МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНО-ОБЧИСЛЮВАЛЬНИХ СИСТЕМ ТА УПРАВЛІННЯ

КОНСПЕКТ ЛЕКЦІЙ
З ДИСЦИПЛІНИ
"Веб-технології та веб-дизайн"
для студентів напряму підготовки
6.050101 "Комп'ютерні науки"

Тернопіль – 2012

Конспект лекцій з дисципліни "Веб-технології та веб-дизайн" для студентів напряму підготовки 6.050101 "Комп'ютерні науки" / Биковий П.Є., Палій І.О., Комар М.П. - Тернопіль: ТНЕУ, 2012.- 92 с.

Укладачі: П.Є. Биковий, к.т.н., доцент
І.О. Палій, к.т.н., доцент
М.П. Комар, викладач

Рецензент: Ю.Р.Піговський, к.т.н., доцент

Відповідальний за випуск: Саченко А.О., д.т.н., професор, завідувач кафедри інформаційно-обчислювальних систем та управління

Конспект лекцій розглянуто та рекомендовано до друку на засіданні кафедри інформаційно-обчислювальних систем та управління, протокол № 11 від 15 травня 2012 р.

ЗМІСТ

1. ОСНОВИ ВЕБ

Історія створення

У 1969 році Міністерство оборони США започаткувало розробку проекту, котрий мав на меті створення надійної системи передачі інформації на випадок війни. Метою цього проекту було створення комп'ютерної мережі з високою надійністю передачі інформації. Вихід із ладу вузла комутації або каналу передачі даних не повинен був спричиняти втрату працездатності мережі. Агентство DARPA (Defrense Advanced Recearch Projects Agency) запропонувало розробити для цього комп'ютерну мережу (ARPANET). Розроблена мережа об'єднувала декілька навчальних закладів та дослідницьких центрів.

Перший сервер ARPANET було встановлено 1 вересня 1969 року у Каліфорнійському університеті в Лос-Анжелесі. Комп'ютер «Honeywell 516» мав 12 кілобайт оперативної пам'яті.

1 січня 1983 року мережа ARPANET перейшла з протоколу NCP на протокол TCP/IP, який досі успішно використовується для об'єднання мереж. Саме у 1983 році за мережею ARPANET закріпився термін «Інтернет».

У 1984 році була розроблена система доменних назв (англ. Domain Name System, DNS). Тоді ж у мережі ARPANET з'явився серйозний суперник — Національний науковий фонд США (NSF) заснував міжуніверситетську мережу NSFNet (англ. National Science Foundation Network). До цієї мережі за рік під'єдналось близько 10 тисяч комп'ютерів; звання «Інтернет» почало плавно переходити доNSFNet.

У 1988 році було винайдено протокол Internet Relay Chat (IRC), завдяки якому в Інтернеті стало можливим спілкування в реальному часі (чат).

У 1990 році мережа ARPANET припинила своє існування, програвши конкуренцію NSFNet. Тоді ж було зафіксовано перше підключення до Інтернету телефонною лінією.

1993 році з'явився служба WWW (word wide web) – всесвітня павутина.

У 1998 році Папа Римський Іван-Павло II заснував всесвітній день інтернету (30 вересня)

У 2004 року з'явився термін Web-2.0, а у 2007 році Web-3.0

Протоколи зв'язку

Обмін інформацією між серверами та клієнтами здійснюється за певними правилами, які називають *протоколами.* Всі дані, що циркулюють у глобальному інформаційному полі, розбито на невеликі блоки і вкладено в пакети. Кожний пакет окрім даних має заголовок, де зберігаються адреса відправника, адреса одержувача та інша інформація, необхідна для збирання пакетів у пункті призначення. Теоретично можливо, що різні пакети одного повідомлення пройдуть різними шляхами, але все одно досягнуть адресата і

будуть зібрані в повне повідомлення.

Поділ даних на пакети та їх збирання у пункті призначення здійснюється під керуванням протоколу TCP (Transmission Control Protocol — протокол керування передаванням), а власне передавання пакетів мережею та досягнення ними адресата забезпечує протокол IP (Internet Protocol — міжмережний протокол).

У Інтернеті використовується велика кількість протоколів, завдяки чому існує широкий спектр служб, які надаються та підтримуються за допомогою цієї глобальної мережі.

Поділ даних на пакети та їх збирання у пункті призначення здійснюється під керуванням протоколу TCP (Transmission Control Protocol – протокол керування передачею), а власне передавання пакетів мережею та досягнення ними адресата забезпечує протокол IP(Internrt Protocol –міжмережевий протокол).

У Інтернеті використовується велика кількість протоколів, завдяки чому існує широкий спектр служб, які надаються та підтримуються за допомогою цієї глобальної мережі.

Однією зі служб є Всесвітня павутина (World Wide Web – WWW), або просто Веб. Ця розповсюджена по всьому світу інформаційна мультимедійна система, яка об’єднує в єдиному просторі інформацію різних типів. Робота у веб подібніша до віртуальної подорожі світом з індивідуванням цікавих місць. Ця служба базується на протоколі HTTP (Hyper Text Transfer Protocol - протокол передавання гіпертексту).

Інші служби базуються та називаються на основі протоколів:

- **FTP** (File Transfer Protocol – протокол передавання файлів). Сервери, що підтримують цей протокол, називаються FTP-серверами. Частина дискового простору таких серверів доступна через Інтернет. Крім того, до служб Інтернету належать електронна пошта, служби миттєвого передавання повідомлень, служба новин User та інші.
- **SMTP** (Simple Mail Transfer Protocol) - протокол передачі пошти.
- **POP3** (Post Office Protocol) – пошта отримання протокол для доставлення повідомлень етектроної пошти, її написання.
- **Telnet** (Telnational Network) – протокол для реалізації тексту інтерфейсу по мережі.

Адресація в Інтернеті

Для того щоб комп’ютер мав змогу передавати та приймати дані з використанням протоколу IP, він повинен мати унікальну адресу, яку називають IP адресою. На даний час використовують дві версії IP: IPv4 та IPv6.

IP версії 4 має наступний формат: xxx.xxx.xxx.xxx, де xxx — число від 0 до 255 (наприклад, 193.205.31.47). Такий формат називається точковим записом. Така адреса є 32-бітною і повинна бути унікальною в рамках однієї мережі. Таким чином, максимальна кількість комп'ютерів в рамках однієї ізольованої

мережі може бути 2^32 = 4.294.967.296 комп'ютери.

IP версії 6 (IPv6) є новою версією протоколу, який є наступником IPv4. Новий інтернет-протокол був розроблений в 1990-х роках і, на відміну від попередньої версії, він використовує 128-розрядну систему адресації. Таким чином, адресний простір у новій версії протоколу міститиме 2128 або ж понад 3,4 E+38 адрес, що забезпечує достатнє розширення Інтернету.

IP-адреса є зручною для комп'ютерів, але людям запам'ятати її важко. Тому серверам присвоюють так звані доменні імена — набори розділених крапками послідовностей символів, наприклад tneu.edu.ua

Останній, найзагальніший елемент доменного імені — в нашому прикладі ua — називають доменом першого рівня, edu — доменом другого рівня і т. д. Кількість доменів у адресі не регламентовано. У багатьох країнах домен першого рівня є кодом країни: ua — Україна, ru — Росія, fr — Франція і т. д.
Домени першого рівня можуть також позначати сферу діяльності: com — комерційні компанії, gov — урядові організації, edu — навчальні заклади, org — некомерційні організації, mil — військові організації.

Проте, для відкритя будьякого документа, що зберігається на комп'ютері, необхідно вказати ім'я файлу та повний шлях до нього. Так само і для доступу до інтернет ресурсу недостатньо знати лише IP адресу або доменне ім'я комп'ютера, на якому цей ресурс розміщено, — ви маєте вказати також папку та ім'я файлу. Крім того в Інтернеті застосовуються різні протоколи, а отже, слід вказати ще й протокол. Адресу, що містить усі зазначені елементи, називають URL (Uniform Resource Locator — єдиний вказівник на ресурс) або адресою ресурсу.

Типовий URL має такий вигляд: протокол://адреса, де протокол визначає метод доступу до ресурсу, наприклад http, ftp; адреса описує місце розташування ресурсу і включає назву сервера, шлях до документа і його ім'я.
Наведемо приклади URL-адрес:

- http://www.syhiv.com/pub/files/school.html — адреса файлу school.html, розміщеного в каталозі pub/files на сервері www.syhiv.com. Доступ до сервера здійснюється за протоколом HTTP;
- http://www.syhiv.com — адреса головної сторінки сервера www.syhiv.com;
- ftp://ftp.syhiv.com/pub/files/school.txt — адреса файлу school.txt, розміщеного в каталозі pub/files на сервері ftp.syhiv.com. Доступ до сервера здійснюється за протоколом FTP.

2. КЛАСИФІКАЦІЇ ВЕБ-САЙТІВ І ГІПЕРТЕКСТОВИХ ДОКУМЕНТІВ

Веб-сторінка та веб-сайт

Як зазначалося, найвідомішою та найпопулярнішою службою Інтернету є Всесвітня павутина (Веб). Саме після її розповсюдження став можливий масовий доступ користувачів до Всесвітньої мережі. Своєю появою Веб має завдячити Тіму Бернесу-Лі, який винайшов протокол HTTP, адреси URL та мову розмітки HTML — технології, на яких ґрунтується Веб.

Служба Веб підтримується сукупністю серверів, які здатні обмінюватися даними за протоколом HTTP. Цих серверів мільйони, й розповсюджені вони по всьому світу. На них містяться *веб-сторінки* — спеціальні документи, створені з використанням мови розмітки HTML. Кожна веб-сторінка має адресу URL, за допомогою якої вона може бути знайдена.

Перегляд веб-сторінок здійснюється у спеціальних програмах — браузерах, найпоширенішими з яких є Internet Explorer, Mozilla, Google Chrome, Opera та ін.

Основною особливістю та перевагою веб-сторінок є те, що інформація на них організована як *гіпертекст*. Це текст, в який вбудовано спеціальні коди (теги), що керують такими додатковими елементами, як форматування, ілюстрації, мультимедійні вставки та гіпертек-стові посилання.

Гіпертекстоее посилання (гіперпосилання, гіперзв'язок чи гі-перлінк) — це об'єкт веб-сторінки, що містить інформацію про адресу іншої веб-сторінки або про певне місце на поточній сторінці. Таким об'єктом може бути фрагмент тексту (зазвичай виділений кольором та підкресленням) або ілюстрація. У разі наведення на гіперпосилання вказівник миші набуває форми руки з витягнутим вказівним пальцем. Клацнувши лівою кнопкою миші, можна виконати перехід за гіперпосиланням. При цьому браузер завантажує веб-сторінку, яка міститься за адресою, зазначеною в посиланні. Ця веб-сторінка також може містити гіперпосилання, які вказують на інші веб-сторінки. Оскільки веб-сторінки можуть бути зв'язані між собою довільно, такий спосіб їх організації отримав назву Всесвітня павутина.

Процес переходу в інші місця поточної веб-сторінки або до інших веб-сторінок за допомогою гіперпосилань називають *навігацією.* Якщо після низки переходів за гіперпосиланнями необхідно повернутися на попередню сторінку, то користуються кнопкою «**Назад»** панелі інструментів браузера. Поруч із нею є стрілка для розкривання списку сторінок, що вже були переглянуті в цьому сеансі роботи; у списку можна вибрати потрібну сторінку і перейти до неї.

Сукупність веб-сторінок, що тематично пов'язані між собою й розроблені як єдине ціле, називають *веб-сайтом* або просто *сайтом.* Сторінки веб-сайту розміщуються на одному сервері та мають однакову адресу сайту, наприклад http://bhv.kiev.ua/.

Веб-сайт може надавати як пасивну інформацію, що читається лише

відвідувачем, так і активну, яку відвідувач може додавати або редагувати. Для організації інтерактивної взаємодії відвідувачів використовують гостьові книги, форуми, чати та блоги, які описані в цьому розділі далі.

Домашні сторінки

У 90-х роках минулого століття виник новий вид творчості — створення *домашніх сторінок.* Спочатку цей термін (від англ. home page) означав дім людини в Інтернеті, місце, де вона **зберігає** нформацію про себе. Особисті сторінки мають тисячі людей. Одні з них містять лише коротку інформацію про власника, а інші - корисну інформацію з певної тематики, графічний матеріал, фотографії тощо.

