 5, ES5), який дозволяє подолати цей недолік. Однак виникли проблеми сумісності старих і нових версій ресурсів. Для їх вирішення скрипти, написані з урахуванням нового синтаксису, містять спеціальну директиву "use strict". Незважаючи на всі переваги "суворого режиму", більшість ресурсів використовують звичайний синтаксис, він і буде використаний у подальшому розгляді, за винятком окремо обумовлених випадків.

3.1. Основи використання мови JavaScript

3.1.1. Особливості мови JavaScript

3.1.1.1. Розміщення сценаріїв

Для розміщення кодів сценаріїв у тексті сторінки слугує спеціальний парний тег SCRIPT. Він може містити або текст сценарію, або посилання на файл з розширенням js, що містить сценарій. Сам тег розміщується або в заголовку, або в тілі сторінки, а може і там, і там. Відмінність полягає в доступності імен і можливості їх використання. Рекомендується розміщувати його в заголовку. Крім того, текст сценарію може бути присутнім в інших тегах, наприклад, у посиланнях.

У загальному випадку тег виглядає таким чином:

```
<SCRIPT LANGUAGE="JavaScript ">
// текст сценарію
</SCRIPT >
```

У разі розміщення сценарію в окремому файлі в тег поміщається шлях та ім'я файла, наприклад:

```
<SCRIPT LANGUAGE="JavaScript " SRC="root/my.js">
</SCRIPT >
```

Усередині файла my.js – тільки текст сценарію та ніяких тегів.

Іноді текст сценарію поміщають всередину коментаря, щоб браузери, які не підтримують тег SCRIPT, не виводили його у вікно. Виглядає це таким чином:

```
<html >
<head >
<title>сценарій </title>
<script language="JavaScript ">
<!--
текст сценарію
 //-->
 </script >
</head >
```

Браузер, що підтримує сценарії, витягуватиме сценарій з коментаря, а який не підтримує – пропустить як коментар.

3.1.1.2. Використання змінних

Додавання змінних проводиться шляхом їх оголошення. Вони стають властивостями об'єкта window, хоча їх зміна не змінює зовнішнього вигляду вікна. Змінні можуть бути простими й об'єктами. Усі імена складаються з букв, цифр і знаку підкреслення, вони повинні починатися з букви або знаку підкреслення. Імена JavaScript є чутливим до регістра.

Прості змінні – це ім'я, з яким пов'язане одне значення. Перед використанням змінні мають бути оголошені за допомогою оператора var.

```
var myAge;
var y, x;
```

Оголошення не пов'язане з виділенням пам'яті для значення і задаванням типу. Допускається взагалі не оголошувати змінну, тоді всі пов'язані з оголошенням дії будуть виконані шляхом використання змінної у виразі або операторі. Пам'ять для зберігання значень змінних виділяється, тільки коли визначені самі значення (наприклад, шляхом привласнення), в ході ініціалізації.

```
var myAge = 45;
var mySt="stroka ";
myAge=6.7+5;
var textToWrite = " of " + navigator.appName + ".";
```

Такий підхід забезпечує певні зручності для розробника, але водночас ускладнює відлагодження, оскільки маскує багато помилок.

Тип змінних явно не задається, а визначається за типом привласнюваного значення та може бути динамічно змінений, тобто одній і тій же змінній можна послідовно привласнювати значення різних типів. Підтримується обробка таких типів: цілі, плаваючі, рядкові, логічні. У разі сумісного використання у вираженнях у більшості випадків виконуються неявні перетворення та приведення до одного типу. Хоча є ряд спеціальних функцій перетворення (parseInt () – перетворення в ціле, parseFloat () – перетворення в плаваюче). Під час перетворення в ціле можна вказати підставу (parseInt(x,8) – перетворення у вісімкове, parseInt(x,16) – перетворення у шістнадцяткове). Наприклад:

```
result = parseInt ("42") // привласнене ціле значення 42
result = parseInt ("42.33") // привласнене ціле значення 42
result = ("" + 2500) // привласнений рядок "2500"
```

Рядкові змінні є об'єктами і мають властивість length, тому:

```
result = ("" + 2500).length // привласнене ціле значення 4
```

Для визначення типу значення, привласненого змінний, використовується спеціальний оператор **typeof**, який повертає рядок "number" для цілих і плаваючих, "string" – для рядкових, "boolean" – для логічних, "undefined" – для помилкових і непроініціалізованих (result=typeof x).

Значення змінних можуть задаватися за допомогою констант (літералів) або вводиться у вигляді рядка символів з подальшим перетворенням до необхідного типу.

Константи основних типів задаються таким чином.

Рядки – послідовність символів, взятих в одинарні або подвійні лапки.

```
st="Просто рядок";
st='Ще один рядок';
st='Рядок в "рядку" ';
```

Рядки можуть включати спеціальні символи (комбінації, що позначають службові або недруковані символи): \b – забій, \f – нова сторінка, \n – новий рядок і т. д. Наприклад, рядок "перший рядок\nдругий рядок" буде сприйнятий як два: "перший рядок" і "другий рядок".

Цілі значення – це послідовності цифр зі знаком. Якщо перша цифра 0, то це число у вісімкової системі, якщо 0x – у шістнадцятковій, в решті випадків – у десятковій. Наприклад:

124, 125 – десяткові;

-0124, 075 – вісімкові;

-0x3A75, 0xAB4C – шістнадцяткові (A – 10, B – 11, C – 12, D – 13, E – 14, F – 15).

Для плаваючих значень роздільником цілої та дробової частин є крапка. Наприклад, 7.8 – відповідає 7,8; 45. – відповідає 45,0; 3.1 E-3 – відповідає 3*10-3.

Логічні значення позначаються зарезервованими іменами true та false. Крім того, ціле значення 0 відповідає false, а 1 – true.

У середовищі браузера для введення та виведення значень змінних можна використовувати дві функції **alert()** і **prompt()**. Аргументом першої може бути будь-яка змінна, її значення перетворюється в рядок символів і виводиться у спеціальному вікні. Друга створює вікно з текстовим полем для введення рядка символів, який потім може бути перетворений у значення необхідного типу. Як аргумент може бути присутнім рядок, який буде виведений у вікні як підказка.

```
x=4;
alert (x);
x=parseInt (prompt ("введіть значення x")).
```

Ці функції зручно використовувати для відлагодження скриптів.

3.1.1.3. Вираження мови

Текст сценарію складається з виражень. Вираження використовуються для задання (обчислення) значень змінних. Вони складаються з операторів і операндів, закінчуються крапкою з комою. Операндами є імена та константи, набір операторів приблизно такий, як у мовах Java і C++ (табл. 3.1).

Таблиця 3.1

Основні оператори

Назви	Позначення	Опис
Арифметичні оператори		
	+, -,*, /, %	Чотири арифметичні дії % – узяття залишку
інкремент	++	Збільшення операнда на 1
декремент	--	Зменшення операнда на 1
злиття	+	Об'єднує два рядки
Оператори привласнення		
	a+=b	Еквівалентне a=a+b
	a-=b	Еквівалентне a=a-b
	a*=b	Еквівалентне a=a*b
	a/=b	Еквівалентне a=a/b
	a%=b	Еквівалентне a=a%b
Порозрядні оператори		
інверсія	~	Змінює все 0 на 1, а 1 на 0
і	&	0 і 0, 0 и1, 1 і 0 = 0; 1 і 1 = 1
або	|	1 і 1, 1 і 0, 0 і 1 = 1; 0 і 0 =0
Логічні оператори		
і	&&	Використовуються з операндами логічного типу
або	||	
не	!	
Оператори порівняння		
	==, <=, >=	Результат виконання – значення логічного типу
	<, >	
	!=	

Порядок виконання операторів для обчислення значень не відрізняється від загальноприйнятого в мовах програмування. Оператори виконуються в порядку убування пріоритету, за умов рівного пріоритету – зліва направо, для зміни порядку використовуються круглі дужки.

Окрім операторів, використовуваних для обчислення значень, існує велика група операторів, що слугують для управління. Їх призначення полягає в управлінні потоком обчислень у програмі. Для цього програма

розбивається на блоки за допомогою фігурних дужок {}, вміст блоку (послідовність операторів усередині дужок) розглядається як один оператор. Наприклад: {x=4; y=3; y+=x;}.

Керівним оператором є спеціальна конструкція ключових слів і блоків, що забезпечує різний порядок виконання блоків. Набір їх досить стандартний: умовні оператори, перемикач, оператори циклів.

Умовний оператор (або оператор розгалуження) дозволяє вибрати для виконання один з двох блоків:

```
if (умова) { блок 1 }
else       { блок 2 };
```

Умова – це вираження, результатом якого є значення логічного типу. Якщо умова виконується (результат має значення true), то відбувається виконання блоку 1 і перехід до наступного оператора. Якщо умова не виконується (результат має значення false, null або undefined), то відбувається виконання блоку 2 і перехід до наступного оператора.

Під час запису логічних виражень використовуються оператори порівняння та логічні (табл. 3.2).

Таблиця 3.2

Оператори порівняння і логічні

Оператори	Опис	Оператори	Опис
<	Менше	&&	Логічне І
>	Більше	||	Логічне АБО
==	Рівне	!	Логічне НЕ
<=	Менше, рівне	===	Строго рівне
>=	Більше, рівне	!==	Строго нерівне
!=	Нерівне		

Оператори === і !== відрізняються від останніх тим, що у ході їх виконання не проводиться приведення даних до одного типу, а з різними типами відразу формується значення false для === і true для !==.

У мові JavaScript є ще один умовний оператор, який обчислює один з двох виразів залежно від умови. Його формат:

(умова)?вираження 1:вираження 2;

Наприклад, a=(x>y)?x:y. У даному прикладі змінній a буде привласнено більше із значень x і y.

У тому випадку, якщо обчислювальний процес має більше двох гілок, необхідно використовувати декілька вкладених операторів if else або спеціальний оператор-перемикач switch. Його формат:

```
switch (вираз ) {
case значення1:оператор;
break;
case значення2:оператор;
break;
...
default:оператор;
}
```

Виконанням оператора проводиться обчислення вираження у круглих дужках, а потім результат послідовно порівнюється зі значеннями після ключових слів case. Якщо чергове значення збіжиться, то виконується відповідний оператор. Оператор break, якщо він стоїть, забезпечує припинення виконання наступних операторів. Інакше будуть виконані всі оператори до кінця блоку.

Наступний фрагмент програми забезпечує введення числа та виведення введеного значення словом.

```
x=parseInt (prompt ("введіть число від 1 до 3"));
switch (x){
case 1:alert ("один");
break;
case 2:alert ("два");
break;
case 3:alert ("три");
break;
default:alert ("введено інше число");
}
```

Оператори циклу слугують для опису ділянок програм (блоків), що повторюються. Використовуються три різновиди циклів: з відомим числом повторювань (**for**), з перевіркою умови на початку циклу (**while**) і з перевіркою умови в кінці циклу (**do-while**).

Формат оператора *for*:

```
for (вираження1 ; умова; вираження2)
{
...
}
```

Вираження 1 потрібне для привласнення початкового значення змінній, яка є лічильником числа повторювань. Умова визначає, за якого значення лічильника слід припинити повторення (цикли продовжуються, поки умова виконується). Вираження 2 змінює значення лічильника. Усередині фігурних дужок записується послідовність операторів, яка циклічно виконуватиметься.