Термін «домашня сторінка» не має чіткого визначення, так називають також головну сторінку сайту, що відкривається у разі введення його доменного імені, та сторінку, яка завантажується під час кожного запуску браузера.

Структура веб-сайтів

Зовнішній вигляд кожного сайту є унікальним, проте в усіх сайтів можна знайти спільні за функціональністю частини. На будь-якому сайті першою відкривається *головна сторінка.* її розробці приділяють особливу увагу, оскільки дослідження показали, що люди не здатні читати інформацію, що відображається на моніторі, так уважно, як книжки або журнали, вони зазвичай лише поверхово переглядають її, наприклад, як рекламу. Якщо головна сторінка містить те, що шукає відвідувач, він читає її далі, а якщо ні — переходить до інших сайтів, яких в Інтернеті дуже багато.

У верхній частині головної сторінки зазвичай розташована так звана «шапка», яку дублюють на інших сторінках сайту. Це роблять спеціально, адже ця частина відображається у вікні браузера першою, і відвідувач насамперед звертає увагу на неї.

Щоб забезпечити швидкий перехід до основних тематичних розділів сайту, створюють *меню сайту* — список гіперпосилань на його розділи. Горизонтальне меню зазвичай розташовують у шапці, іноді дублюючи його в нижній частині сторінки, а вертикальне — переважно в лівій частині сторінки, у місці, звідки відвідувач починає її переглядати. Меню є одним із найважливіших компонентів сайту, користувач постійно звертає на нього увагу, і тому вимоги до нього високі. Меню має бути зручним, помітним і зрозумілим, інакше користувач не знатиме, як потрапити в потрібний розділ, і вийде із сайту. Пункти меню мають бути чітко відділені один від одного.

На рис. 2.1 показано головну сторінку сайту міжнародної конференції IDAACS. Меню сайту на ній розміщене ліворуч.

Рис. 2.1. Головна сторінка сайту міжнародної конференції IDAACS

Основи Інтернету

Гіперпосилання, розміщені в тексті чи у вигляді графічних об'єктів, дозволяють переходити на різні сторінки сайту або навіть на інші сайти. На сайтах із дуже великим обсягом інформації є сторінки третього рівня, а якщо необхідно — то й четвертого, п'ятого і т. д.

Загалом же виділяють три структури веб-сайтів — *лінійну, деревоподібну* та *довільну.* Подорожуючи сайтом із лінійною структурою, з головної сторінки ви перейдете на другу сторінку, з неї — на третю і т. д. На сайті з деревоподібною структурою з головної сторінки можна потрапити на одну зі сторінок другого рівня, звідти — на одну зі сторінок третього рівня і т. д. Сайт із довільною структурою видається зовсім неорганізованим, але саме у цьому й полягає принцип його створення. Подорожуючи таким сайтом, ви можете переходити з однієї його сторінки на інші у різні способи, і ваш шлях назад не обов'язково має бути таким самим.

Створення та підтримка веб-ресурсів

Для того щоб сайт став доступним широкому колу відвідувачів, йому необхідно призначити доменне ім'я і розмістити в мережі Інтернет. Розміщення сайту на сервері та подальше його адміністрування називають *хостингом.* Наданням такої послуги займаються спеціальні організації. Хостинг буває платний і безкоштовний. На серверах, що забезпечують хостинг, крім власне розміщення сайту зазвичай надається можливість створювати веб-сторінки та організовувати форуми і чати в автоматизованому режимі. Однією з найпростіших форм автоматизованого створення веб-ресурсів є *блоги* (онлайнові щоденники), які дають змогу публікувати та впорядковувати (зазвичай у хронологічному порядку) на веб-сторінках різноманітні записи. Зараз ви ознайомитеся з цими засобами створення й підтримки веб-ресурсів.

Форуми та чати на веб-сайтах

Є можливість не лише автоматично створювати та оформляти веб-сторінки, а й створювати та адмініструвати форуми і чати. У стародавній Греції слово «форум» означало ринок або базар — місце, де люди могли спілкуватися. *Електронний форум* також надає можливість відвідувачам сайту спілкуватися між собою. Вони мають зареєструватися у форумі, заповнивши спеціальну форму, після чого можуть вводити текст повідомлень у призначених для цього полях. Для спілкування учасників не потрібно, щоб вони одночасно були присутніми на сайті. Висловивши свою думку, користувач може прочитати відповідь інших учасників форуму упродовж певного часу в іншому сеансі спілкування, оскільки його інформація залишається на сайті й може довго бути доступною для перегляду.

Чат також дозволяє кільком відвідувачам сайту поспілкуватися між собою. Його відмінність від форуму полягає в тому, що спілкування відбувається у *режимі реального часу* (коли обмін інформацією здійснюється майже миттєво, без відчутних затримок) і лише між тими користувачами, які одночасно звернулися до сайту з чатом і зареєструвалися для участі в електронній розмові. У наступному сеансі інформація про попередній відсутня.

У форумах, і у чатах відвідувачі спілкуються з використанням *логінів,* або *ніків* — імен, спеціально вибраних для цього. Логіни не завжди відповідають реальним іменам людей, і в цьому полягає певна інтрига електронного спілкування.

Ведення блогів

Блог (англ. blog — weB LOG) — це мережний журнал, що містить записи у зворотному хронологічному порядку та забезпечує можливість додавання читачами коментарів. У блог можна записати свої враження про новий фільм, гру, сайт або програму, розповісти історію.

Блог може бути особистим, груповим або колективним. *Особистий блог* називають онлайновим щоденником, у ньому автор може публікувати свої думки, приватні фотографії, наводити посилання на цікаві інтернет-ресурси для друзів або для широкої інтернет-аудиторії. *Груповий блог* присвячують певній тематиці або спільній сфері інтересів (клубні, корпоративні блоги). *Колективний блог* відкритий для спільного редагування. Блог може бути як самостійним ресурсом, так і частиною якогось веб-проекту.

Технічно мережні журнали реалізують за допомогою різних веб-технологій. Є спеціалізоване програмне забезпечення (наприклад, WordPress, Pivot), що дає змогу людині, яка не володіє спеціальними знаннями з розробки веб-ресурсів, спростити створення, обслуговування і ведення блога.

З кінця 90-х років минулого століття все популярнішими стають безкоштовні служби, що надають можливість вести особистий блог за умови реєстрації на сайті, який надає такі послуги. Найвідомішими серед них є

www.livejournal.com, www.blogger.com, www.jj.ru, www.rax.ru, а також українська служба blog.net.ua. Ресурс www.livejournal.com надає кожному користувачу можливість вести власний журнал, який інтегрує блоги інших користувачів, організовує спілкування між ними.

Читачі можуть об'єднуватися в спільноти. *Спільнота* — це журнал, до якого пишуть різні користувачі, й усіх їх вважають його власниками. Кожен, хто цікавиться якоюсь конкретною темою, може знайти або створити спільноту, присвячену цій темі. Наприклад, мешканці одного міста можуть створити спільний журнал для обміну інформацією про це місто чи для оголошень про місцеві події.

Інформація блога орієнтована на широку аудиторію. Ще однією його особливістю є те, що вміст журналу унікальний, ніде не повторюється і тому може бути цінним. Безкоштовна можна створити журнал на сайті http://www.livejournal.ru/

Розміщуючи інформацію в журналі, необхідно дотримуватися певних правил. Зокрема, у блозі не можна:

- Закликати до агресії, ображати за національною, етнічною, географічною, політичною або релігійною ознаками.
- Створювати *флуд* (багато однакових або беззмістовних повідомлень) у журналах, спільнотах та коментарях, розповсюджувати повідомлення рекламного характеру.
- Публікувати зображення, вміст яких не відповідає загальноприйнятим нормам і є неприємним для більшості користувачів.
- Розміщувати у повідомленнях, коментарях чи фотоальбомах зображення, якщо люди, котрі на них зображені, або особи, що мають на них права, не бажають цього.
- Приписувати собі авторство чужих текстів і зображень, вести щоденник від імені іншої особи.

Власник блога може визначати права доступу користувачів до свого журналу та окремих його записів, а також видаляти блог та переводити його в режим спільноти. Надається також можливість зберегти журнал на своєму комп'ютері.

Щоб почати вести журнал, не потрібно спеціальних знань. Навпаки, внаслідок постійної творчої роботи над блогом ви набудете багато нових знань та навичок.

Отже, журнал складається з повідомлень, опублікованих різними користувачами. Будь-який користувач може надіслати до журналу повідомлення, а власник — створити список друзів із тих, хто надсилає йому такі повідомлення.

Популярним є також створення *аватара* — невеликого зображення, яке співзвучне настрою або почуттям користувача.

3. УВЕДЕННЯ В HTML

Структура мови HTML

HTTP - це протокол прикладного рівня, розроблений для обміну гіпертекстовою інформацією в мережі Internet. Протокол використовується в одному з найпопулярніших ресурсів мережі Internet - Word Wide Web - з 1990 року.

Реальна інформаційна система вимагає набагато більшої кількості функцій, ніж просто пошук. HTTP дозволяє реалізувати в рамках обміну даними набір методів доступу, що базуються на специфікації універсального ідентифікатора ресурсів (Universal Resource Identifier), що застосовується у формі універсального локатора ресурсів (Universe Resource Locator) або універсального імені ресурсу (Universal Resource Name). Повідомлення в мережі при використанні протоколу HTTP передаються у форматі, подібному до формату поштового повідомлення Internet (RFC-822) або до формату повідомлень MIME (Multiperposal Internet Mail Exchange). HTTP використовується для взаємодії програм-клієнтів із програмами-шлюзами, що дозволяють доступ до ресурсів електронної пошти Internet (SMTP), списків новин (NNTP), файлових архівів (FTP), систем Gopher і WAIS. Протокол розроблений для доступу до цих ресурсів з допомогою проміжних програм-серверів (proxy), що дозволяють передавати інформацію між різними інформаційними службами без втрат. Протокол реалізує принцип "запит/відповідь". При роботі в Internet для обслуговування HTTP-запитів використовується 80 порт TCP/IP.

В даний час у практиці World Wide Web реально використовуються тільки три методи доступу: *POST, GET, HEAD.*

GET - метод, що дозволяє одержати дані, задані у формі URL в запиті ресурсу. Якщо посилаються на програму, то повертається результат виконання цієї програми, але не текст програми. Додаткові дані, які треба передати для обробки, кодуються в запит ресурсу. Існує різновид методу GET - умовний GET. При використанні цього методу сервер відповість на запит тільки в тому випадку, якщо будуть виконані умови передачі. Це дозволяє розвантажити мережу та позбавити її від передачі непотрібної інформації. Умова вказується в полі "if-Modified-Since" заголовка запиту. При використанні методу GET у поле тіла ресурсу повертається власне викликана інформація (текст HTML-документа, наприклад).

HEAD - метод, аналогічний ***GET***, але він не повертає тіло ресурсу. Використовується для одержання інформації про ресурс. Умовного ***HEAD*** не існує. Даний метод використовується для тестування гіпертекстових посилань.

POST - цей метод, розроблений для передачі великого обсягу інформації на сервер. Ним користуються для анотування існуючих ресурсів, відсилання поштових повідомлень, роботи з формами інтерфейсів до зовнішніх баз даних і

зовнішніх програм, що виконуються. У відмінності від GET і HEAD, у POST передається тіло ресурсу, що і є інформацією з поля чи форм, або інших джерел уведення. У перших версіях протоколу були визначені й інші методи доступу (DELETE, наприклад), але вони не знайшли належного застосування. Багато функцій, що покладали на ці методи, можна успішно виконувати через POST.

Елементи HTML

Одним з інструментаріїв створення WEB- сторінок є мова HTML. Крім того на даний момент є досить перспективна мова XML. Основним елементом HTML є дескриптор (tag).

Тег - це основний елемент кодування, прийнятий в стандарті HTML. В документі HTML все залежить від дескрипторів. Дескриптори визначають правила розмітки в документі. Всі дескриптори беруться в кутові дужки < >. Наприклад дескриптор розриву абзацу <P> (Page), дескриптор <HR> - розмітка горизонтальною лінією (horizontal role). Всі дескриптори мають певні властивості, які називаються атрибутами. Атрибут – управляючий елемент дескриптора, відповідно розміщується всередині кутових дужок дескриптора.