Наприклад, наступний фрагмент забезпечує обчислення значення факторіалу:

```
x=parseInt (prompt ("введіть ціле додатне число"));
p=1 ;
for (i=1 ;i<=x;i++) // можна і так for (i=1 ;i<=x;i=i+1)
{p=p*i ;};
alert (p);
```

Цикл **while** забезпечує повторення блоку до тих пір, поки виконується **умова**. Його формат:

```
while (умова)
{...}
```

Передбачається, що змінні, які входять у логічний вираз, змінюються у ході виконання блоку. Наприклад, обчислення факторіалу з використанням цього оператора виглядає так:

```
x=parseInt (prompt ("введіть додатне число"));
p=1;i=1;
while (i<=x )
{p=p*i;
i++ ;};
alert (p);
```

Оператор **do-while** відрізняється від while тим, що умова перевіряється після виконання блоку. Обчислення факторіалу в цьому випадку виглядає так:

```
x=parseInt (prompt ("введіть додатне число"));
p=1 ;
i=1 ;
do
{p=p*i ;
i++ ;}
while (i<=x );
alert (p);
```

Додаткові можливості з організації циклічних обчислень забезпечуються операторами **break** і **continue**. Перший перериває цикл і забезпечує перехід до наступного оператора. Другий забезпечує пропуск усіх операторів до кінця блоку та перехід до перевірки умов.

І останнє, цикли можуть бути вкладеними один в іншій.

3.1.1.4. Масиви

Масив – це набір однотипних елементів, наприклад: усі гіпертекстові посилання даної сторінки, набір всіх картинок на даній сторінці, всі елементи форми і тому подібне.

У JavaScript, як і в мові Java, масиви є об'єктами. Тому масиви створюються поданням значень у квадратних дужках або спеціальним конструктором Array(). Варіанти створення масивів можна подати таким чином:

```
new_array = new Array ();      //масив без елементів
new_array5 = new Array (5);   //масив з п'яти елементів          (нумерація 0...4)
colors=new Array ("red","white","blue ");//масив з трьох елементів
mas=new Array (Array (2),Array (2)); //двовимірний масив
mas= [3, 4, 5, 8];      //масив з чотирьох елементів
mas= [3, 4,, 5, 8];      //масив з п'яти елементів (mas [2] – undefined)
mas= [[3,4],[5,6]];    //двовимірний масив
```

Розмірність масиву може динамічно змінюватися. Можна спочатку визначити масив, а потім привласнити значення одному з його елементів. Як тільки це значення буде привласнене, зміниться і розмірність масиву:

```
colors = new Array (); // масив без елементів
colors [4]= "red ";       // масив з п'яти елементів
```

Звернення до елементів масиву проводиться за індексами. Індекс найчастіше буває цілочисельним, початок відліку 0:

```
colors [0] ="red"; //привласнили значення "red "
colors [0]="blue"; // змінили значення на "blue "
```

Елементами масивів можуть бути інші об'єкти. У документі є масив форм, масив посилань, масив зображень і т. д. До них можна звертатися з використанням індексу.

document.images[1].src="car1.gif" – робота з масивом зображень сторінки (друге зображення на сторінці).

3.1.1.5. Функції

Як і в інших мовах, функції JavaScript – це оформлені спеціальним чином ділянки програми, яким привласнене ім'я. Використовуючи це ім'я, можна виконати пов'язаний з ним код.

Перед використанням функція має бути оголошена. Оголошення функції:

```
function FC (x)
{if (x<=1 ) return 1;
else return x*FC (x-1 );
}
```

Усі змінні, оголошені всередині функції, разом з аргументами є локальними, тобто використовувати їх можна тільки всередині функції.

Параметри передаються за значенням, а не за посиланням. Отже, повертати значення можна, привласнивши його імені функції за допомогою оператора **return** (він завершує виконання функції та привласнює імені значення виразу). Якщо значень декілька, то можна використовувати глобальні змінні.

Звернення до функції:

```
Var x=3 ;
FC (x);
```

Якщо повернене значення не потрібне, то функція викликається з оператором void:

```
void FC (x);
```

Ім'я функції може бути використане в операторі typeof (поверне рядок "function"), а також привласнене змінній (після цього ім'я змінної стане синонімом імені функції).

На закінчення слід зазначити, що JavaScript містить набір вбудованих функцій, серед яких найчастіше використовуються функції перетворення parseInt, parseFloat.

3.1.1.6. Використання об'єктів у мові JavaScript

Відповідно до основної ідеї ООП, **об'єкт** – це інкапсульований набір властивостей і методів. **Властивості** – це змінні (як прості, так і об'єкти), **методи** – функції, що виконують зміни властивостей.

У програмі об'єкти представлені класами (аналог типів у простих змінних) і екземплярами класів (власне об'єкти в пам'яті з конкретними властивостями).

Створення екземпляра об'єкта проводиться **конструктором** – спеціальною функцією, ім'я якої збігається з ім'ям класу. Конструктор повертає посилання на об'єкт, яке і привласнюється імені об'єкта:

```
var NamObj;
NamObj=new ObjClass ();
```

Тут NamObj – ім'я об'єкта, new – оператор, ObjClass () – конструктор класу. Конструктор об'єктів може мати параметри.

Доступ до властивостей і методів об'єкта здійснюється з використанням так званої точкової нотації. Наприклад, NamObj.len – звернення до властивості len, NamObj.fun(); – виклик функції fun(). Якщо властивістю об'єкта є об'єкт, то в ланцюжку імен, зв'язаних крапками, буде і його ім'я.

Об'єкти в JavaScript допускають автоматичне розширення. Можна додати нову властивість в об'єкт, просто призначивши йому значення.

Платою за цю зручність є ускладнення відлагодження, особливо з урахуванням того, що JavaScript є чутливою до регістра. Наприклад:

```
<SCRIPT LANGUAGE="JavaScript ">
alert (document.title);                  // Виведення заголовка документа
document.Title = "Не заголовок"; // Додавання властивості Title.
alert (document.Title);                  // Виведення нової властивості Title.
</SCRIPT >
```

У даному прикладі два повідомлення є різними. Більш того, другий рядок не генерує помилки, хоча вона є – задано неправильне ім'я властивості. Елемент Title додається як властивість об'єкта document. Тому слід бути обережним за умови написання програми на JavaScript. Відлагодження великих сценаріїв може бути дуже складним.

Для вирішення проблем відлагодження Internet Explorer пропонує властивість документа expando, яку можна використати для відключення можливості неявного додавання властивості в JavaScript.

Властивість expando не відключає явне додавання властивостей у вікно шляхом опису змінних, але відключає неявне додавання, як подано на наведеному прикладі:

```
<SCRIPT LANGUAGE="JavaScript ">
document.expando = false ;
var x = 0;              // Додане явно
alert (window.x); // Помилки немає
window.y = 10;    // Помилка – відсутня властивість у
</SCRIPT >
```

Оскільки будь-який об'єкт може містити будь-яку кількість властивостей, то для доступу до них використовується спеціальний оператор циклу **for...in**. З його допомогою операції виконуються для кожної використовуваної властивості в об'єкті. Приведений нижче код виводить імена всіх властивостей об'єкта window:

```
<SCRIPT LANGUAGE="JavaScript ">
// Відображає повідомлення зі всіма властивостями вікна і їх значеннями.
var sProps = "Window Properties \n ";
for (props in window )
sProps += props + ": " + window [props ]+ "\n ";
alert (sProps );
</SCRIPT >
```

У мові JavaScript користувач не може явно створювати власні класи. Під час створення сценаріїв для Web-сторінок цілком достатньо вбудованих класів JavaScript і доступу до об'єктів браузера.

Найчастіше використовуваними класами в JavaScript є Data, Image, Array, String, Math.

Клас Data. Об'єкти **Data** створюються конструктором var d=new Date(). За допомогою використання конструктора без параметрів створюється об'єкт з поточними на момент виконання програми значеннями часу і дати. За наявності параметрів значення формується на їх основі:

```
var sDate=new Date ("Month dd,yyyy hh:mm:ss ")
var sDate=new Date ("Month dd,yyyy ")
var sDate=new Date ("yy,mm,dd,hh,mm,ss ")
var sDate=new Date ("Month yy,mm,dd ")
```

У об'єкта є декілька методів:

sDate.getTime() – м сек від 00:00 1.1.70
sDate.getYear() – рік
sDate.getMonth() – місяць (січень-0)
sDate.getDate () – число місяця
sDate.getDay() – день тижня (понеділок-0)
sDate.getHour() – годинник
sDate.getMinutes() – хвилини
sDate.getSeconds() – секунди
sDate.toLocaleString() – вивід в національному форматі

Клас Image. Конструктор для об'єктів **Image**:

```
new_image = new Image ();
new_image = new Image (width,height);
```

Часто для цілей створення мультиплікації створюють масиви графічних об'єктів, які потім послідовно прокручують:

```
img_array = new Array ();
for (i=0 ;i<10;i++) img_array [i]= new Image (50,100);
```

Цим оператором буде створений масив з десяти зображень, розміром 50x100. Самі зображення ще не визначені, це тільки "заготівка" для їх розміщення.

У об'єкта Image існує багато властивостей, з яких, мабуть, найважливішим є src. Так, для привласнення конкретних картинок елементам масиву img_array можна скористатися такою послідовністю команд:

```
for (i=0 ;i<10;i++) img_array [i].src="dg"+i+".gif";
```

Властивості src кожного елемента масиву будуть послідовно привласнені імена файлів dg0.gif, dg1.gif і так далі.

Об'єкт типу Image можна пойменувати в HTML-теги IMG, використовуючи атрибути NAME або ID.

Після цього можна звертатися до нього за іменем. Слід враховувати, що якщо Image використовується всередині форми, то він є властивістю цієї форми. Це означає, що для наступного графічного об'єкта мають бути використані різні складені імена.

Малюнок в документі: **<IMG NAME="CAR " SRC="CAR.GIF">.**

Звернення до нього: document.car.src="CAR1.GIF".

Малюнок у формі:

```
<form name=kuku>
<IMG NAME=CAR SRC=CAR.GIF>
</form >
```

Звернення до нього: document.kuku.car.src = "CAR1.GIF".

Проте найчастіше в прикладах використання скриптів можна зустріти звернення до Image за індексом у масиві всіх графічних об'єктів даної сторінки. Якщо наш об'єкт, наприклад, є другим Image на сторінці, то, будь він усередині форми або за її межами, до нього завжди можна звернутися за індексом:

```
document.images[1].src = "CAR1.GIF"
```

Клас Array. Масиви в JavaScript є також об'єктами. Для них визначено декілька методів, найчастіше використовуються такі: join, reverse, sort. Join об'єднує елементи масиву в рядок символів, як аргумент в цьому методі задається роздільник:

```
colors = new Array ("red","white","blue");
string = acolors.join("+");
```

У результаті виконання надання значення рядку символів string створюється рядок: "red + white + blue".

Інший метод, reverse, змінює порядок елементів масиву на зворотний, а метод sort відсортовує їх у порядку зростання.

Для масивів визначена властивість length, вона містить число елементів масиву, її можна використовувати у програмі:

```
color = new Array ("red","white","blue");
alert (color.length);
```

Таким чином, об'єктам і масивам у JavaScript властива певна подвійність. Тому об'єкти та масиви надають два методи доступу до їх змісту: пряме посилання на зміст як властивість за допомогою точкової (.) нотації або посилання на індекс у масиві за допомогою дужок ([index]).

Індекс у масиві JavaScript може бути як цілочисельним, так і значенням типу String, яке представляє ім'я властивості.

Точкова нотація дозволяє здійснювати прямий доступ до властивості, коли ім'я властивості відоме заздалегідь. Властивість, що викликається, може бути і значенням змінної. У будь-якому випадку до нього можна звернутися, використовуючи ім'я властивості як індекс:

```
var prop = "title ";
 alert (document.title); //Доступ до назви за допомогою точкової нотації.
 alert (document [prop ]);//Доступ до властивості, на яку посилається
                          //змінна prop.
```

Ще один приклад:

```
var ar = new Array (1,2,3);//Створено масив з трьома елементами
ar.myProp = "Demo "; //Добавлено властивість з ім'ям myProp
alert (ar.myProp); //У вікні виводу – рядок Demo
alert (ar ["myProp "]); //У вікні виводу – рядок Demo
nam="myProp "; //Змінний привласнений рядок myProp
alert (ar [nam ]); //У вікні виводу – рядок Demo
alert (ar [0]); //У вікні виводу – 1
```

Клас Math. Об'єкти цього класу не вимагають створення, його властивостями є математичні константи, а методами – математичні функції. Не дивлячись на те, що на JavaScript рідко розробляються програми обчислювального характеру, в деяких випадках вони можуть виявитися корисними. У табл. 3.3 подані деякі константи, а в табл. 3.4 – функції.

Таблиця 3.3

Математичні константи

Властивість	Опис
E	Константа Ейлера
LN10	lg 10
LN2	ln 2
LOG10E	lg e
PI	число пі

Таблиця 3.4

Основні математичні функції

Метод	Опис	Метод	Опис
Acos()	Арккосинус	Ceil()	Найближче ціле зверху
Asin()	Арксинус	Floor()	Найближче ціле знизу
Atan()	Арктангенс	Round()	Найближче ціле
Cos()	Косинус	Max()	Максимальний з переліку
Sin()	Синус	Min()	Мінімальний з переліку
Exp()	Експонента (ex)	Sqrt()	Корінь квадратний
Log()	Логарифм натуральний	Random()	Випадкове число (0...1)

Звернення до властивостей і методів цього класу проводиться з використанням точкової нотації – Math.ім'я_функцій (аргументи). Наприклад, після виконання оператора **alert(Math.random())** у вікні діалогу з'явиться псевдовипадкове значення, тобто з кожним повторним виконанням воно буде іншим.

Рядкові дані (тип string) також є об'єктами **класу String**. Вони створюються наданням рядкового значення змінній або з використанням конструктора:

```
varSt=new String ("рядок символів");
```

Клас містить властивість length і безліч методів, з яких найчастіше використовують cartAt (i), – повертає символ, що стоїть на і-му місці; indexOf ("...",i) – шукає входження підрядка, пошук починається з позиції і повертає номер позиції початку першого входження.

Під час роботи з рядками можна використовувати регулярні вирази (свого роду шаблони для пошуку певних комбінацій символів). Прикладом використання **регулярного виразу** може слугувати всім відомий шаблон для пошуку файлів *.htm. Регулярні вирази пишуться з використанням спеціального синтаксису, запозиченого з мови Perl. Будуються вони з літералів ("\" – ознака нелітерала; "." **–** будь-який символ; "^" – початок рядка; "$" – кінець рядка; {число} – попередній символ зустрінеться конкретну кількість разів; "*" – попередній символ може зустрічатися скільки завгодно разів або не зустрінеться взагалі; "+" – попередній символ може зустрічатися один і більше разів; "?" – попередній символ зустрінеться один раз або не зустрінеться взагалі і тому подібне) і символів.

Круглі дужки означають, що комбінація має бути запам'ятована. Регулярні вирази полягають в /.../. Не вдаючись до деталей, слід навести приклади простих регулярних виразів:

/www/ або /w{3}/ – три w підряд;

/^www/ або /^w{3}/ – три w на початку рядка;

/.$/ – будь-який символ у кінці рядка;

/\/$/ – слеш в кінці рядка;

/\\$/ – зворотний слеш у кінці рядка.

Для роботи з регулярними виразами об'єкти класу String мають декілька методів. Вони використовують як аргументи регулярні вирази:

match(р.в.) – шукає задані послідовності та повертає масив з результатами пошуку;

replace(р.в.,"строка") – повертає рядок із заміною заданих послідовностей рядком символів;

search(р.в.) – пошук заданої послідовності символів, повертає номер позиції першого входження.

Наприклад, у результаті виконання наступного оператора зворотний слеш у кінці рядка (якщо він є) буде замінений на прямий:

```
St=St.replace(/\\$/,"/");
```

3.1.2. Виконання сценаріїв

Сценарії виконуються **інтерпретатором** – спеціальним програмним модулем, що аналізує початковий код і виконує необхідні дії, без створення виконуваного модуля в машинних кодах.

Оператори сценарію, розміщені в тегу Script, виконуються, коли браузер інтерпретує вміст тега.

Оператори, розміщені у функціях, виконуються після виклику функцій (після звернення до функції). Виклик відбувається або в тегу (там, де записано звернення), або коли відбувається подія, з якою пов'язана функція (така функція називається **обробником події**). Саме через обробку різних подій йде управління виконанням скриптів на WEB-сторінках.

З доступних для обробки подій велика частина пов'язана з елементами сторінки. Найчастіше використовувані події наведені в табл. 3.5.

Таблиця 3.5

Перелік подій

Об'єкт	Подія	Момент виникнення
Window	Onload	Закінчення завантаження сторінки
	Onunload	Відхід зі сторінки для завантаження нового
Елементи сторінки	Onclick	Клацання мишею
	Onmouseout	Відведення курсора з посилання
	Onmouseover	Наведення курсора на посилання
Елементи форми	Onblur	Втрата фокуса
	Onfocus	Отримання фокуса
	Onselect	Вибір тексту всередині поля text, textarea
	Oncange	Зміна значення усередині поля
	Onclick	Клацання мишею на кнопці або перемикачі

Зв'язування обробників з подіями може проводиться за замовчуванням (наприклад, для клацання мишею на посиланні), статично – за допомогою спеціальних атрибутів у тегах елементів або динамічно – в сценаріях з використанням спеціальних методів API браузера.

У разі статичного зв'язування (для написання коду сторінки) в тег елемента додається атрибут відповідної події (onclick, onmouseout, onmouseover і т.д.), значенням якого є ім'я функції-обробника:

```
<Input id = "myBut" type = button value = "Натисни тут" onClick = "foo ();">
```

Динамічне, тобто в ході виконання сценарію, зв'язування може проводиться шляхом присвоєння імені функції відповідної властивості об'єкта сторінки (для цього використовується тільки ім'я, дужки не ставляться):

```
document.all.myBut.onclick = foo;
```

Для динамічного зв'язування можуть використовуватися і спеціальні методи: addEventListener(event, handler) – додає обробник у перелік обробників даного елемента; removeEventListener(event, handler) – видаляє оброблювач з переліку. Параметр event визначає ім'я події, handler – ім'я функції-обробника.

Наприклад, такий рядок коду забезпечить додавання обробника:

document.getElementById ("myBut"). addEventListener ("click", foo);

Використання цих методів зручне тим, що з подією можна пов'язати кілька обробників.

У разі виникнення події та запуску відповідного обробника йому передається в якості параметра об'єкт event, що містить інформацію про подію. Його властивості містять: ім'я події (type); відомості про елемент, з яким пов'язана подія (currentTarget); координати миші (clientX, clientY – для подій, пов'язаних з мишею); код натиснутої клавіші миші (button) тощо.

Приклад 11. Цей приклад демонструє зв'язування обробників з подіями та виконання скриптів.

```
<!DOCTYPE html>
<HTML>
<HEAD>
<TITLE>Виконання скриптів</title>
<head>
<BODY onload="f1();">
<img ID="myImg" src="pic1.jpg" width=200 height=150 ><br>
<input ID="my1" type=button value="розміри"><br>
<input ID="my2" type=button value="подія">
<SCRIPT LANGUAGE="JavaScript">
function f1()
{
document.all.my1.onclick = f2;
document.all.my2.addEventListener ("click", f3)
}
function f2()
{
s=prompt("введіть ширину");
document.all.myImg.width=parseInt(s);
s=prompt("введіть висоту");
document.all.myImg.height=parseInt(s);
}
function f3(e)
{
alert("подія "+e.type + " на " + e.currentTarget);
}
</SCRIPT>
</body>
</html>
```

У цьому прикладі обробники (функції f1, f2, f3) зв'язуються з подіями різними способами. Оброблювач f1 зв'язується з подією закінчення завантаження сторінки (onload) статично через атрибут onload. Сам обробник містить програмний код, що забезпечує динамічне зв'язування обробників f2 і f3 з подіями натисканням кнопок. Обробник f2 забезпечує зміну розмірів зображення, f3 – виведення властивостей об'єкта event.

Події можуть створюватися і програмно. Наприклад, з використанням таймера. Зв'язування з обробником проводиться за допомогою спеціальних методів. Для роботи з таймером передбачено декілька методів:

id=setInterval("ім'я_фун",n) – установлює період, з яким таймер видаватиме сигнал, n – період, мсек., "ім'я_фун" – ім'я обробника;

id=setTimeout("ім'я_фун",n) – установлює час, через який буде викликана функція;

clearInterval(id) – скидання таймера;

clearTimeout(id) – скидання таймера.

Запуск таймера може здійснюватися під час обробки інших подій або безпосередньо у скрипті. Обробка події відбувається у викликаній функції, туди ж зазвичай включають і управління таймером.

3.2. Застосування сценаріїв

Основна ідея застосування скриптів полягає в можливості зміни значень атрибутів і властивостей елементів HTML-сторінок,проте При цьому перезавантаження сторінки не відбувається. На практиці це виражається в тому, що можна, наприклад, змінити колір фону сторінки або інтегровану в документ картинку, відкрити нове вікно або видати попередження.

Для дії на вікно і властивості елементів у сценарії використовують властивості та методи відповідних об'єктів браузера. **Властивості** – це спеціальні змінні, значення яких визначають зовнішній вигляд та інші характеристики об'єктів (вікна та розташовані в них елементи).

Методи – це функції для виконання різних дій з властивостями. Порядок і момент виконання методів задаються їх використанням у тексті сценарію.

3.2.1. Доступ до елементів сторінок

Для управління браузером і його вікном з відображуваною WEB-сторінкою використовують відповідні об'єкти, що утворюють ієрархічну структуру. Ця структура получила назву **об'єктна модель браузера** (рис. 3.1).

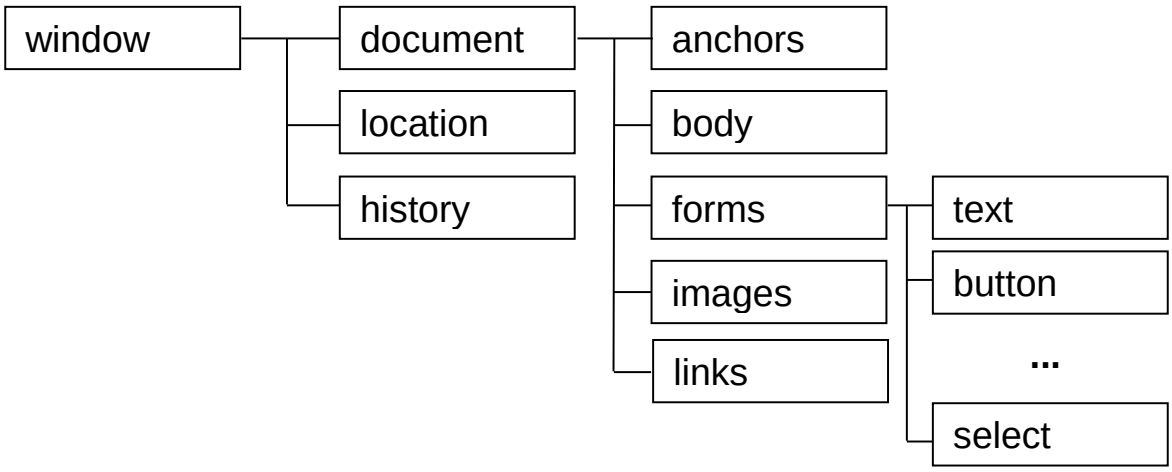

Рис. 3.1. **Ієрархія основних об'єктів браузера**

На самому верхньому рівні знаходиться об'єкт window (вікно). Він містить загальні для всього вікна властивості – такі, наприклад, як defaultStatus і status (визначають текст, що розміщується в рядку статусу). Об'єкт location містить адресну інформацію про активну сторінку, а history – історію переміщення за сторінками.

Об'єкт Document містить властивості, що визначають вид сторінки.