Існує два типи дескрипторів.

1. ***Контейнерні*** – це дескрипторна пара, яка складається з початкового (відкриваючого) і кінцевого (закриваючого) дескрипторів а також вмісту контейнеру над яким проводиться операція. Запис контейнерного дескриптора:

<*TAG*> вміст контейнеру </*TAG*>

де, TAG – ім’я реального контейнерного дескриптора.

Таким чином, контейнерний дескриптор виконує задачу розмітки над вмістом контейнеру, наприклад:

<*PRE*> Це відформатований текст </*PRE*>

2. ***Одиночний*** дескриптор, або пустий не містить інформації і виконує в основному самостійну задачу. Наприклад, дескриптор <HR> створює горизонтальну лінію в HTML документі.

Отже – ***елементи HTML-документу*** – все те, що знаходиться між початковим і кінцевим дескрипторами а також самі дескриптори.

Елементами документу можуть бути – зображення; фрагменти тексту; форми; таблиці; списки; посилання; текстове поле; кнопки; заголовок документу; основне тіло документу.

Зауваження: В HTML-код не потрібно вставляти ні пробіли ні пусті рядки. Броузер HTML-коду відображає документ згідно дескрипторів і всі зайві пробіли і пусті рядки відкидаються. Тому пробіли і пусті рядки використовуються тільки для того, щоб ваш код мав чіткий і читабельний вигляд. На вашій Web-сторінці пробіли і пусті рядки відображаються за допомогою відповідних дескрипторів.

Структура Web-сторінки

Всі ново створені Web сторінки повинні мати однакову загальну структуру. Обов'язковими елементами цієї структури повинні бути:

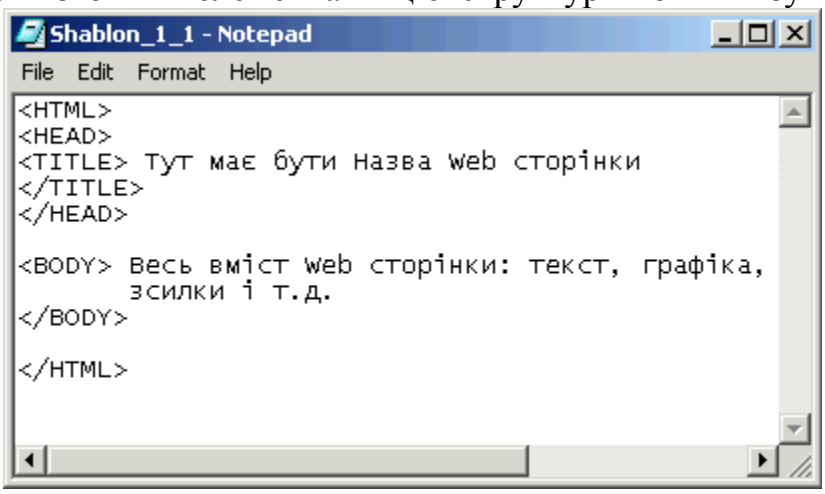


Рис. 3.1. Лістинг програми

Цей лістинг HTML документу може служити у вигляді шаблону для вашої сторінки.

В екрані браузера будемо мати:

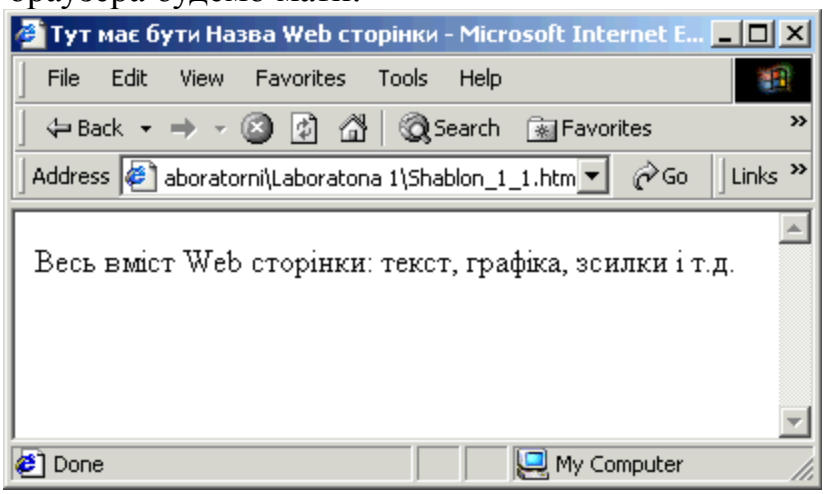


Рис. 3.2. Приклад вашої першої WEB-сторінки

Будь-яка WEB-сторінка починається з дескриптора <*HTML*>, він означає початок WEB-сторінки. Дескриптор </*HTML*> означає кінець WEB-сторінки, все що буде написано після цього дескриптора відображатись на вікні браузера відображатись не буде.

<*HEAD*> - заголовку, що позначає заголовну частину документу HTML;

<*TITLE*> - дескриптор назви документу, розміщує надпис в верхньому лівому куті браузера;

<*META*> – повідомляє браузеру додаткову інформацію про документ;

<*BASE*> – задає базову адресу документу;

<*STYLE*> – визначає стилі форматування зовнішнього вигляду документу;

<*SCRIPT*> – визначає оператори інтерпретації мов програмування

JavaScript і VBScript;

<*LINK*> – (не підтримується більшістю браузерів);

<*BODY*> - це тіло документу HTML, в ньому знаходиться вся інформація що виводиться браузером на екран, тобто вся інформація, що ви хочете подати користувачу.

Дескриптор <*META*> має корисне застосування. Він дозволяє добавити інформацію про вашу сторінку в списки індексації та таблиці каталогів сторінок пошукових машин. Ідея полягає в розміщенні в полі дескриптора атрибутів *NAME* і *CONTENT*.

Приклад застосування цієї ідеї:

<*HEAD*>

<*TITLE*> Вирощування ароматичних трав. Інструкція </*TITLE*>

<*META NAME=”keywords” CONTENT=”*вирощування, ароматичні трави, м’ята, пряноші”>

</*HEAD*>

Дескриптор <*META*> використовується також для автоматичного завантаження сторінок, для створення презентацій, слайдів і анімацій.

Тіло HTML документа. Дескриптор BODY

Найпотужнішим в HTML є елемент *BODY,* котрий визначає тіло документа. Його атрибути дозволяють керувати більшістю візуальних можливостей Web-сторінки: кольором і дизайном фону, кольором тексту, кольорами зв’язків, розміщаючи їх при цьому в середині одного відкриваючого дескриптора <*BODY*>...</*BODY*>.

Фактично з цього дескриптора починається уся видима частина сторінки. Переважно, зображення також вносять привабливість і забезпечують великі візуальні можливості, але основне керування кольором знаходиться саме в дескрипторі <*BODY*>...</*BODY*>. Дескриптор <*BODY*>...</*BODY*> відповідає за те, що називається браузерним дизайном. Це значить, що ще до того, як на сторінці буде розміщене будь-яке зображення, її вже можна зробити барвистою і цікавою, працюючи усього лише з палітрою кольорів і встановлюючи різні атрибути дескриптора <*BODY*>...</*BODY*>. Браузер, за винятком фонових зображень, не запитує цю інформацію на сервері після завантаження сторінки, і якщо ви правильно кодуєте інформацію, зв'язану з фоном, текстом і атрибутами зв’язків, те всі ці дані відображаються прямо в документі сторінки. У такий спосіб підвищується швидкість завантаження й обробки сторінки при ускладненні її дизайну.

У цій лабораторній ви довідаєтеся, як використовувати дескриптор <*BODY*>...</*BODY*> і його атрибути, щоб одержувати ефективні сторінки, що швидко завантажуються та ведуть від нудного стандартного оформлення в царство барвистого дизайну.

Елемент <*BODY*>...</*BODY*> є елементом форматування документа, тому якщо в нього немає атрибутів, то він просто відповідає за демаркацію області

Web-браузера, що повинна містити тіло документа: ***текст, зображення*** і ***медіа.***

Цей елемент має відкриваючий і закриваючий дескриптори і розташовується за контейнером *HEAD*. Разом з елементами *HEAD*, *HTML* і *TITLE* він складає оболонку HTML.

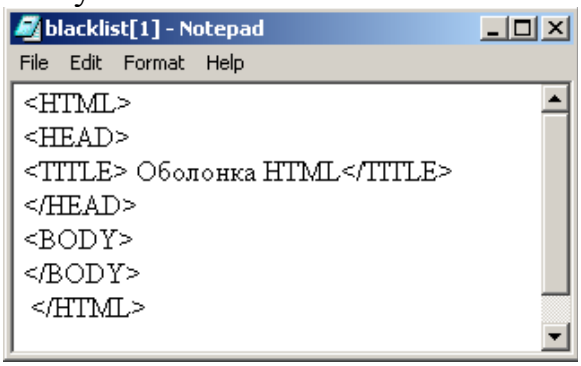


Рис. 3.3. Лістинг програми дескриптора BODY,
що є складовою частиною документа HTML

Хоча для відкриваючого дескриптора <BODY>...</BODY> атрибути не обов'язкові, але саме вони підсилюють його потужність і вплив у звичайному HTML.

Керування кольором фону, тексту і зв’язків

Для керування кольором фона, тексту і зв’язків у дескрипторі *<BODY>...</BODY>* використовуються наведені нижче атрибути (їхніми значеннями служать кольори, представлені у вигляді імен або шістнадцятковому коді формату RRGGBB).

- *text="x"* - колір тексту по замовчуванню.
- *link="x"* - колір зв’язку по замовчуванню. Щоб не порушувати цілісність дизайну, необхідно завжди користатися цим атрибутом і привласнювати йому значення, відповідне палітрі вашого вузла, навіть якщо це значення іншого кольору, прийнятого по замовчуванню у більшості браузерів.
- *vlink="x"* - колір зв’язку після відвідування. Як і у випадку з атрибутом link, якщо не встановлене це значення, те браузер буде використовувати колір по замовчуванні (звичайно це фіолетовий) чи той, що встановлено користувачем. По цих ж причинам, що і для *link="x"*, у рядок BODY необхідно включати відповідний колір зв’язку після відвідування.
- *alink="x"* - колір активного зв’язку. Він з'являється, коли зв’язок стає активний: при натисненні на ній мишею коли над нею проходить курсор миші - це залежить від використовуваного браузера.
- *bgcolor="x"* - колір фону. Раніше в браузерах по замовчуванню застосовувався жахливий сірий колір, але тепер використовується білий. Користувачі можуть самі його встановлювати, так що і вам завжди випливає це робити, навіть якщо ви вирішили користатися

фоновим зображенням (колір фону не перешкодить навіть у цьому випадку).

- *background="URL"* - використовується тільки тоді, коли на сторінці потрібно розмістити фонове зображення, з графічного файлу.

Робота з Web-кольорами

Професіонали звичайно задають кольори як шістнадцяткові значення з 216-колірної Web-палітри. Дивний ефект "зникнення" тексту вийде в тому випадку, якщо зробити однаковими колір активного зв'язку і колір фону. Після натиснення на зв'язку і повернення на основну сторінку, зв'язок знову стане темно-сірим.

Колір фону

При роботі з кольором фону не слід забувати про контраст. Наприклад, чорний текст розташований на білому фоні. Це дуже сильний контраст. Але, якщо встановити темно-синій фон і світло-сірий текст, то контраст зменшиться, що полегшить читання. Це особливо відчувається при читанні такої сторінки протягом більш-менш тривалого часу.

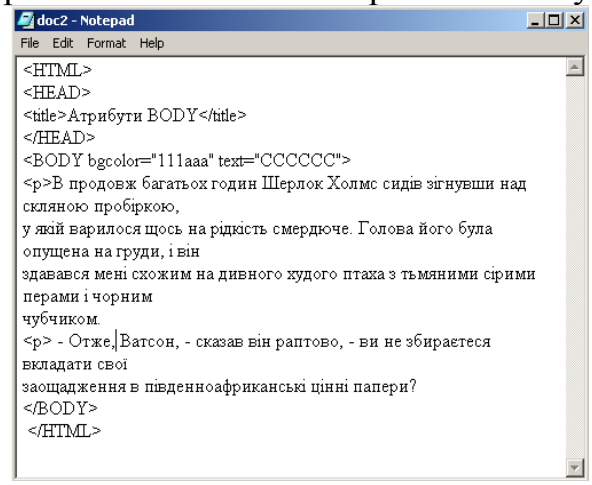


Рис.3.4. Лістинг програми використання атрибутів bgcolor і text для встановлення кольору фону і тексту

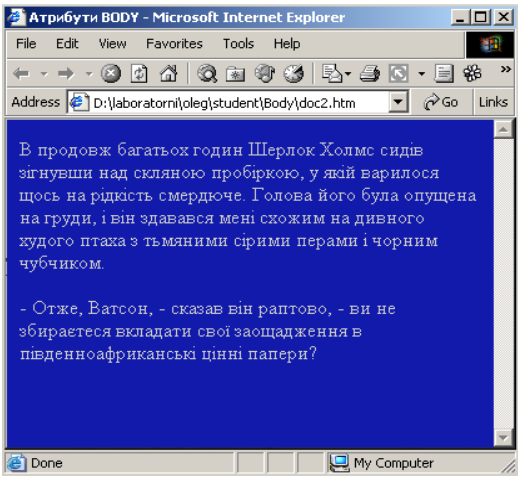


Рис.3.5 Приклад використання атрибутів bgcolor і text для встановлення кольору фону і тексту

Слабкий контраст ускладнює читання, особливо протягом довгого часу читання сторінки з екрану монітору.

Щоб уникнути цієї проблеми, необхідно вибирати колір фону, що добре контрастують із кольорами тексту і зв'язок.

Додавання фонового зображення

Тепер, коли у вас є колірна палітра, можна ввести фонове зображення. Це не є обов'язково і цілком залежить від вашого дизайну.

Стандартним способом розміщення на сторінці фонового зображення є

використання атрибута *background* разом з його значенням - URL файлу зображення:

<BODY background="images/gray_paper.gif">

Дизайн доповнений фоновим зображенням:

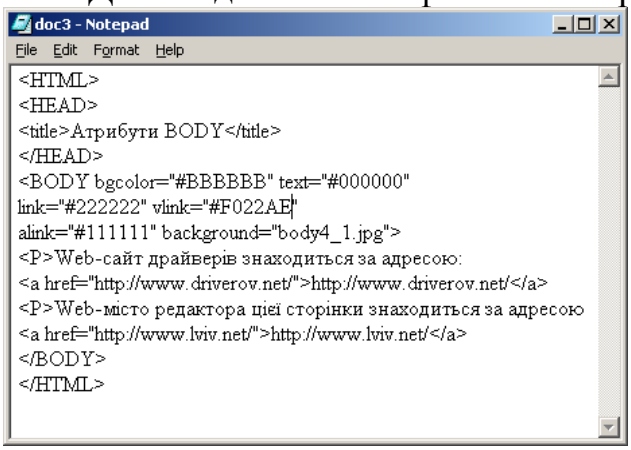


Рис.3.6. Лістинг програми використання атрибутів link, vlink, alink і background для встановлення кольору зв'язків і фонового зображення

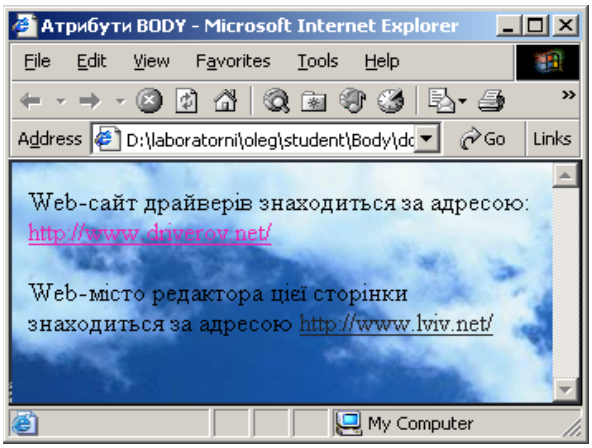


Рис.3.7. Приклад використання атрибутів link, vlink, alink і background для встановлення кольору зв'язків і фонового зображення

Атрибути браузера IE

Існує кілька корисних атрибутів, які можна використовувати тільки в браузері Internet Explorer.

- *bottommargin* - визначає нижнє поле сторінки. Це значення встановлюють у пікселях.
- *topmargin* - визначає верхнє поле сторінки.
- *leftmargin* - визначає ліве поле сторінки.
- *rightmargin* - визначає праве поле сторінки.

У лістингу 4 показано застосування фіксованих полів. Тут використовуються завищені значення, щоб можна було ясно побачити, як усе це працює. Зазвичай, будете застосовувати більш придатні значення.

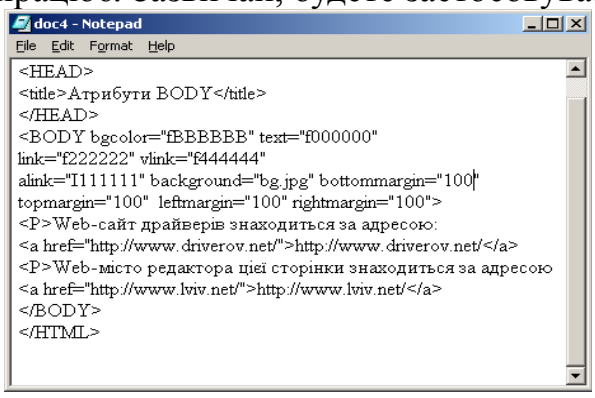


Рис. 3.8. Лістинг програми використання фіксованих полів в браузері Internet Explorer

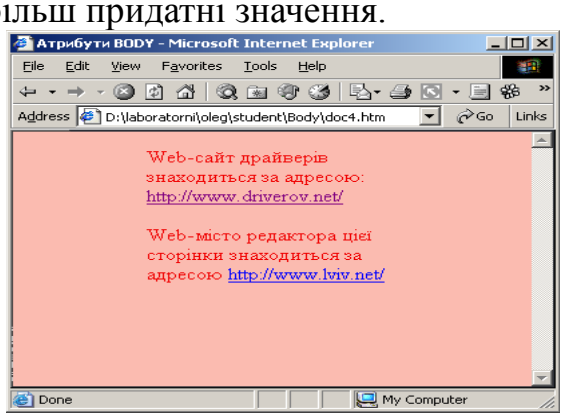


Рис. 3.9. Приклад використання фіксованих полів в браузері Internet Explorer

Значення полів впливають на вид сторінки в Internet Explorer. Цей метод, на жаль, обмежений типом браузера.

Колір зв'язків

Чи завжди колір зв'язку після відвідування повинний відрізнятися від кольору звичайних зв'язків?

Інший колір зв'язку після відвідування допомагає відрізнити, які зв'язки вже переглянуті, а які - ні. Буває, що (через особливості чи дизайну вмісту) зв'язок після відвідування варто зробити того ж кольору, що і звичайні. У такому випадку просто присвоюйте атрибуту ***vlink*** те ж значення кольору, що й атрибуту ***link*.**

Підбір кольорів

Через метод стиску, застосовуваного в JPEG, майже неможливо підібрати відповідність між кольором JPEG і Web-кольором. Спробуйте знайти найбільш придатні кольори тексту і зв'язків. Якщо немає абсолютної необхідності використання JPEG-зображення для фону, краще використовуйте GIF-фон).

Форматування та вирівнювання тексту

Основна частина всіх WEB сторінок розміщується в дескрипторі *<BODY>...</BODY>*. Цей дескриптор дозволяє створити структуру документу, а також абсолютно необхідний для введення в документ складних елементів таких як фоновий звук і зображення. Форматування тексту можна охарактеризувати такими компонентами:

1. **Дескриптор заголовків**, який застосовується для заголовків і підзаголовків вмісту. В цьому дескрипторі є діапазон: *<H1>...</H1>-<H6>...</H6>*-(для прикладу візьмемо текстовий вираз: Використання енергії атома).

   На екрані браузера буде виведено:

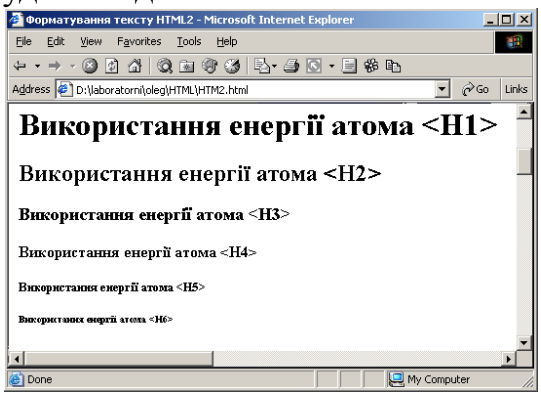


Рис. 3.10. Приклад роботи дескриптора заголовків <H1>…<H6>

2. **Дескриптор розриву *
***, який рівнозначний одному поверненню каретки при його застосуванні документ буде мати такий вигляд:

   Приклад використання подвійного повернення каретки.

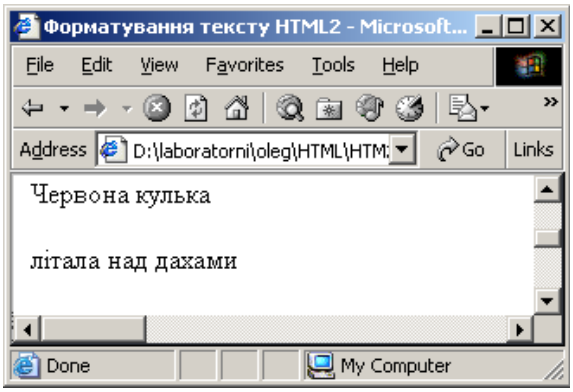


Рис. 3.11. Приклад роботи дескриптора розриву

При застосуванні **дескриптора *<P>* абзацу** будемо мати такий вигляд тексту, який використовується для позначення абзацу з відступом. Запишемо в HTML коді речення таким чином: «Червона кулька *<P>* літала над дахами», а в вікні браузера буде такий вигляд:

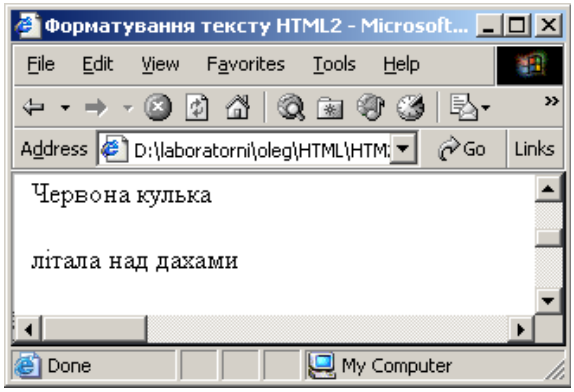


Рис.3.12. Приклад роботи дескриптора абзацу <P>

Існує один спосіб позначення *<P>...</P>* дескриптора абзацу з допомогою відкриття/закриття, дозволяє використовувати для форматування тексту (центрування, вирівнювання по лівій, правій стороні чи по ширині). Приклади застосування буде наведено на інших WEB-сторінках.

3. На цьому прикладі зображено **дескриптор *<PRE>...</PRE>* попереднього форматування тексту.** Запишемо в HTML коді:

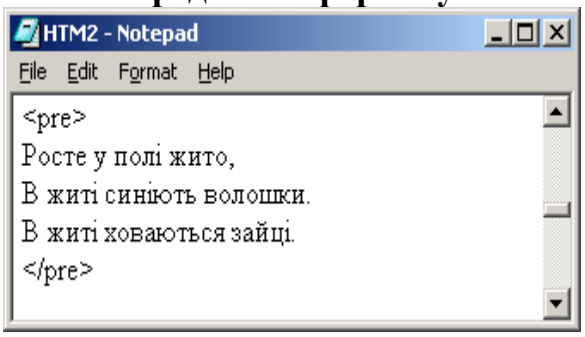


Рис. 3.13. Лістинг програми дескриптора попереднього форматування тексту </PRE>

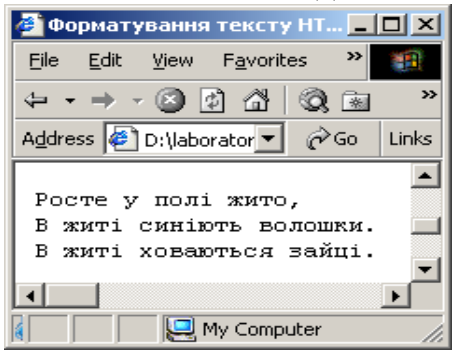


Рис.3.14. Приклад програми дескриптора попереднього форматування тексту </PRE>

Припустимо ви хочете використати стрічку без її звичайного розриву, то використовується дескриптор нерозривності <*NOBR*>...</*NOBR*>. Офіційно він не входить в HTML 4.0, але широко підтримується і часто використовується. При його застосуванні на екрані текст буде мати наступний вигляд нерозривності. Ця стрічка в екрані браузера не буде змінюуватись, коли ви будете зменшувати розмір вікна WEB-сторінки.