Об'єкти links, anchors, forms і images є властивостями об'єкта Document і масивами посилань, якорів, форм і зображень. Ієрархія об'єктів використовується для доступу до об'єктів, які відповідають конкретним елементам. Для отримання доступу до елемента або його властивостей необхідно визначити місце об'єкта, відповідного елементу, в ієрархії, та задати його ім'я. Наприклад: window.document.images[0].src – адреса найпершого малюнка на сторінці.

Якщо бути точними, то всі ці об'єкти (посилання, форми і т. д.) є колекціями (наборами об'єктів), і доступ до них забезпечується з використанням методу item (), аргументом якого є номер елемента в колекції або його ID. Проте ім'я методу можна опускати. У зв'язку з цим можлива певна плутанина, оскільки допускається використання таких різних синтаксичних конструкцій.

Наприклад, можна вважати images за масив зображень і отримати доступ до властивості width найпершого зображення на сторінці так: document.images[0].width. Але до цієї властивості можна звернутися і так: document.images.item(0).width, вважаючи images за колекцію об'єктів, або так: document.images(0).width, або за ім'ям, якщо воно відоме, document.images("im1 ").width, або за ім'ям, але вважаючи images за асоціативний масив, document.images["im1"].width. Слід нагадати, що до елементів можна звертатися і з використанням точкової нотації document.images.im1.width.

Це, з одного боку, забезпечує деякі зручності (гнучкість у вибиранні образотворчих засобів), але, з іншого – можуть виникнути проблеми, якщо якийсь браузер чогось не підтримує, що часто трапляється.

Об'єктна модель браузера дозволяє отримати доступ до елементів сторінки – таких, як зображення, посилання і т. п. Крім того, є можливість доступу до інших елементів за їх id через спеціальну колекцію all: document.all.myImg (буде обраний елемент з атрибутом id="myImg").

Однак об'єктні моделі різних браузерів мають відмінності, які необхідно враховувати у розробці. У зв'язку з цим для доступу до властивостей елементів краще використовувати DOM (Document Object Model).

Об'єкт document містить опис документа у вигляді дерева вкладених елементів і набору методів для доступу до цих елементів і їх зміни. Ця структура отримала назву "об'єктна модель" документа (DOM). Використовуючи DOM, можна змінювати властивості елементів, видаляти елементи, додавати нові. Відповідно до цих змін буде змінюватися вигляд сторінки у вікні браузера. DOM стандартизована, і основні браузери підтримують цей стандарт. Тому доступ до елементів з використання DOM більш універсальний.

Для роботи з DOM є значна кількість методів (переміщення по дереву від елемента до елемента, видалення елементів, додавання нових і т. д.). Усі вони є методами об'єкта document. Для доступу до властивостей елементів сторінки найбільш популярні: getElementById, getElementsByName, getElementsByTagName, getElementsByClassName.

Метод getElementById() виконує пошук за атрибутом id і повертає посилання на знайдений елемент. Значення id має бути унікальним. Наприклад, var el= getElementById("idn").

Метод getElementsByName() виконує пошук за атрибутом name, повертаючи посилання на колекцію знайдених елементів. Тому для подальшого використання потрібно обрати члена колекції, наприклад, так var el=getElementsByName("namn")[i], або з використанням метода item var el= getElementsByName("namn").item(i).

Метод getElementsByTagName() виконує пошук за іменем тега та повертає посилання на колекцію знайдених елементів.

Метод getElementsByClassName() виконує пошук за значенням атрибута class і повертає посилання на колекцію знайдених елементів.

Наступний фрагмент коду демонструє використання цих методів.

```
...
<p name="np" id="ip1" class="cp">Зміст абзацу1</p>
<p name="np" id="ip2" class="cp">Зміст абзацу2</p>
<SCRIPT LANGUAGE="JavaScript">
alert(document.getElementById("ip1").innerText);
alert(document.getElementsByTagName("p")[0].innerText);
alert(document.getElementsByName("np").item(0).innerText);
alert(document.getElementsByClassName("cp")[0].innerText);
</SCRIPT>
...
```

Сценарій містить чотири звернення до функції виведення. У кожному здійснюється доступ до елемента одним з описаних методів і виведення у вікно текстового вмісту.

3.2.2. Доступ до властивостей елементів

Отже, будь-який елемент на сторінці представлений об'єктом DOM, тобто своїм набором властивостей і методів. Після здобуття доступу до елемента стають доступними його методи та властивості. Методи, як правило, загальні для більшості елементів:

getAttribute(name_atr, pr) – отримання значення атрибута за його ім'ям, pr={true,false} – враховувати чи ні регістр для завдання імені;

setAttribute(name_atr, val,pr) – завдання значення (val) атрибута;

scrollIntoView(pr) – прокрутка у вікні, що робить елемент видимим, pr={true,false} – біля верхнього або нижнього краю вікна.

Крім того, складні елементи мають додаткові методи, наприклад, insertRow для вставки рядків у таблицях.

Більшість елементів володіють також рядом загальних властивостей:

all – посилання на колекцію дочірніх елементів;

className – ім'я класу, пов'язане з таблицею стилів;

clientHeight – висота елемента (доступно лише для читання);

clientWidth – ширина елемента (доступно лише для читання);

clientLeft – товщина лівої рамки елемента (доступно лише для читання);

clientTop – товщина верхньої рамки елемента елемента (доступно лише для читання);

id – ідентифікатор елемента відповідно до атрибута id;

innerHTML – вміст разом з тегами дочірніх елементів;

innerText – текст, включаючи тексти дочірніх елементів без тегів;

outerHTML – вміст, включаючи теги, які створюють елемент;

outerText – текст із тегами, включаючи теги, які створюють елемент.

Крім того, кожен елемент має властивості, які співпадають з атрибутами тегів: width, height, src, href, style та ін.

Наступний фрагмент коду демонструє використання властивостей і методів елементів.

```
…
<style>
P.c1 {color:red}
</style>
<SCRIPT LANGUAGE="JavaScript">
function f1()
{
el=document.getElementById("p1"); // доступ до елемента
s=el.getAttribute("class")+" c1";       // формування значення атрибута
el.setAttribute("class", s) ;               // привласнення значення атрибута
}
</SCRIPT>
…
<P id="p1" class="c1"> перший абзац тексту</p>
<P id="p2" class="c2"> другий абзац тексту</p>
<INPUT TYPE="button" value="додати клас" onClick="f1();">
…
```

Основним засобом форматування HTML-документів є таблиці стилів. У мові JavaScript передбачені засоби роботи як з таблицями в цілому, так і зі стилями окремих елементів.

Для роботи з таблицями стилів використовується об'єкт **styleSheets** – колекція таблиць стилів, які створені за допомогою тегів style, link або директиви @import. Кожній таблиці відповідає окремий член колекції. Властивостями членів можуть бути href (якщо таблиця з файлу), id (якщо таблиця з тегу style), disabled (false – дозвіл на використання таблиці, true – заборона) і type (визначає синтаксис таблиць text/css або text/javascript).

Наприклад, можна змінити таблицю стилів таким чином:

```
document.styleSheets(0).href="sheet1.css";
```

Для доступу до стилю елемента, який заданий в тегу style або присвоєний в сценарії, використовується властивість style. У DOM властивість style елемента є об'єктом, який містить правила форматування.

З урахуванням особливостей синтаксису JavaScript імена властивостей записуються без роздільника "-", а початок кожної складової частини імені, окрім першого, починається із головної літери. Наприклад, в CSS ім'я властивості пишеться border-bottom-color, а в скрипті ця ж властивість має ім'я borderBottomColor. Винятком є властивість float, її слід записувати як cssFloat. Значення всіх властивостей є даними рядкового типу.

Використовуючи об'єкт **style** і його властивості, можна динамічно змінювати зовнішній вигляд конкретних елементів. Наприклад, наступний код змінює ширину елемента та додає йому обтічність:

```
var el=document.getElementById("im2");
el.style.width="200px";
el.style.cssFloat="left";
```

Досить часто для виконання сценаріїв потрібно використовувати значення властивості, сформоване з урахуванням усього каскаду таблиць стилів (поточне використане значення). Для цього може бути використаний метод window.getComputedStyle. Метод повертає об'єкт, що містить усі застосовані властивості. Наприклад, наступний фрагмент коду зменшить ширину обраного елемента вдвічі:

```
var el=document.getElementById("im1");
var styl=window.getComputedStyle(el);
el.style.width=parseInt(styl.width)/2;
```

Необхідно звернути увагу на використання отримуваних значень. Оскільки всі вони рядкового типу, то для виконання обчислень було потрібно явне перетворення до цілого типу.

Слід зауважити, що об'єктні моделі у різних браузерів розрізняються дуже суттєво. Усе це необхідно враховувати під час розроблення скрипів для різних браузерів.

Приклад 12. Даний приклад демонструє використання властивостей і методів об'єктів DOM для створення форми, яка управляє фоном сторінки. Користувачеві надається можливість вибрати фоновий малюнок або монохромний колір.

```
<!DOCTYPE html>
<html>
<head>
 <title>Сторінка з вибором фону </title>
<script>
var fon;
function f1()
{
fon=document.body.background;                                    //запам'ятати фон
document.getElementById("vid").style.visibility="hidden"; //скрити поле
var el=document.getElementsByTagName("select")["menu"];
el.addEventListener("change",f2);                                 //назначити обробник
document.getElementsByName("col")[0].onchange=f5;       //назначити обробник
document.getElementById("frm").style.position="absolute"; // позиціювання
document.getElementById("frm").style.right="0px";              // форми
document.getElementById("frm").style.bottom="0px";          //
}
function f2()
{
var i=document.forms["sel"].elements["menu"].selectedIndex;
switch (i){
       case 0: f4();break;
       case 1: f3("1.jpg"); break;
       case 2: f3("2.jpg"); break;
       case 3: f3("3.jpg"); break;
       case 4: document.getElementById("vid").style.visibility="visible";break;
         }
function f3(im)
{
document.body.style.backgroundImage="url("+im+")";     //додати фон стилем
document.getElementById("vid").style.visibility="hidden";//скрити поле
}}
function f4()
{
document.getElementById("vid").style.visibility="hidden"; //скрити поле
document.body.style.backgroundImage="";     //видалити фон, заданий
                                                                  //стилем
document.body.background=fon;                  //додати фон значенням атрибуту
}

function f5()
{
document.body.background="";                  //видалити фон, заданий
```

```
                                                    //атрибутом
document.body.style.backgroundImage=""; //видалити фон заданий стилем
document.body.style.backgroundColor=document.forms["sel"].elements["col"].va
lue;//встановити колір фону
}
</script>
 </head>
<body background="4.jpg" onLoad="f1();">
<div id="frm">
<form name="sel">
  <fieldset>
    <legend>Керування фоном</legend>
     <p><label>Вибір фону</label> <select name="menu">
                                          <option selected>замовчування
                                          <option>замок
                                          <option>зима
                                          <option>літо
                                          <option>монохромний
                                       </select></p>
<p id="vid"><label>Вибір кольору</label><input type="color" name="col"></p>
  </fieldset>
</form>
</div>
</body>
</html></HTML>
```

Функції f1, f4 і f5 пов'язані з подіями; f2 і f3 викликаються всередині скрипта. f1 викликається після закінчення завантаження сторінки та виконує початкові налаштування. f4 викликається зі зміною значення в поле зі списком фонових малюнків, і виконує зміну малюнка. f5 викликається зі зміною обраного монохромного фону та виконує його установку.

Висновки й узагальнення

Використання сценаріїв на WEB-сторінках є досить простим (для ознайомлених з основами програмування) і забезпечує дуже широкі можливості зі зміні зовнішнього вигляду й адаптації сторінки до умов перегляду та дій користувача. Тому в мережі практично відсутні сторінки, що не містять сценаріїв.

Шляхом розроблення сценарію вирішуються два завдання: визначення моменту виконання та доступ до властивостей елемента, який необхідно змінити. Перше завдання вирішується шляхом призначення обробника відповідного події. Друге – з використанням об'єктної моделі браузера або документа.

Теоретичні запитання

1. Які програмні елементи може містити Web-сторінка?

2. Скільки тегів SCRIPT може містити Web-сторінка?

3. Дані яких типів можна обробляти з використанням скриптів?

4. Що називають вираженням у JavaScript? Як одне вираження відділяється від іншого?

5. Що таке блок?

6. Запишіть ім'я функції для перетворення даних у рядок.

7. Запишіть приклади можливих констант, допустимих в JavaScript.

8. Як визначити тип даних, привласнених змінній?

9. Запишіть приклади різних операторів привласнення.

10. Запишіть формат і поясніть особливості виконання оператора switch.

11. Запишіть і поясніть особливості виконання різних форм оператора while.

12. Чи можуть у тегу SCRIPT міститися виконувані оператори? Виклики функцій? Якщо так, то вкажіть момент і число повторень їх виконання.

13. Якими способами можна передати значення всередину функції?

14. Запишіть фрагмент програми, що містить глобальні та локальні змінні.

15. Назвіть способи скріплення обробників з подіями.

16. Запишіть оператор, використовуючи який, можна набути значення поточної дати та часу.

17. Запишіть ім'я об'єкта, що забезпечує доступ до властивості видимості елемента.

18. Запишіть ім'я об'єкта, що забезпечує доступ до розміру зображення, до адреси зображення.

19. За допомогою якого атрибута можна ідентифікувати форму, елемент форми?

20. Що таке об'єкт з ім'ям all? Як і де він використовується?

21. Як получити значення властивості, що задана у кількох таблицях стилів?

22. Який тип мають значення властивостей об'єкта style?

23. Для чого потрібен таймер?

24. Поясніть механізм використання подій, пов'язаних з таймером.

25. Назвіть методи об'єкта Window.

Контрольні завдання

1. Проаналізуйте код, опишіть дію, знайдіть помилки та налагодьте скрипт:

```
<HTML>
<HEAD> <TITLE>Завдання 1</title>
<SCRIPT>
function f1()
{
 setTimeout ("f2()",500);
}
function f2()
{
im.style.width=parseInt(im.style.width)*0.5;
im.style.height=0.5*parseInt(im.style.height);
im.style.top=document.body.offsetHeight/2-25;
im.style.left=document.body.offsetWidth/2-25;
}
</script>
</head>
<BODY onLoad="f1();">
<img ID="im" src="690.jpg" STYLE="position:absolute;witdh=100;height=100">
</body>
</html>
```

2. Проаналізуйте код, опишіть дію, знайдіть помилки та налагодьте скрипт:

```
<HTML>
<HEAD>
<TITLE>Завдання 2</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function f1()
{
var ob;
ob=event.target;
if (ob.tagName == "DIV")
{
P1.style.zIndex=f2(P1.style.zIndex);
P2.style.zIndex=f2(P2.style.zIndex);
```

```
P3.style.zIndex=f2(P3.style.zIndex);
P4.style.zIndex=f2(P4.style.zIndex);}
}
function f2(x)
{
x=x+1;
if (x>4) x=1;
return x;
}
</SCRIPT>
</HEAD>
<BODY onclick="f1();">
<DIV id="P1"
style="position:absolute;width:400;height:300;left:20;top:20;background-
color:red;z-index:1">
<p> Text 1</p>
</div>
<DIV id="P2"
style="position:absolute;width:400;height:300;left:20;top:20;background-
color:red;z-index:2">
<p> Text 2</p>
</div>
<DIV id="P3"
style="position:absolute;width:400;height:300;left:20;top:20;background-
color:red;z-index:3">
<p> Text 3</p>
</div>
<DIV id="P4"
style="position:absolute;width:400;height:300;left:20;top:20;background-
color:red;z-index:4">
<p> Text 4</p>
</div>
</BODY>
</HTML>
```

3. Проаналізуйте код, опишіть дію, знайдіть помилки та налагодьте скрипт:

```
<HTML>
<HEAD>
<TITLE>Завдання 3</TITLE>
```

```
<SCRIPT LANGUAGE="JavaScript">
x=0;
function f1()
{
x=x+0.00628;
if (x>6.28) x=0;
i1.style.left=200+200*sin(x);
i1.style.top=200*(1-cos(x));
setTimeout("f1()",10);
}
</SCRIPT>
</HEAD>
<BODY onload="setTimeout('f1()',10);">
<IMG src=690.jpg ID=i1 style="position:absolute;width:40;height:40;left:200;top:0;z-
index:2">
<DIV ID=P2
style="position:absolute;width:400;height:300;left:40;top:40;background-
color:blue">
</div>
</BODY>
</HTML>
```

4. Проаналізуйте код, опишіть дію, знайдіть помилки та налагодьте скрипт:

```
<html>
<head>
<title>Завдання 4</title>
<head>
<BODY bgcolor=white>
<DIV ID=rt STYLE="background:black; width:100; height:100;"
onClick="document.rt.style.background='white'"></div>
<FORM NAME=f>
<input id=tt type=text onClick="document.form[0].elements[0].value='jjj';">
<input type=button onClick="document.images[0].width=50;">
<img id="im1" src="690.jpg" width=150>
</form>
<a href="javascript:void();"
onClick="javascript:window.document.im1.width=100;">bbbbbb</a>
</body>
</html>
```

Розділ 4. Засоби створення WEB-сайтів

Основна ідея розділу

Основними напрямами застосування сценаріїв є створення динамічних ефектів (різних галерей, меню і т.п.), підтримка призначеного для користувача інтерфейсу, здійснення обчислень на боці клієнта (реалізація прикладних алгоритмів) і взаємодія з сервером. Для забезпечення ефективності таких розробок створено значну кількість різноманітних бібліотек і фреймворків. Їх використанню і присвячений розділ.

Ключові поняття розділу: об'єкти DOM, властивості та методи об'єктів, динамічна графіка, бібліотеки, jQuery, плагіни jQuery, хостинг, типи хостингу, публікація сайта, тестування сайта.

Питання розділу:

4.1. Створення динамічних елементів та ефектів.

4.1.1. Створення динамічних сторінок.

4.1.2. Створення ефектів.

4.2. Публікація WEB-сайта.

4.2.1. Розміщення сайта у мережі.

4.2.2. Тестування та супроводження сайта.

Цілі вивчення розділу

Інформація, подана у розділі, надає студентові можливість сформувати такі **компетентності**:

знання:

сучасних технологічних засобів створення динамічних сторінок;

найбільш поширених бібліотек, їх можливостей;

програм і методів створення документів для сервісу WWW;

дій з публікації WEB-сайтів і вимог до хостінгу;

уміння:

створювати динамічні сторінки в різних середовищах;

використовувати найбільш поширені бібліотеки та фреймворки;

створювати WEB-сайти в середовищі сучасних засобів;

розміщувати створені документи в мережі Інтернет;

комунікації:
взаємодія з замовником у ході прийняття замовлення на розроблення;
взаємодія з розробником під час супроводження розроблення;
обґрунтування та пояснення рішень, прийнятих членами команди;
автономність і відповідальність:
пошук альтернативних засобів для створення WEB-сторінок;
самостійний пошук і вибір засобів для створення WEB-сторінок;
самостійний пошук і вибір місця розміщення сайта.

Вступ

Сценарії, вбудовувані у WEB-сторінки, надають розробникові повний контроль над змістом і оформленням сторінки. Крім того, з'являється можливість реалізації інтерактивного інтерфейсу з користувачем. Розроблення таких динамічних елементів є творчим процесом, який вимагає не тільки знання основ програмування, але і використання технологічних засобів, що полегшують процес.

Такими засобами є шаблони, бібліотеки, фреймворки, різні середовища розробки. Вони забезпечують скорочення часу розроблення та дозволяють отримати більш якісний результат. У цьому розділі буде розглянуто використання найбільш популярної бібліотеки – jQuery,

Заключним етапом розроблення є тестування і публікація створеного ресурсу в мережі Інтернет. Основні дії, які виконуються на цьому етапі, також розглянуті в цьому розділі.

4.1. Створення динамічних елементів та ефектів

4.1.1. Створення динамічних сторінок

Загальний підхід до створення динамічних сторінок на боці клієнта полягає в зміні властивостей об'єктів, відповідних елементам, у відповідь на події, що відбуваються. Події відбуваються з виконанням користувачем дій з мишею або клавіатурою, внаслідок внутрішньої динаміки обчислювальної системи (закінчення завантаження сторінки або елемента, втрата фокусу і т. д.), у тому числі і надходження сигналів від таймера.

Приклад 13. Приклад демонструє використання властивостей і методів об'єктів DOM для створення розділу, який може переміщатися мишею на сторінці.

```
<HTML>
<HEAD>
<TITLE>Переміщення елемента</TITLE>
<SCRIPT>
var sx; var sy; var be=0;
function f1()
{var ob=event;
if (be==1) {document.getElementById("i1").style.left=parseInt(ob.x)-sx;
            document.getElementById("i1").style.top=parseInt(ob.y)-sy;  }}
function f2()
{var ob=event;
if (ob.button == 0)
{sx=parseInt(ob.offsetX);   sy=parseInt(ob.offsetY);be=1; }}
function f3()
{be=0; }
</SCRIPT>
</HEAD>
<BODY onMousemove="f1();">
<div id="i1" STYLE="background-
color:red;position:absolute;left:100;top:100;width:200;height:200"
onmouseup="f3();" onmousedown="f2();">
 Об'єкт для переміщення
</div>
</BODY>
</HTML>
```

На сторінці оголошені три функції: f1, f2 і f3. Перша зроблена обробником події переміщення курсора у вікні (атрибут onMousemove в тегі BODY). Друга – натисканням клавіші на розділі, якій буде переміщуватися, третя – відпусканням натиснутої клавіші.

Глобальні змінні sx і sy слугують для запам'ятовування положення курсора, be – ознаки натиснутої клавіші. Натисканням клавіші на розділі викликається f2, аналізується код натиснутої клавіші, запам'ятовуються координати курсору й ознака натискання клавіші. Переміщення миші викличе обробник f1 і змістить розділ. Відпусканням клавіші буде викликаний f3, відбудеться обнулення змінної be та переміщення припиниться.

Використання графіки на сторінках має статичний характер, тобто зображення зберігаються на сервері та вбудовуються в сторінку в ході розмітки. Однак є можливість їх динамічного створення під час виконання сценаріїв. Для цього існує спеціальний елемент, створюваний тегом canvas.

Тег canvas призначений для створення в вікні браузера області малювання (полотна), в якій за допомогою сценаріїв можна малювати різні елементарні фігури. Створюване зображення є растровим, проте може змінюватися очищенням областей і виведенням нових фігур.

Тег парний: текст, що міститься всередині, інтерпретується, якщо браузер не підтримує можливості, пов'язані з canvas. Основних атрибутів три: id, height і width. Елемент можна форматувати за допомогою таблиць стилів.

Таких областей може бути кілька, для вибору використовується значення атрибута id (наприклад, за допомогою getElementById):

```
var canvas = document.getElementById ( 'canva');
```

Перед початком малювання встановлюється режим (2d або 3d). Нині підтримується тільки 2d. Отримане шляхом установки режиму посилання використовується для доступу до вмісту полотна:

```
ctx = canvas.getContext ( '2d');
```

Далі можна виводити на полотно елементи зображення (примітиви). У цілому послідовність дій зі створення зображення виглядає так:

```
...
<Canvas id = 'canva' width = 640 height = 480> Оновлення браузер </ canvas>
...
var canvas = document.getElementById ( 'canva'); // доступ до елементи
var ctx1 = canvas.getContext ( '2d'); // Встановити режим 2d
...
ctx.strokeRect (15, 15, 266, 266); // Вивести прямокутник
...
```

Основні методи для виведення примітивів наведені в табл. 4.1.