Форматування тексту

<*B*>...</*B*>: **Дескриптор для встановлення напівжирного напису тексту.**

<*I*>...</*I*>: *Дескриптор для встановлення курсиву в тексті.*

<*U*>...</*U*>: Дескриптор для підкреслення тексту.

<*TT*>...</*TT*>: Дескриптор моноширинного шрифту.

<*SMALL*>...</*SMALL*>: Дескриптор, що зменшує шрифт.

<*BIG*>...</*BIG*>: Дескриптор, що збільшує шрифт.

<*SUP*>...</*SUP*>: Дескриптор, що робить $^{\text{надстрочний}}$ шрифт.

<*SUB*>...</*SUB*>: Дескриптор, що робить $_{\text{підстрочний}}$ шрифт.

<*STRIKE*>...</*STRIKE*>: Дескриптор, що робить перекреслений шрифт.

В тих місцях де стоїть три крапка (**...**) пишеться текст, що ви його форматуєте.

Дескриптори логічних стилів вказують на характер виділення, а не на спосіб його відображення на сторінці. Браузер на дескриптор логічних стилів реагує таким чином, що сам визначає як відформатувати текст найбільш оптимально по відношенню до іншої частини тексту Web сторінки. Всі дескриптори логічних стилів є контейнерні.

<*EM*>...</*EM*> - *виділений текст*, відмінність від <*I*> - браузер сам визначає стиль виділення тексту.

<*STRONG*>...<*STRONG*>: **сильно виділений текст**.

<*CITE*>...</*CITE*>: *текст у вигляді цитати*.

<*CODE*>...</*CODE*>: текст у вигляді фрагменту коду HTML.

<*DEN*>...</*DEN*>: текст у вигляді визначення.

<*SAMP*>...</*SAMP*>: текст у вигляді фрагменту коду (аналогічно стилю <*CODE*>).

<*KBN*>...</*KBN*>: текст у вигляді назви клавіші клавіатури.

<*VAR*>...</*VAR*>: *текст, котрий визначає перемінну або значення*.

<*ACRONYM*>...</*ACRONYM*>: абревіатура (акронім) і його розшифрування.

Приклад:

<*ACRONYM TITLE*=”Оперативний запам’ятовуючий пристрій”>ОЗУ</*ACRONYM*>

На екрані браузера будемо мати:

ОЗУ

Вирівнювання тексту

Форматування і вирівнювання тексту є важливим способом виводу тексту на екран з точки зору читабельності і естетики. Вирівнювання відіграє велику роль, визначаючи, яким чином текст розташований на сторінці по відношенні до інших об'єктів.

Переважно текст вирівнюють по лівому краю (як мінімум в більшості західних мов), по замовчуванню всі браузери вирівнюють текст по лівому краю сторінки.

Для вирівнювання тексту в HTML є дескриптор *<DIV>...</DIV>* з атрибутом *align="value"*, value набирає таких значень:

- *value = left* вирівнює по лівому краю,
- *value = right* вирівнює по правому краю,
- value = center вирівнювання по центру,
- *value = justify* вирівнює по ширині, як в стандартних дескрипторах HTML.

Дескриптор *<DIV>* є потужним інструментом форматування тексту, він використовується не тільки для розбиття тексту HTML на розділи і вирівнювання інформації в них, але є також одним з основ каскадних листів стилів.

Іноді дескриптор *<DIV>* використовується в парі з елементом *<BLOCKQUOTE>...</BLOCKQUOTE>*. В стандарті HTML 4.0 цей елемент поступився місцем позиціюванню з допомогою листів стилів. Але він залишається простим і ефективним способом створення блоків тексту в сторінках, не використовуючи листи стилів. Важливо зрозуміти, що *BLOCKQUOTE* ніколи не створювався для створення границь і вільних ділянок в тексті, але тепер таке застосування стає звичайним.

Приклад застосування дескриптора *<DIV>* зі значенням *align="left"*, форматування по лівому краю:

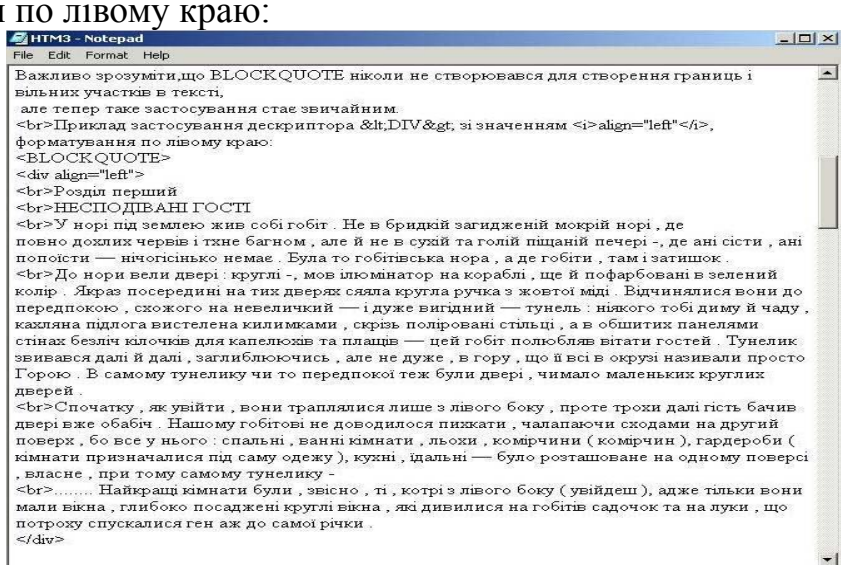


Рис.3.15. Лістинг програми дескриптора <DIV> зі значенням атрибута align="left"

В вікні браузера будемо мати:

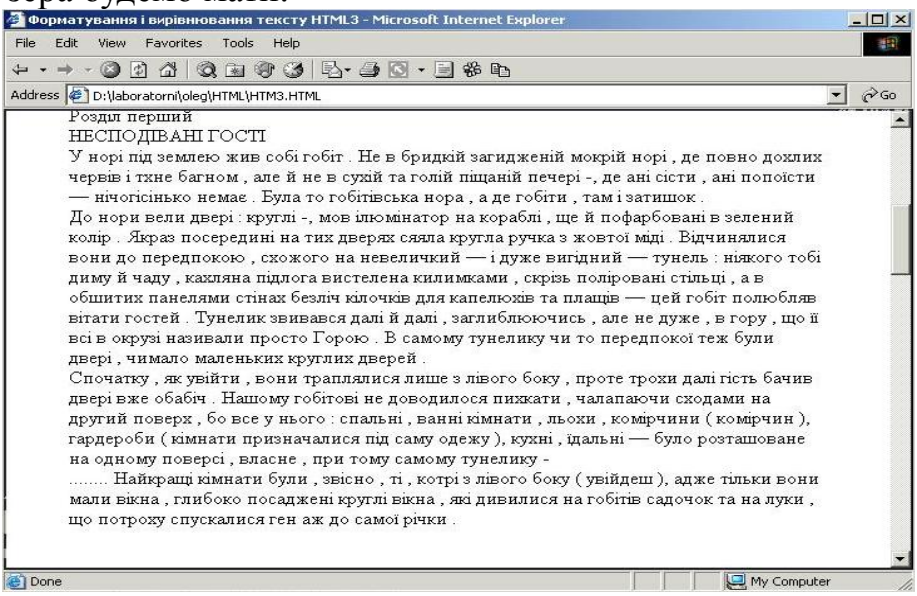


Рис.3.16. Приклад роботи дескриптора <DIV>
зі значенням атрибута align="left"

Приклад застосування дескриптора *<DIV>* зі значенням *align="right"*, форматування по правому краю:

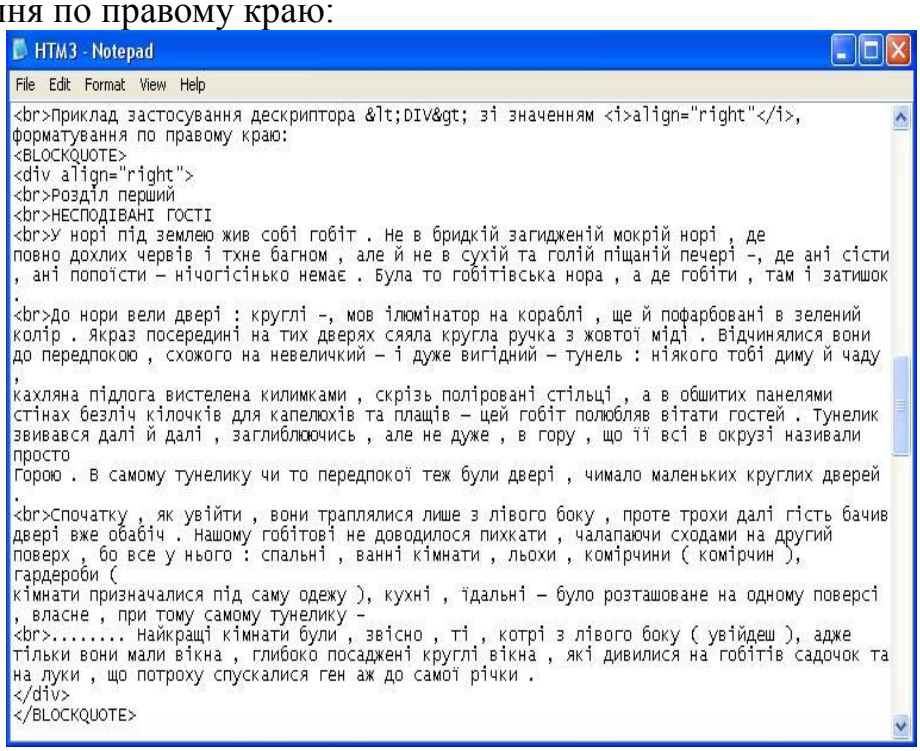


Рис.3.17. Лістинг програми дескриптора <DIV>
зі значенням атрибута align="right"

В вікні браузера будемо мати:

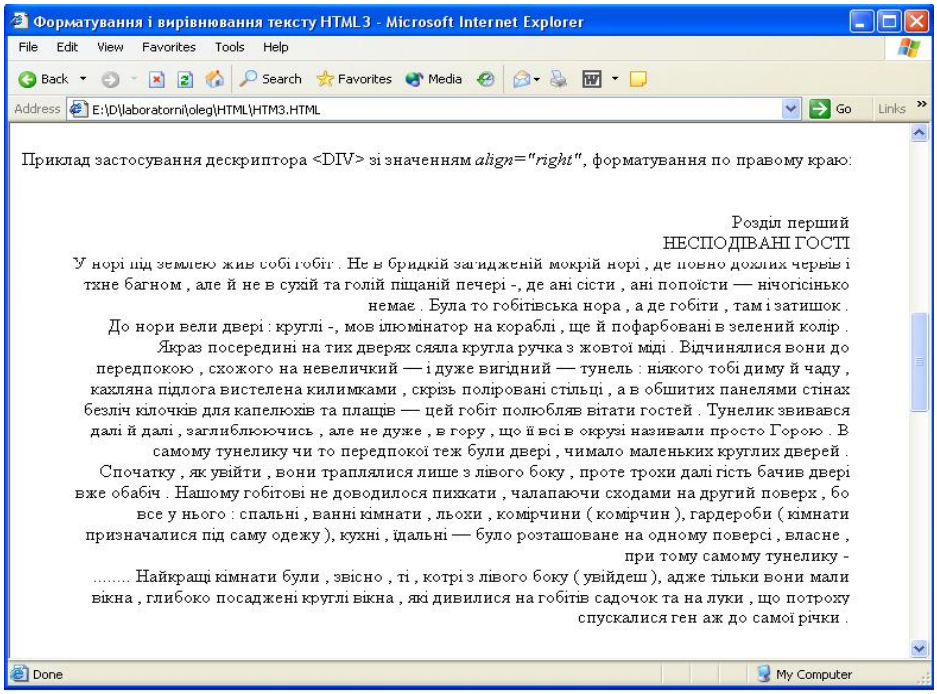


Рис.3.18. Приклад роботи дескриптора <DIV>
зі значенням атрибута align="right "