Таблиця 4.1

Методи створення основних примітивів

Ім'я методу	Призначення
1	2
strokeRect(x,y,w,h)	Прямокутник (координати кута, ширина, висота)
fillRect x,y,w,h ()	Зафарбований прямокутник (координати кута, ширина, висота)

1	2
clearRect(x,y,w,h)	Очищення області (координати кута, ширина, висота)
beginPath()	Початок окреслення фігури
closePath()	Кінець окреслення, замикає кінці
stroke()	Промальовування фігури (робить фігуру видимою)
fill()	Заливка фігури
moveTo(x, y)	Переміщення курсора (без лінії)
lineTo(x, y)	Лінія з поточної позиції в точку
arc(x, y, r, startAngle, endAngle, anticlockwise)	Дуга (координати центру, радіус, кут початку, кут кінця, напрямок)
quadraticCurveTo(…)	Крива Безьє другого порядку
bezierCurveTo(…)	Крива Безьє третього порядку

Для управління кольором елементів використовується дві властивості (визначають поточний колір): fillStyle (визначає колір заливки) і strokeStyle (визначає колір ліній). Значення кольорів задаються так, як і в css:

```
ctx.fillStyle = "orange";
ctx.fillStyle = "#FFA500";
ctx.fillStyle = "rgb(255,165,0)";
ctx.fillStyle = "rgba(255,165,0,1)"
```

Далі наведено фрагмент коду та створюване зображення (рис. 4.1).

```
…
<canvas id='example'>Обновите браузер</canvas>
<script>
   var example = document.getElementById("example");
   var ctx = example.getContext('2d');
   example.height = 480; //встановити висоту холста
   example.width = 640; //встановити ширину холста
   ctx.beginPath();// почати лінію
   ctx.arc(80, 100, 56, 3/4*Math.PI, 1/4*Math.PI, true); //дуга
   ctx.fill();                  //залиті дугу

   ctx.moveTo(40, 140); //перенос курсору
   ctx.lineTo(20, 40);      //лінія
   ctx.lineTo(60, 100);   //лінія
   ctx.lineTo(80, 20);     //лінія
   ctx.lineTo(100, 100); //лінія
   ctx.lineTo(140, 40);   //лінія
   ctx.lineTo(120, 140); //лінія
   ctx.stroke();            //показати фігуру
</script>
…
```

Рис. 4.1. **Фігура на полотні**

Додаткові можливості зі створення зображення на полотні надають методи translate, scale і rotate. Їх виклик в деякій точці викликає трансформацію (зміщення, масштабування та поворот) усіх наступних виведених елементів.

Метод translate(x,y) переміщує елементи, що виводяться на Canvas, в іншу точку координатної сітки.

Метод scale(x,y) змінює розміри виведених елементів. Аргумент x задає коефіцієнт масштабування по горизонталі, а аргумент y задає коефіцієнт масштабування по вертикалі.

Метод rotate(angle) повертає елементи на заданий кут.

Приклад 14. Приклад демонструє можливості динамічного створення зображень. За допомогою сценарію створюється діаграма на основі даних з таблиці, яка заповнюється користувачем. Для створення видимої динаміки промальовування кожного стовпчика діаграми виконується за таймером.

```
<!DOCTYPE html>
<html><head><title>Приклад 14</title></head>
<body onLoad="init();">
<table>
<tr><td><input type=text id=vl0><td><input type=text id=vl1>
    <td><input type=text id=vl2><td><input type=text id=vl3>
<tr><td><input type=text id=sg0><td><input type=text id=sg1>
    <td><input type=text id=sg2><td><input type=text id=sg3>
</table>
<button onClick="f1();">Збудувати</button><br>
<canvas id="canvas" width="400" height="500">No Canvas</canvas>
<script>
```

```
var i, ctx,n,cw,ch,d;                          // глобальні змінні
var vl=new Array();var sg=new Array(); // змінні для даних
function init()
{
  var canvas = document.getElementById('canvas');// доступ до полотна
  ctx=canvas.getContext('2d');                    // задання режиму
  cw=canvas.width;  ch=canvas.height;            //запам'ятовування розмірів
  ctx.font = 'italic 20px sans-serif';           //задання характеристик
  ctx.textBaseline = 'top';                      //шрифта
}
function f1()
{
  i=0;
  ctx.clearRect(0,0,400,500);                    //очищення полотна
  for (j=0;j<4;j++) { v="vl"+j; s="sg"+j;
              vl[j]=parseInt(document.getElementById(v).value);//зчитування
              sg[j]=document.getElementById(s).value;};          //даних
  n=vl.length;
  d=(cw-20)/n;                                   //обчислення ширини шпальти
  tim=setTimeout(draw, 500);                     //затримка виведення колонки
}
function draw()
{
 ctx.fillStyle = "#ffaacc";                      //промальовування
 ctx.fillRect(10+i*d,ch-40-vl[i]*2,d,vl[i]*2);   // стовпчика
 ctx.fillStyle = '#00f';                         // i
 ctx.fillText (sg[i], i*d, ch-40);               // підписи
 if (i<n-1) tim=setTimeout(draw, 500);    // затримка виведення наступної колонки
 i++;
}
</script>
</body>
</html>
```

Функція init (обробник події закінчення завантаження сторінки) виконує дії з підготовки полотна до висновку діаграми. Обробник події клацання на кнопці (f1) виконує зчитування даних з таблиці та запуск таймера для виклику функції draw (малювання чергового стовпчика). Більш детальний опис дій сценарію міститься в коментарях. Вигляд вікна з побудованою діаграмою наведено на рис. 4.2.

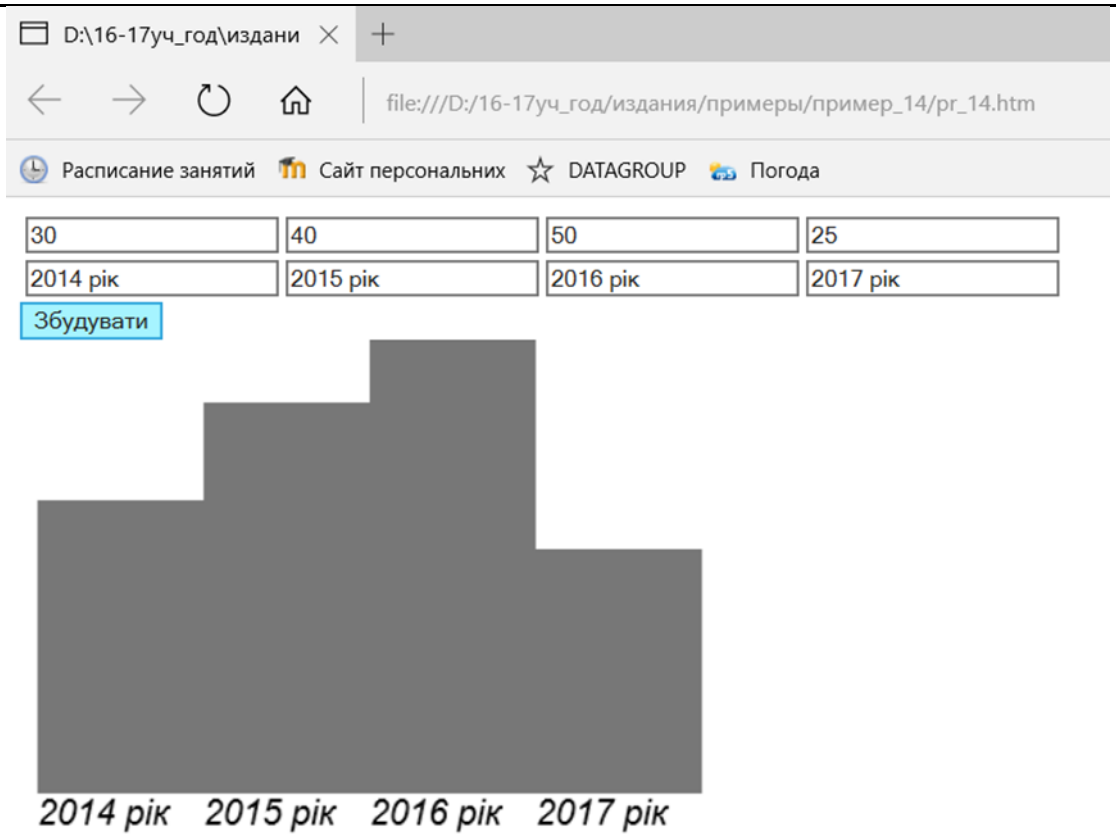

Рис. 4.2. **Вигляд сторінки з побудованою діаграмою**

4.1.2. Створення ефектів

Для підвищення ефективності розробки клієнтських скриптів створено достатньо багато бібліотек і фреймворків.

Бібліотеки у програмуванні є набором об'єктів і функцій, які можуть використовуватися багаторазово. Включення їх у код скрипта забезпечує вищий рівень абстракції під час розроблення, зменшується обсяг написаного коду, розширюється кросплатформеність і так далі.

Однією з найпопулярніших бібліотек, що розширюють можливості зі створення клієнтських скриптів, є jQuery.

Підключення здійснюється шляхом включення в текст сторінки тега script з посиланням (href = "url") на файл з бібліотекою, який може знаходитися як у папці сайта, так і на довільному ресурсі мережі. Після підключення для використання доступні базові можливості jQuery, серед яких: робота з селекторами; робота з атрибутами; обхід дерева DOM; маніпуляції елементами; робота з CSS-властивостями елементів; робота з подіями; візуальні ефекти; взаємодія з ajax; утиліти.

Додаткові можливості доступні після підключення відповідних плагінів (наприклад, gallery.js для створення галерей зображень і т.п.).

Загальний підхід до використання бібліотеки полягає в знаходженні елемента або колекції елементів, з якими передбачається виконувати деякі дії. Для цього існує велика кількість засобів (підтримуються всі селектори та синтаксис Xpath мови XML). Потім з елементом (або елементами колекції) зв'язується ланцюжок методів, що виконують необхідні дії.

Перед початком застосування скриптів, що містять звернення до jQuery, необхідно виключити ситуацію, коли звернення може статися до елемента, який ще не проінтерпретований браузером. Здійснюється це виконанням спеціального методу ready (f), який в якості аргументу отримує ім'я функції ініціалізації. Ця функція буде виконана тільки після остаточної побудови у браузері дерева DOM.

Звернення до бібліотеки для знаходження елементів здійснюється з використанням функції $() або її синоніму jQuery(). Аргументами виступають селектори або вирази Xpatch. Наприклад, my = jQuery (" <p> Текст! </ P> </ div>") – вибір елемента за контекстом або my = $ ("# unique") – вибір елемента за id.

Після вибору елемента для виконання необхідних дій до нього можна застосувати ланцюжок методів. Наприклад, наступний фрагмент коду забезпечує послідовну зміну елемента з класом c1: додається вміст і змінюються властивості (колір і видимість). Функція fn() буде виконана тільки після остаточної побудови DOM.

```
...
<style>
 div { color:blue; font-size:18px; visibility:hidden}
 </style>
 <script src="jquery.js"></script>
<script>
$(document).ready(fn);
function fn()
{
$("div.c1").html("<b>Це новина!</b> Була знайдена...").css("color",
"red").css("visibility","visible");
}
</script>
...
 <div></div>
 <div class="c1"></div>
 <div></div>
...
```

Для створення ефектів (зміна прозорості, видимості, показ елементів) бібліотека має спеціальні методи, а також зручний метод *animate(properties, [duration], [easing], [callback])*, що дозволяє управляти зміною властивостей елементів (тільки для властивостей, які задані числовими значеннями). Параметри мають такий сенс:

properties – перелік CSS-властивостей, що беруть участь в анімації й їх кінцевих значеннях;

duration – тривалість виконання анімації;

easing – зміна швидкості анімації до кінця;

callback – функція, яка буде викликана після завершення анімації.