Приклад застосування дескриптора *<DIV>* зі значенням *align="center"*, форматування по центру:

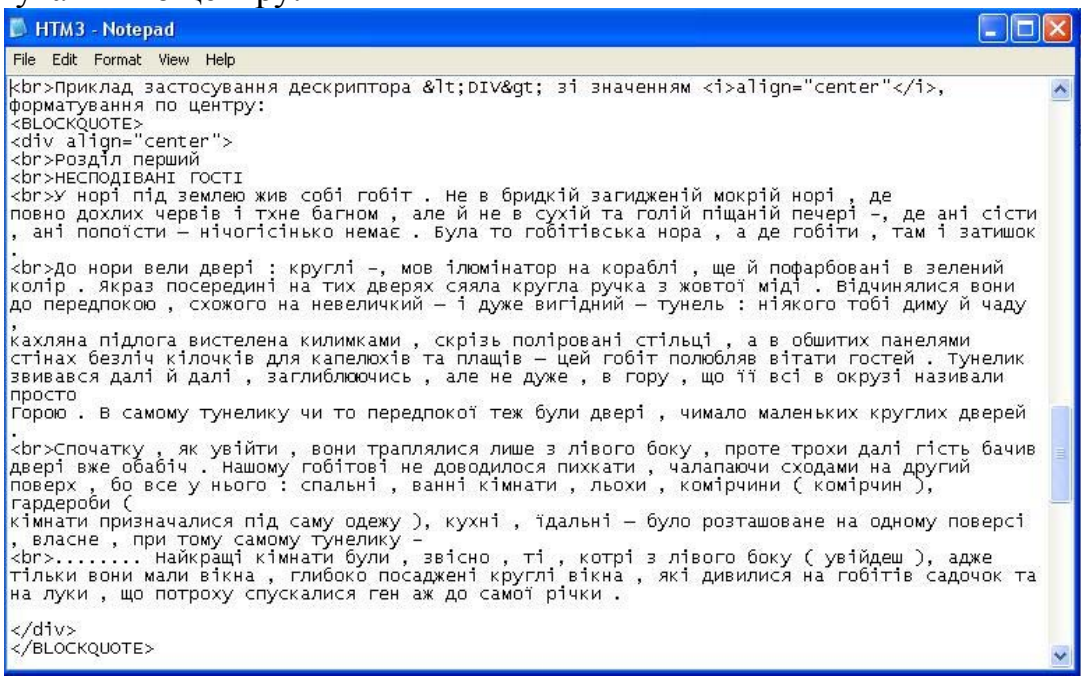


Рис.3.19. Лістинг програми дескриптора <DIV>
зі значенням атрибута align=" center "

В вікні браузера будемо мати:

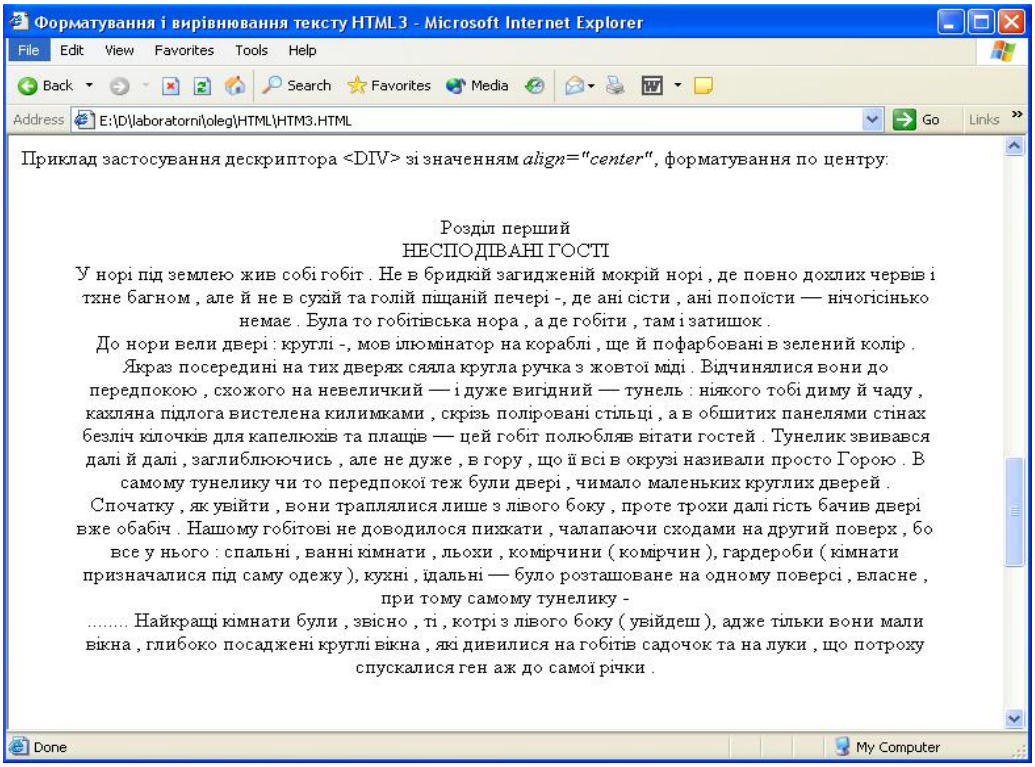


Рис. 3.20. Приклад роботи дескриптора <DIV>
зі значенням атрибута align="center"

Приклад застосування дескриптора *<DIV>* зі значенням *align="justify"*, форматування по ширині:

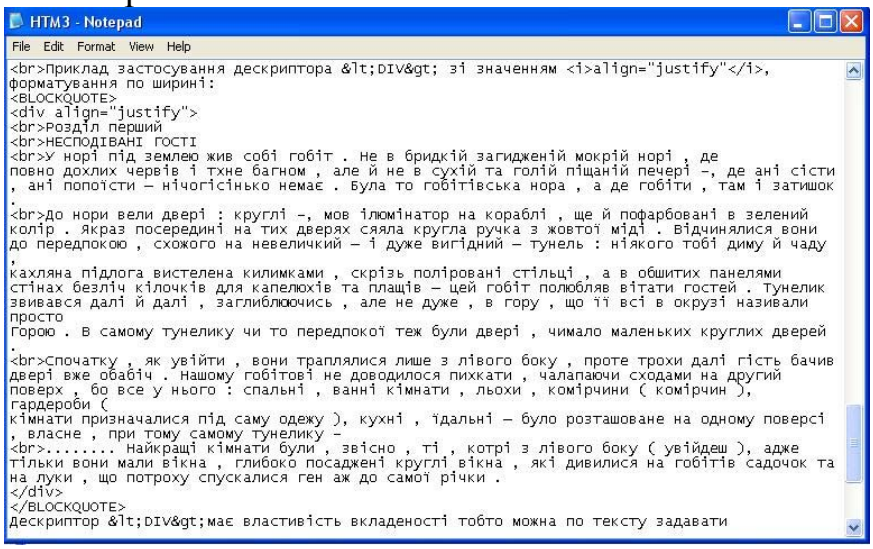


Рис.3.21. Лістинг програми дескриптора <DIV>
зі значенням атрибута align=”justify "

В вікні браузера будемо мати:

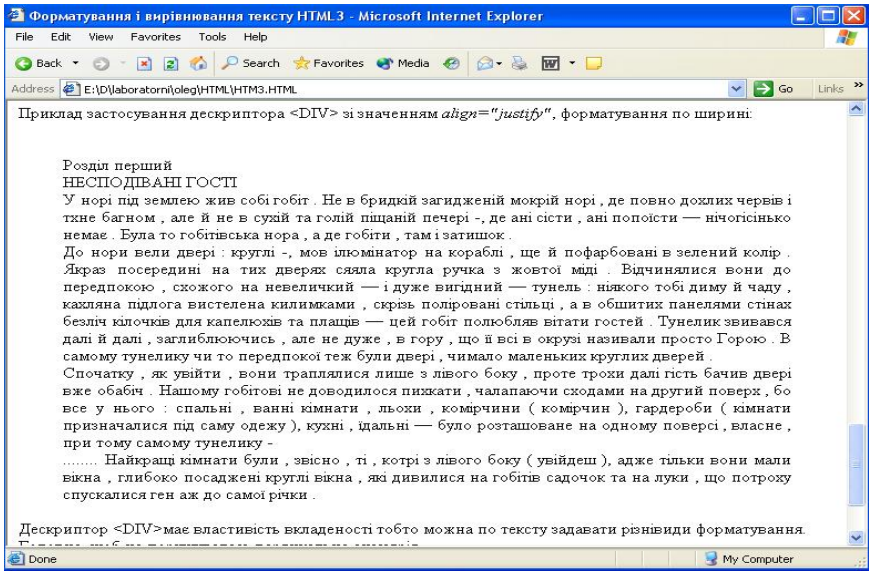


Рис. 3.22. Приклад роботи дескриптора <DIV> зі значенням атрибута align="justify".

Дескриптор *<DIV>...</DIV>* має властивість вкладеності тобто можна по тексту задавати різновиди форматування. Головне, щоб не порушувалась вертикальна симетрія.

Для вирівнювання можна використовувати відкриваючий і закриваючий дескриптор абзаца *<P>...</P>* в поєднанні з тими ж атрибутами, що застосовуються в дескрипторі *<DIV>*. Тобто *<P align="value">...</P>* де value набирає тих самих значень, що і в *<DIV>*. При використанні дескриптора абзацу слід пам'ятати для форматування, що він повинен завжди закриватись </P>, і всі властивості форматування будуть стосуватися тільки одного абзацу. Слід пам'ятати про вертикальну симетрію.

Робота з нумерованими, маркованими списками та глосаріями

Списки в HTML дозволяють розділити інформацію на логічні послідовності елементів. Створювані на основі стилів форматування тексту, ці списки достатньо стабільні, так як підтримуються браузерами з самих рівнів етапів розвитку мови. Хоча в строгій інтерпретації HTML 4.0 деякі особливості атрибутів, що відносяться до списків, оголошені виключеними, хоча самі дескриптори залишаються в силі.

В HTML існують три типи списків

- Марковані.
- Нумеровані.
- Глосарії або списки визначень.

Для оголошення типу списку використовують дескриптор *<UL>...</UL>*. Будь-який елемент списку маркованого типу повинен починатись з дескриптора *<LI>*. *<LI>* - це дескриптор елементу списку, слід зауважити, що в даному випадку закриваючий дескриптор не потрібний.

Приклад застосування невпорядкованого списку:

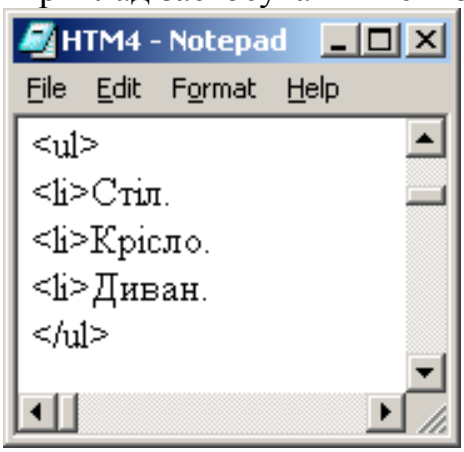


Рис. 3.23. Лістинг програми використання дескриптора маркованого списку <UL>

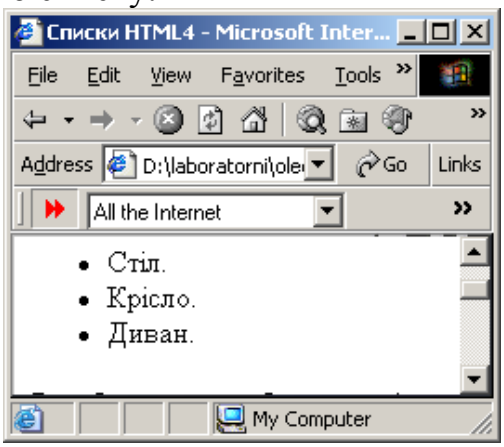


Рис. 3.24. Приклад програми використання дескриптора маркованого списку <UL>

Ви бачите, що браузер після кожного елементу списку природнім чином виконує повернення каретки. Якщо між окремими елементами потрібно задати більші інтервали, їх необхідно задавати з допомогою дескрипторів абзаців і розривів:

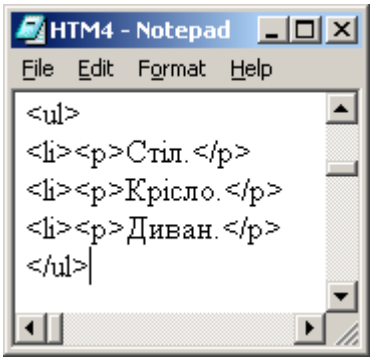


Рис. 3.25 Лістинг програми використання дескриптора маркованого списку <UL> із застосуванням дескриптора абзацу <P>

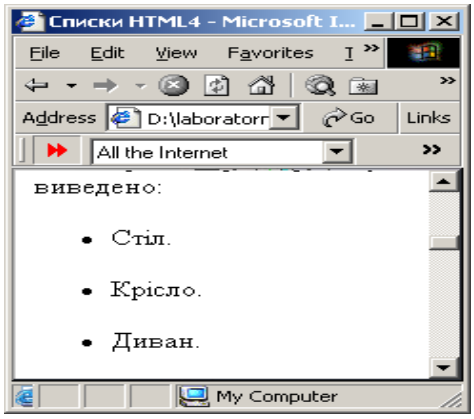


Рис. 3.26 Приклад програми використання дескриптора маркованого списку <UL> із застосуванням дескриптора абзацу <P>

Нумеровані або впорядковані списки - це списки, які мають маркування у вигляді цифр (послідовних числових значень). Нумеровані списки позначаються дескриптором *<OL>...</OL>*. Приклад:

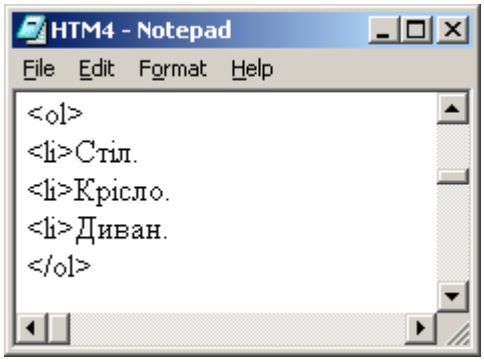


Рис. 3.27. Лістинг програми використання дескриптора нумерованого списку <OL>

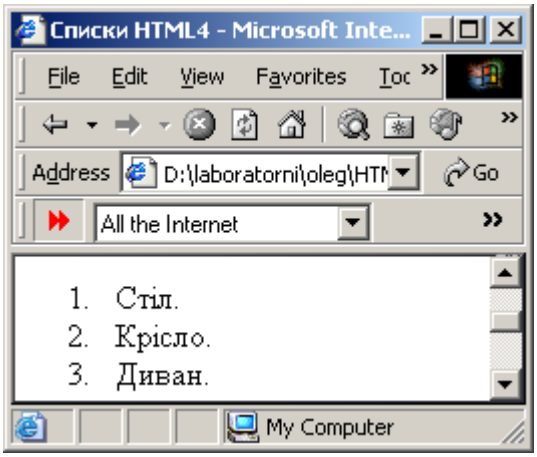


Рис. 3.28. Приклад програми використання дескриптора нумерованого списку <OL>

Списки визначень використовуються, коли інформацію потрібно розміщати з відступом, аналогічно тому, як це робиться в словниках. Ці списки складались для роботи з глосаріями. Синтаксис списків визначень виглядає доволі дивно в порівнянні з послідовною структурою впорядкованих і не впорядкованих списків. Хоча основою дескриптор визначення списку *<DL>...</DL>*. В списку використовується два різних внутрішніх дескриптора - *<DD>* і *<DT>*. Приклад:

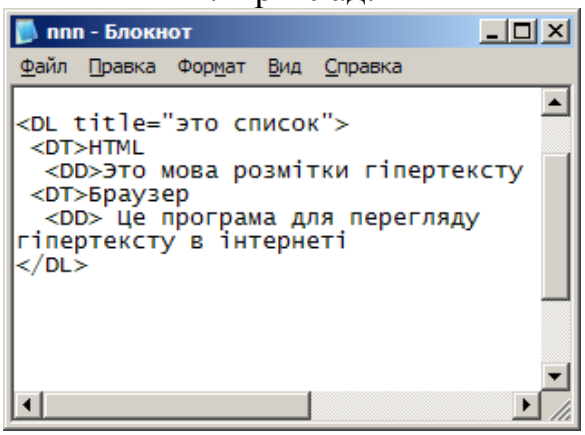


Рис. 3.29. Лістинг програми використання дескриптора списку визначень

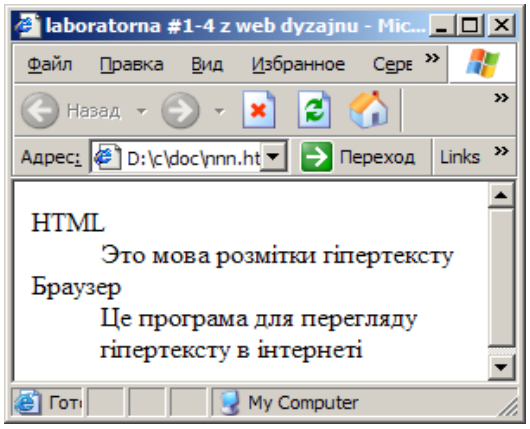


Рис. 3.30. Приклад програми використання дескриптора списку визначень

Існує ряд прийомів, що дозволяє більш ефективно маніпулювати списками і їх атрибутами. До них відносяться вкладені списки, використання списків для відступів і зміни атрибутів списків.

Вкладеність - це розміщення одного контейнера в іншому. Списки можуть вкладатись, забезпечуючи цим загальний стиль представлення інформації. При вкладеності слід пам’ятати про правила горизонтальної симетрії, інакше можуть виникнути проблеми. Прикладом цього може бути:

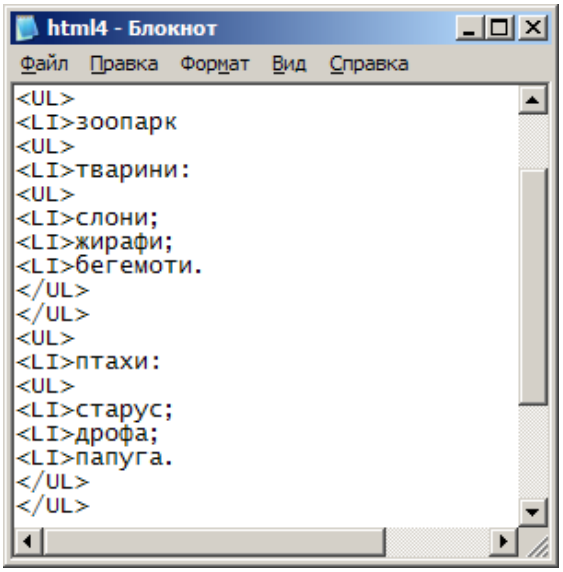


Рис. 3.31. Лістинг програми використання вкладених списків.

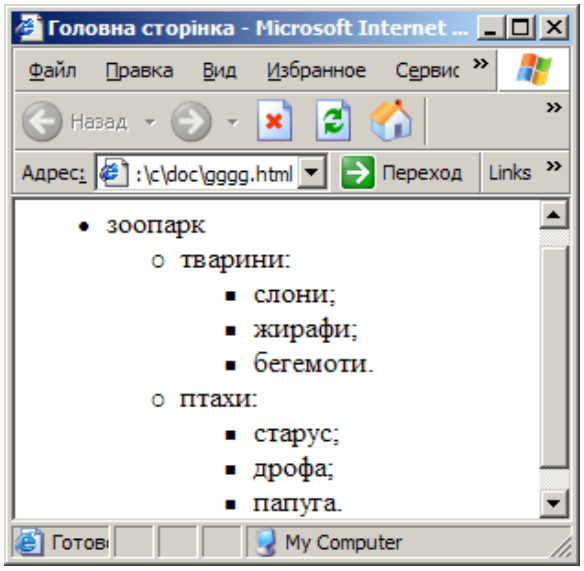


Рис. 3.32. Приклад програми використання вкладених списків.

Іноді виникає необхідність створювати нумеровані списки з нумерацією не від початку, а від певного числового значення, чи створити список з різними маркерами. Для цього використовується атрибути списків як *value* і *type*. Приклад нумерованого списку:

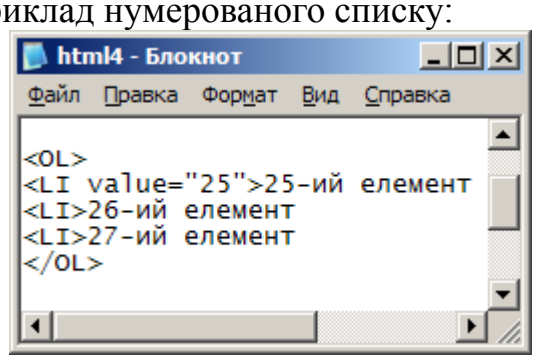


Рис. 3.33. Лістинг програми використання нумерованих списків <OL> із атрибутом value

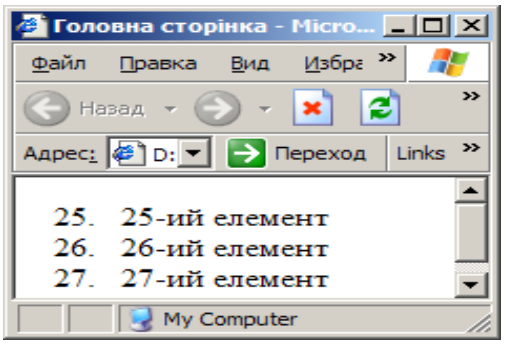


Рис. 3.34. Приклад програми використання нумерованих списків <OL> із атрибутом value

Приклад списку маркерів:

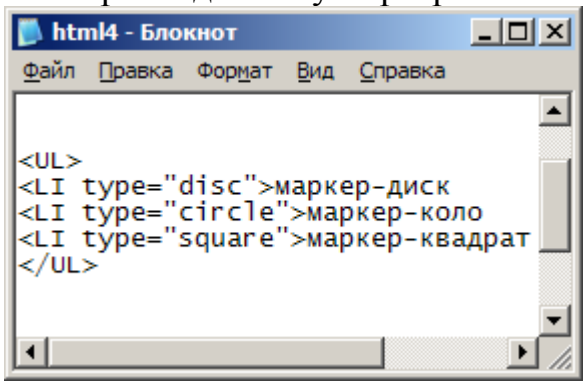


Рис. 3.35. Лістинг програми використання різних маркерів.

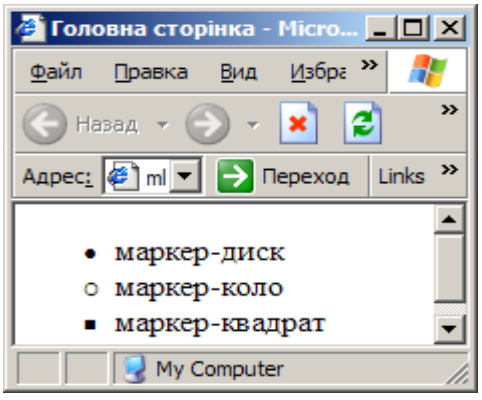


Рис. 3.36. Лістинг програми використання різних маркерів.

Шрифти

Робота з шрифтами - одна з тих областей, де особливо потрібні дизайнери, вміння добитись такого точного сполучення. Робота зі шрифтами в Web настільки важка, наскільки може бути важким HTML-дизайн взагалі.