Наприклад, наступний фрагмент забезпечить плавне збільшення та переміщення розділу, в якому є напис.

```
…
<style>
   div.c1 {background-color:#bca;width:120px;height:50px;
   border:1px solid green;}
   #i1 {color:red;font-size:5px; opacity: 0;margin-top:20px}
  </style>
  <script src="jquery.js"></script>
…
  <button id="go">Запуск анимации</button>
  <div class="c1"><p id="i1">ВНИМАНИЕ!</p></div>
  <script>
$("#go").click(fa);
function fa()
     { $("div.c1").animate({ width: "70%",
                             height:"150px",
                             marginTop:"15%" ,
                             marginLeft: "15%",
                             borderWidth: "10px"}, 2000,fa1);}
function fa1()
       {$("p").animate({fontSize:"100px",opacity:1},2000);}
  </script>
…
```

Усі функції animate виконуються одночасно. За необхідності їх послідовного виконання слід використовувати таймер або параметр call-back (останній у переліку параметрів animate).

Для успішного застосування бібліотеки jQuery необхідно ознайомитися з іншими методами, а також плагінами, використовуючи довідкову літературу або мережу Інтернет.

Наступний приклад демонструє використання плагіна для створення галереї зображень.

Приклад 15. Під галереєю розуміють динамічний елемент, що забезпечує перегляд серії зображень на сторінці. Для його створення використано плагін ppGallery.js, який підключається разом з jQuery. Плагін розроблений таким чином, що може вбудовуватися в різні контейнери на сторінці. Після підключення вимагає ініціалізації викликом функції ppGallery (). Зображення, які будуть використані, поміщаються в перелік ul з атрибутом id = "gallery" (цей перелік необхідно помістити в контейнер і позиціонувати відповідно до загального задуму).

```
<!DOCTYPE html>
<html>
<head>
<title>Галерея з використанням JQuery</title>
<link href="ppgallery.css" rel="stylesheet" type="text/css" />
<script src="jquery.min.js"></script>
<script src="ppgallery.js"></script>
<script>
$(document).ready(function() {
        $('#gallery').ppGallery(); });
</script>
</head>
<body>
<ul id="gallery">
  <li><a href="images/im_1.jpg"><img src="images/im_1.jpg"></a></li>
  <li><a href="images/im_2.jpg"><img src="images/im_2.jpg"></a></li>
  <li><a href="images/im_3.jpg"><img src="images/im_3.jpg"></a></li>
</ul>
</body>
</html>
```

У початковому стані (після завантаження) галерея стає переліком мініатюр (рис. 4.3).

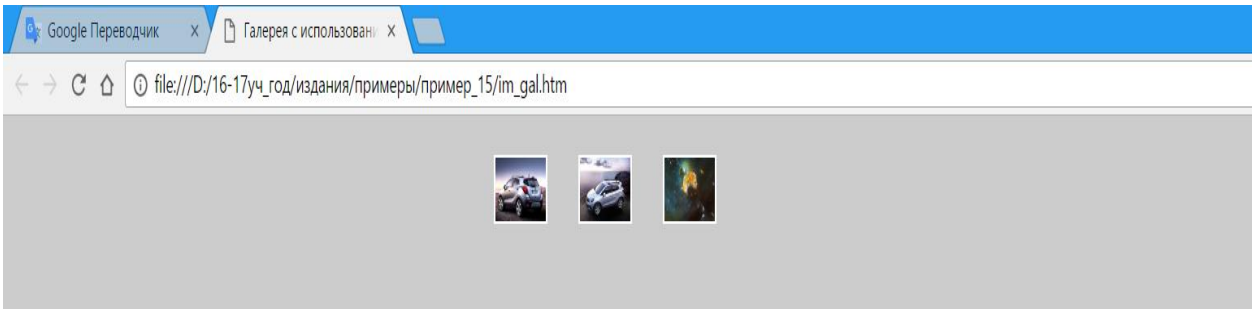

Рис. 4.3. **Вигляд сторінки у початковому стані**

Після клацання на одній з мініатюр затінюється основне вікно, поверх нього відкривається вікно перегляду зображень з елементами управління переглядом (рис. 4.4). Після закриття (кнопка close) вікно документа повертається в початковий стан.

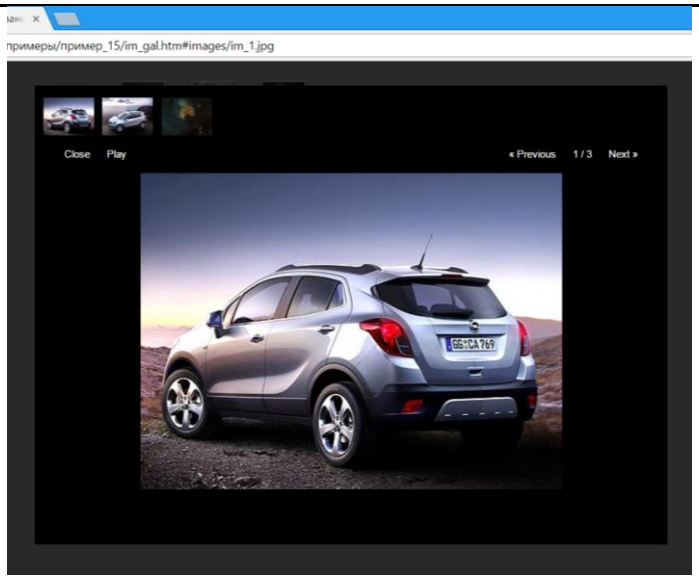

Рис. 4.4. **Вікно з галереєю**

Існує безліч інших плагінів, які розширюють бібліотеку. Бібліотека може бути розширена і за рахунок підключення функцій, які розроблені користувачем.

4.2. Публікація WEB-сайта

Для розміщення (публікації) створеного сайта в мережі Інтернет необхідно знайти місце для його розміщення, отримати для нього доменне ім'я та розмістити на одному з серверів мережі.

Після розміщення сайт піддається тестуванню. Потім настає період супроводження сайта, серед інших завдань виконується оцінювання ефективності публікації.

4.2.1. Розміщення сайта у мережі

Сайт необхідно розмістити на машині, що має постійне підключення до мережі, постійну адресу та запущений Web-сервер. Такою машиною може бути ваш власний комп'ютер. Однак рішення організаційних завдань, пов'язаних з цим, і адміністрування такого комп'ютера вимагають певних витрат і знань. У зв'язку з цим простіше скористатися послугами з розміщення сайтів, наданими в мережі Інтернет. Така послуга називається **хостінгом**. У цьому випадку надається місце на одному з серверів мережі. Надається ця послуга як безкоштовно, так і на комерційній основі.

Недоліки безкоштовного хостингу – це нав'язування реклами, непривабливе ім'я (домени третього і навіть четвертого рівня) та неперед-

бачуваний час життя. "Безкоштовність" розглядається в більшості випадків як крок до розміщення на комерційній основі. Тому загальною є ситуація, коли спочатку все функціонує нормально, але з часом починаються проблеми. Це, в кінцевому підсумку, складає відповідне ставлення до сайтів на безкоштовних хостінгах, хоча в ряді випадків таке рішення може виявитися цілком прийнятним.

Соліднішим виглядає розміщення сайта на платному хостінгу. Розвиток цієї послуги призвів до появи спеціалізованих організацій (дата-центрів), які надають цілий ряд різних типів хостингу. Це віртуальний хостинг – розподіл ресурсів сервера (простір пам'яті, пропускна здатність каналів) між безліччю сайтів. Віртуальний сервер – розподіл ресурсів між декількома сайтами. Виділений сервер – монопольне використання сервера ресурсом. Розміщення власної апаратури в дата-центрі для обслуговування та супроводження (колокейшн). Надання для використання орендованих ресурсів (ресселінг).

Вибір типу хостінгу визначається виходячи з можливостей і завдань проекту. Дані про різні хостінги нескладно отримати в Інтернеті (наприклад, за посиланням http://www.tophosting.in.ua/).

Наступний крок – це отримання доменного імені. Для цього необхідно зареєструвати домен в одному з існуючих. Найчастіше хостери пропонують послугу з реєстрації домену, однак для довгострокового проекту краще це зробити окремо. Оскільки за реєстрації домену в хостера домен належить йому, то в разі відмови від використання цього хоста ім'я буде втрачене.

Найбільш часто в якості доменів для реєстрації вибирають територіальні домени країни перебування .ua (Україна), .ru (Росія), .by (Білорусія), .md (Молдова) і т. д. Варто зауважити, що в кожної доменної зони є свої правила реєстрації. Наприклад, для отримання доменного імені в зоні .ua відповідно до Правил домену (від 1 квітня 2014 р.; http://www.hostmaster.net.ua/policy/) спочатку потрібно отримати реєстрацію відповідного позначення як знак для товарів і послуг. Свідоцтво на зареєстроване позначення видається відповідно до Закону України "Про охорону прав на знаки для товарів і послуг". У цьому випадку доменне ім'я має збігатися повністю або з компонентом зареєстрованого знака для товарів і послуг, а в якості власника такого свідоцтва й особи, яка претендує на доменне ім'я, має бути та сама особа. Позначення доменного імені, відповідно до Правил домену, має бути виконане обов'язково буквами латинського алфавіту. З цієї причини більшість вітчизняних сайтів розташовані в зоні .com.ua, в якій реєстрація вільна.

Після вирішення питання про вибір хоста та доменного імені (орендувати у хостера або зареєструвати самостійно) можна переходити до розміщення сайта.

Перш за все необхідно зареєструватися на обраному хості. Для безкоштовного хоста реєстрація полягає в заповненні запропонованої форми й отриманні на адресу електронної пошти реквізитів для доступу (зазвичай – посилання на майбутній ресурс і пароль). У разі зареєстрованого самостійно доменного імені необхідно налаштувати DNS-сервер, як правило, через спеціальну адміністративну панель.

Для розміщення своїх сторінок на сайті платного хостера недостатньо заповнення форми, зазвичай необхідно укласти контракт (найчастіше електронний) і оплатити послугу. У зв'язку з цим доцільно вибрати хостера, який має офіс, який можна відвідати (знаходиться в доступному місці). З виникненням проблем з ним буде легше контактувати, особисто приїхавши для з'ясування обставин, якщо виникне така потреба.

Останній етап – це пересилання файлів сайта на сервер. Ця процедура може виконуватись різними способами.

Часто це можна зробити через WEB-сторінку власника хоста. Зазвичай для цього слід авторизуватися (ввійти) в центр управління своїми сторінками на сайті хостера, ввівши логін і пароль у формі. Надалі буде наданий інтуїтивно зрозумілий інтерфейс для виконання всіх необхідних дій. Як правило, є і файли з відповідними довідками.

Пересилання файлів забезпечується програмами для розроблення Web-сторінок, наприклад, Dreamweaver. Для виконання необхідних дій в середовищі цих програм слід вивчити відповідні розділи довідкової системи.

Пересилання файлів може бути виконане і за протоколом FTP, якщо хостер підтримує такий доступ. Програм FTP-клієнтів досить багато, в тому числі й відомі файлові менеджери FAR і Total Commander. Для FAR досить набрати в командному рядку ftp: // ваш_логін: ваш_пароль @ ім'я_сайта та натиснути Enter. Far з'єднається з сайтом, перелік файлів якого відобразиться на одній з панелей. А подальші дії виконуються як для копіюванні файлів з одного накопичувача на інший.

4.2.2. Тестування та супроводження сайта

Тестування – це процес, який полягає в перевірці відповідності сайта заявленим характеристикам і вимогам (вимогам до експлуатації

в різних середовищах, з різними навантаженнями; вимогам з безпеки, ергономіки та до зручності використання).

Усі перевірки можна розподілити на три великі групи: технічні, ергономічні та відповідність цільовим призначенням.

Технічні перевірки полягають в отриманні оцінок показників функціональності та порівнянні їх значень із зафіксованими в технічному завданні (частина з них також характеризує якість сайта). Як правило, вони легко вимірювані й однозначні, наприклад:

перегляд сайта на моніторах з різною роздільною здатністю;

перевірка часу завантаження всіх сторінок сайта із заданою швидкістю з'єднання з Інтернет;

перевірка можливості перегляду сайта та правильності відображення кольорів за різної їх кількості, встановленої на моніторі;

перевірка сайта у ході перегляду його на різних браузерах і їх версіях;

перевірка гіперпосилань, пошук і усунення непрацюючих;

перевірка правильності відображення шрифтів на різних браузерах і їх версіях;

перевірка завантаження всіх графічних матеріалів сайта (малюнки, фотографії і т. д.);

перевірка заміщувальних написів графічних матеріалів;

перевірка працездатності функціональних елементів, установлених на сторінках сайта (лічильників і т. п.);

перевірка працездатності сайта на сервері з великою кількістю запитів тощо.

Ергономічні вимоги пов'язані зі зручністю використання (usability) сайта. Як правило, їх безпосереднє вимірювання утруднено, для їх отримання використовують експертні оцінки або опосередковані показники.

Зручність використання (usability) досліджується на прикладі випробувальних або експертних груп: як користувач сприймає продукт, як він уявляє собі шляхи його використання, скільки часу витрачає на ту чи іншу дію, які проблеми у нього виникають і чи в стані він їх вирішити. З цією метою передбачається висвітлення таких питань:

аналіз головної сторінки сайта, її ефективності для залучення відвідувачів до найбільш важливих розділів, для забезпечення повторних відвідувачів сайта і т. п.;

оцінювання логічності структури сайта для користувачів;

виявлення помилок у рубрикації, перевірка посилань на зрозумілість, оптимізація навігації сайта;

час завантаження й обсяг сторінок, можливості для їх "полегшення" без втрати якості;

оцінювання обраного дизайнерського рішення, його відповідності цілям і завданням сайта;

оцінювання якості верстання текстів: використання шрифтів, заголовки, виділення, посилання, ілюстрації, розподіл на абзаци, стійкість верстання;

виявлення невдалих форм подання інформації.

Якість сайта (ступінь його відповідності цільовим призначенням) залежить від багатьох складових. Оцінювання якості, тобто відповідності цільовим призначенням сайта, може проводитися за різними напрямами, серед яких можна виділити такі:

безпека, вразливість сайта;

відвідуваність (середня кількість Інтернет-відвідувачів сайта в день; середня кількість сторінок сайта, відвідуваних WEB-відвідувачами; середній час, проведений відвідувачем на сайті);

вартість залучення одного Інтернет-відвідувача;

окупність і рентабельність сайта;

якість інформації, розміщеної на сайті.

Як формулювання показників, так і отримання їх оцінок для сайта є найбільш складним. Багато в чому це творчий, евристичний процес, у якому можна використовуватися пов'язані, опосередковані показники або порівняння з існуючим рівнем, досягнутим кращими сайтами на дану тему.

Висновки й узагальнення

Якщо проаналізувати напрями розвитку WEB-ресурсів, то загальною тенденцією є збільшення обсягу програмних елементів, які перетворюють інформаційні сайти на WEB-додатки. Відповідно до цього змінюються і вимоги до розробника: знання програмування та відповідних технологічних засобів стає обов'язковим.

Jscript у поєднанні з можливостями зміни елементів через DOM – гнучкий і потужний засіб створення динамічних сторінок, широко використовуваний у розробці WEB-ресурсів. Основних напрямів застосування клієнтських сценаріїв декілька. Однак головним є підтримка призначеного для користувача інтерфейсу з метою надання йому більшої зручності й інтерактивності. Саме в цьому напрямі розвиваються технологічні засоби.

Загальною тенденцією в підвищенні ефективності розробок можна вважати використання шаблонів, бібліотек і фреймворків. Найбільш популярною бібліотекою є jQuery.

Теоретичні запитання

1. У чому полягають особливості обробки подій таймера?
2. Назвіть засоби, які забезпечують створення динамічної графіки на сторінках.
3. Сформулюйте послідовність дій в сценарії у ході створення зображень на "полотні".
4. Який ефект забезпечує застосування методу scale (x, y)?
5. Що дає розробнику використання бібліотеки jQuery?
6. Сформулюйте дії, які необхідно виконати для використання jQuery.
7. Назвіть групи функцій, реалізованих у бібліотеці jQuery.
8. Опишіть механізми селекції елементів за допомогою бібліотеки jQuery, поясніть прикладами.
9. Яку обробку елементів і даних забезпечують функції бібліотеки jQuery?
10. Опишіть ефекти, які можна створювати засобами бібліотеки jQuery.
11. Назвіть відомі вам плагіни для розширення можливостей JQuery.
12. Назвіть дії, які необхідно виконати для публікації сайта в мережі Інтернет.
13. Дайте порівняльну характеристику різних варіантів хостингу.
14. У чому полягає тестування сайта?

Контрольні завдання

1. Самостійно завантажте плагін меню та створіть сторінку з ним.
2. Розробіть сторінку, розмістивши галерею jQuery в спеціально відведеній області сторінки.
3. Розробіть сторінку для динамічної побудови графіка функції sin(x) із заданим діапазоном зміни x.
4. Створіть на сторінці зображення з рухомою стрілкою компаса.
5. Розмістіть одну зі створених сторінок на безкоштовному хостингу, задокументувавши скриншотами основні дії.

Рекомендована література

1. Гультяев А. К. Macromedia Home Site 5. Инструмент подготовки веб-публикаций : практ. пособ. / А. К. Гультяев. – Санкт-Петербург : Учитель и ученик; КОРОНА принт, 2002. – 1504 с.
2. Дронов В. А. Самоучитель Macromedia Dreamweaver 8 / В. А. Дронов. – Санкт-Петербург : БХВ-Петербург, 2006. – 320 с.
3. Кирсанов Д. Веб-дизайн : книга Дмитрия Кирсанова / Д. Кирсанов – Санкт-Петербург : Символ-Плюс, 1999. – 376 с.
4. Лоусон Б. Изучаем HTML5. Библиотека специалиста / Б. Лоусон, Р. Шарп. – Санкт-Петербург : Питер, 2011. – 272 с.
5. Методичні рекомендації до самостійної роботи з навчальної дисципліни "Основи проектування WEB-видань" для студентів напряму підготовки 6.051501 "Видавничо-поліграфічна справа" всіх форм навчання / укл. В. П. Молчанов – Харків : ХНЕУ ім. С. Кузнеця, 2016. – 56 с.
6. Методичні рекомендації по виконанню лабораторних робіт з навчальної дисципліни "Основи проектування WEB-видань" для студентів спеціалізації "Комп'ютеризовані технології та системи видавничо-поліграфічних виробництв" усіх форм навчання / укл. В. П. Молчанов, Т. Ю. Андрющенко. – Харків : Вид. ХНЕУ, 2009. – 84 с.
7. Молчанов В. П. Основи проектування WEB-видань : конспект лекцій / В. П. Молчанов. – Харків : Вид. ХНЕУ, 2008. – 168 с.
8. Николаенко Д. В. Практические занятия по JAVASCRIPT для начинающих / Д. В. Николаенко. – Москва : Наука и техника, 2000. – 430 с.
9. Нильсен Я. Дизайн WEB-страниц. Анализ удобства и простоты использования 50 узлов / Я. Нильсен, М. Тахир ; пер. с англ. – Москва : ИД "Вильямс", 2002. – 336 с.
10. Робоча програма навчальної дисципліни "Основи проектування WEB-видань" для студентів напряму підготовки 6.051501 "Видавничо-поліграфічна справа" всіх форм навчання / уклад. В. П. Молчанов. – Харків : ХНЕУ ім. С. Кузнеця, 2016. – 46 с.
11. Сакс Т. Дизайн и архитектура современного вебсайта. Опыт профессионалов / Т. Сакс, Г. Мак-Клейн ; пер. с англ. – Москва : ИД "Вильямс", 2002. – 304 с.
12. Хестер Н. Создание Web-сайтов в Microsoft Expression Web / Н. Хестер. – Москва : ДМК Пресс, 2007. – 252 с.

Інформаційні ресурси

13. Справочник по HTML [Электронный ресурс]. – Режим доступа : http://htmlbook.ru/html.

14. Справочник CSS [Электронный ресурс]. – Режим доступа : http://htmlbook.ru/css.

15. Справочник JS [Электронный ресурс]. – Режим доступа : http://javascript.ru/manual.

16. Справочник HTML5 Canvas [Электронный ресурс]. – Режим доступа : http://xiper.net/manuals/canvas.

17. jQuery [Электронный ресурс]. – Режим доступа : http://jquery.page2page.ru/index.php5.

Глосарій

Адаптивна верстання – верстання, параметри якого (вид сторінки) автоматично змінюються залежно від умов перегляду для кращого сприйняття.

Бібліотека (jQuery) – в програмуванні, збірник підпрограм або об'єктів, які використовуються для розроблення програмного забезпечення.

Бриф (англ. brief) – стисла письмова форма узгодження основних параметрів проекту.

WEB-сайт – кілька WEB–сторінок, пов'язаних інформаційно, єдиним оформленням, розміщенням і URL.

WEB-сторінка – гіпертекстовий документ, розміщений в мережі Інтернет, доступ до якого здійснюється по URL.

WEB-технологія – сукупність засобів і методів передавання та використання даних у комп'ютерній мережі Інтернет.

Верстання сторінок сайта – процес розміщення елементів на сторінках засобами HTML і CSS відповідно до деякого макету.

Динамічна графіка – створення зображень на екрані комп'ютера в ході виконання програм.

DOM (Document Object Model) – уявлення вмісту документа у вигляді дерева об'єктів, що відповідають елементам сторінки.

Клієнтський сценарій – програма, яка включена в текст сторінки у вигляді вихідного тексту та виконується браузером.

Медіазапити – умовні конструкції, що використовують значення характеристик пристрою користувача для обрання варіанту правил CSS.

Мова розмітки (англ. markup language) – набір символьних міток (тегів), що вставляються в текст для передавання інформації про його виведення або будову.

Обробник подій – функція, виклик якої здійснюється з настанням певної події в системі.

Підстановки (англ. entities) – спеціальні розмічувальні позначення для вставки в текст символів, яких немає на клавіатурі або які мають спеціальне значення.

Регулярні вирази – формальна мова пошуку та здійснення маніпуляцій з підрядками в тексті, заснована на використанні метасимволів.

Селектор CSS – формальна конструкція, яка зв'язує набір значень властивостей з конкретним елементом сторінки.

Семантична розмітка – підхід до написання коду мовою HTML, заснований на використанні тегів відповідно до їх семантики (призначенням); передбачає логічну та послідовну ієрархію елементів сторінки.

Специфікація CSS (Cascading Style Sheets) – формальні засоби опису зовнішнього вигляду документа, розробленого з використанням мови розмітки.

Тег (англ. tag) – одиниця розмітки, що розділяє вихідний текст на структурні елементи.

Фреймворк (англ. Framework) – програмне забезпечення, що полегшує розроблення й об'єднання різних компонентів великого програмного проекту.

Хостінг (англ. hosting) – послуга з надання ресурсів для розміщення даних на сервері, підключеному до мережі Інтернет.

Юзабіліті (англ. Usability) – ергономічна характеристика ступеня зручності роботи користувача з WEB–сайтом.

Предметний покажчик

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

Молчанов Віктор Петрович

ОСНОВИ ПРОЕКТУВАННЯ WEB-ВИДАНЬ

Навчальний посібник

Самостійне електронне текстове мережеве видання

Відповідальний за видання *О. І. Пушкар*

Відповідальний редактор *М. М. Оленич*

Редактор *Н. І. Ганцевич*

Коректор *Т. А. Маркова*

План 2017 р. Поз. № 18 ЕНП. Обсяг 159 с.

Видавець і виготовлювач – ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Науки, 9-А

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру ***ДК № 4853 від 20.02.2015 р.***