Багато художників-графіків цілком усвідомлюють важливу роль шрифтів, але, цілком можливо, просто не знають, як "змусити" - шрифт працювати в HTML-сторінках. І навпаки, багато технологів добре знайомі з використанням шрифтів у HTML, але не мають знання типографських концепцій.

Всі шрифти поділяються на певні категорії, сімейства і накреслення

Шрифтів так багато, що їх необхідно поєднувати в групи, - інакше в цій "юрбі" легко загубитися.

Існує три основних види таких груп.

- **Категорія** - є головною *групою*, чи сімейством. Можете вважати її “національністю" шрифта.
- **Гарнітура** - у межах моєї національності мене ідентифікують по призначенні до *визначеної родини*. Таким ідентифікатором є назва шрифту. Назва родини, до якого належить шрифт (гарнітура шрифту), і є його "прізвищем".
- **Накреслення** - це "неповторний зовнішній вигляд" окремого шрифту, що має при цьому визначені гарнітуру і категорію.

Категорії шрифтів

Нижче перераховані головні сімейства, чи категорії шрифтів.

- **Serif** - це стандартна, знайома багатьом група, що характеризується насічками на буквах.
- **Sans-Serif** - розповсюджена в Web-дизайні група шрифтів без насічок. Сімейства цієї категорії характеризуються заокругленістю букв і відсутністю на них насічок.
- **Monospaced** - у цій групі кожна буква шрифту має ту саму ширину. На російській такі шрифти називаються моноширинними, а також *машинописними*, тому, що нагадують шрифт однакової ширини старомодних друкованих машин.
- **Script** - у цю категорію входять сімейства всіх шрифтів, схожих на рукописні.
- **Decorative** – це група (іноді називають Fantasy) відрізняється спеціальними декоративними елементами, таким як крапки, штрихи й інші особливості дизайну, застосовані до її гарнітур і накреслень.

Накреслення шрифту

Дуже часто наш зовнішній вигляд залежить від сімейної спадковості. Стрункі чи угодовані, високі чи низькі, сутулі чи прямі - це в основному залежить від нашої природи і, звичайно, харчування і виховання. Так і зі шрифтами. Як видно шрифти мають визначені атрибутами чи накресленням формою <*FORM*>. Накреслення містить у собі товщину, ширину і нахил.

Товщина шрифту

Можна помітити, що накреслення одних шрифтів виглядають темними і важкими, а інших - світлими і стрункими. Деякі мають середню "статуру", тобто середню товщину. Товщина шрифту помітно впливає на його зовнішній вигляд.

- ***Regular (звичайне накреслення***). Це середня товщина шрифту; текст при цьому виходить простий і невигадливий.
- ***Bold (напівжирне*)*.*** Напівжирне накреслення виділяє текст. Лінії стають товстіші, а сам текст небагато ширшим, ніж при звичайному накресленні.
- ***Light (світле).*** Тонкі, світлі накреслення шрифтів роблять менш сильне враження, чим звичайні чи напівжирні, але вони незамінні, коли потрібно легкий і простий зовнішній вигляд.

Ширина шрифту

Накреслення шрифтів можуть мати різну ширину - місце, яке зайняте накресленням уздовж горизонтальної осі. Найбільш поширені два її види.

- ***Condensed (вузьке накреслення).*** Вузьке, чи *стиснуте* накреслення, при якому ширина букв менша, ніж у нормальному накресленні.
- ***Expended (широке накреслення).*** Деякі дизайнери називають широкий шрифт *розширеним*. На відміну від вузького, широке накреслення займає більше місця по горизонталі.

Нахил

Нахил шрифту - це кут нахилу його знаків.

- ***Italic (курсив).*** Як і напівжирне накреслення, курсив виділяє текст. Він з’являється шляхом нахилу тексту вправо.
- ***Oblique (похиле накреслення).*** Це накреслення тільки для екранного шрифту. Він має менший нахил, чим курсив.

Висота знаків і пропорції шрифту

При роботі зі шрифтами враховується також висота їхніх знаків (кегль) і пропорції шрифтів по відношенню один до одного і до інших елементів на сторінці.

Висоту знаків шрифтів вимірюють у різних одиницях, в тому числі в пунктах і пікселях. Розмір у пунктах засновано на типографських одиницях виміру, а в пік селях - на використанні екранних пікселів для інтерпретації розміру в пунктах.

Звичайно при завданні графічного шрифту використовуються пункти. Однак у HTML-шрифтах для висоти знаків є обмеження, про які мова йтиме трохи пізніше.

Шрифт у 12 пунктів, знаки якого приблизно такої ж висоти виводяться за замовчуванням у більшості браузерів, вважається одною з найлегших для читання і хорошим для тексту тіла.

Дуже важлива *пропорція* даного шрифту стосовно іншого. Висота знаків (кегль) може показати роль шрифту на сторінці: шрифт побільше використовується для заголовків, середній - для тексту тіла, малий кегль - для приміток, адрес електронної пошти і менш важливої інформації.

Орієнтація

Напрямок шрифту називається *орієнтацією*. Напрямок, у якому розташовується шрифт, може вплинути на те, як він буде сприйнятий. Стандартний шрифт розташовується вздовж горизонтального рядка, але він може бути вертикальним, йти в зворотному напрямку, чи вниз приймати визначені обриси.

Горизонтальний шрифт більш стабільний і не так наповнений рухом. От чому він використовується як текст тіла. Однак якщо потрібно зробити яскраве враження, то при дизайні свого вузла можна подумати і про інші варіанти. Орієнтація шрифту може викликати відчуття руху і навіть заінтригувати.

Інтерліньяж

У тексті дуже важлива відстань між рядками (*leading*). В аркушах стилів йому відповідає поняття інтерліньяж (*line hieght*).

Близько чи далеко розташований один рядок від іншої - від цього залежить, чи легко буде читатися текст. Як правило, для тексту тіла інтерліньяж, майже рівний у пунктах власному кеглю шрифту. Занадто великий інтерліньяж також утрудняє читання. Виділення тексту за допомогою нестандартного інтерліньяжу ефективно тільки на невеликих ділянках, таких як заголовки, примітки і т.д. осторонь від основного тексту. Але для основного тексту не варто задавати занадто нестандартний інтерліньяж.

Кернінг і пробіли між словами

Кернінг - це пробіли між буквами того самого шрифту. При звичайних його значеннях букви майже стикаються одна з одною, і це може певним чином вплинути на легкість читання.

Друкар має можливість регулювати ці пробіли. Для цього потрібно доступ до інформації, що міститься всередині шрифтового компонента. Вона знаходиться в таблиці кернінга, що містить математичні значення, пов'язані з одиницями кожної букво форми всередині цього компонента.

Пробіли між словами (spacing) - це горизонтальні інтервали між буквами, які входять в різні шрифтові компоненти. Інакше кажучи, за допомогою таблиць кернінга вже не можна встановити букву ближче до цілого слова чи далі від нього. В свою чергу, пробіли між словами працюють з цілими словами чи фразами, а не з окремими буквами.

Колір шрифтів

Кольорові шрифти додають сторінці своєрідність. Але, як і у випадку з кеглем і накресленням, тут важливий принцип "хорошого - потрохи". Звичайно,

найкраще постійно дотримуватись двох кольорів: один - для заголовків і допоміжного тексту, а інший - для тексту тіла. Однак у дизайні шрифтів потрібні різні кольори, насамперед тому, що вони допомагають виділити яке-небудь слово стосовно інших. Тому, коли справа стосується колірного оформлення шрифтів, варто застосовувати такий прийом, як колірний контраст. На думку більшості дизайнерів, він допомагає привернути увагу до потрібних слів тексту.

У Web колір можна призначати тексту, текстовим заголовкам, зв'язками після відвідування й активних зв'язків. Тут, звичайно ж, є великі можливості, але не забувайте: "гарного - потрохи".

Навіть якщо у вас тільки чорний, білий і сірий кольори, то все рівно завдяки контрасту на ваших сторінках буде почуватися колірне оформлення. Крім того, як уже відзначалося, для залучення уваги до тексту можна користатися такими засобами, як напівжирне, курсив і похиле накреслення. Легкий шрифт м'якше і ніжніше, ніж напівжирний, котрий на екрані сильніше впадав в око.

Дескриптор FONT

З допомогою дескриптора *<FONT>* можна отримати максимальний контроль над HTML-шрифтами. Використовуючи елемент HTML, можна створювати коди для роботи з браузерами, яким "не подобаються" стандарти HTML 4.0. І звичайно ж, усе, що ви робите за допомогою дескриптора *<FONT>*, можна зробити за допомогою стилів. Атрибути елемента FONT:

- ***fase (гарнітура)*** - цей атрибут дозволяє задавати назву необхідного шрифта.
- ***size (кегль)*** - цей атрибут допомагає визначити висоту знаків тексту;
- ***color (колір)*** - встановлюється колір шрифту;

Якщо для визначеного тексту потрібно вибрати назву шрифта, то це можна зробити за допомогою атрибута *face*, використовуючи в якості його значення назва категорії, чи *головного сімейства*.

От приклад коду тексту з атрибутом *face*:

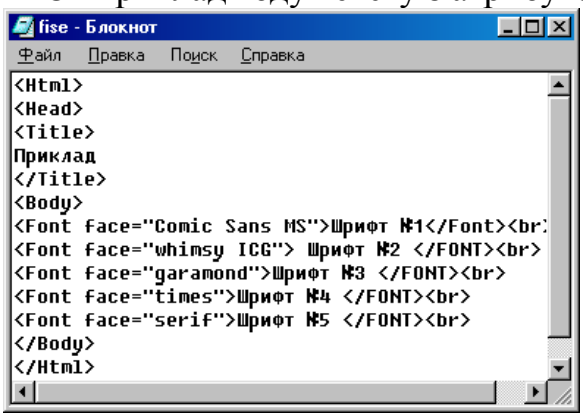


Рис. 3.37. Лістинг програми дескриптора шрифтів <FONT> і його атрибута face

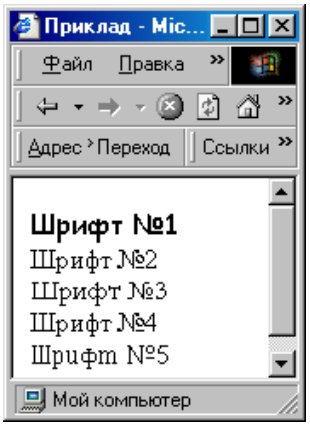


Рис. 3.38. Приклад програми дескриптора шрифтів <FONT> і його атрибута face

Атрибут *size* за замовчуванням має значення – size=”3”.Перед числовим значенням можна ставити знаки „+” або „-”. В такому випадку вказане число буде добавлено до значення по замовчуванню. Діапазон значень розміру кегля шрифту в дескрипторі <*FONT*> складає від 0 до 7.

Деякі дизайнери використовують дескриптор <*BASEFONT*>, щоб задати для всієї сторінки гарнітуру, кегль і колір знаків тексту за замовчуванням. У цьому випадку будуть перевантажені значення браузера за замовчуванням. При цьому в документі можуть вводитися додаткові дескриптори <*FONT*>.

Дескриптор <*BASEFONT*> використовується у верхній частині сторінки (під відкриваючим дескриптором <*BODY*>) з необхідними атрибутами і значеннями.

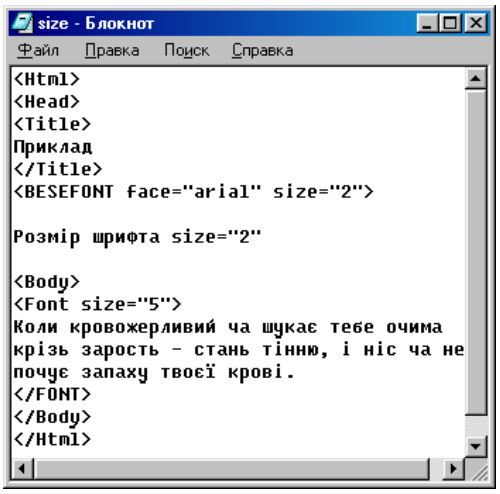


Рис. 3.39. Лістинг програми дескриптора шрифтів <FONT> і його атрибута size

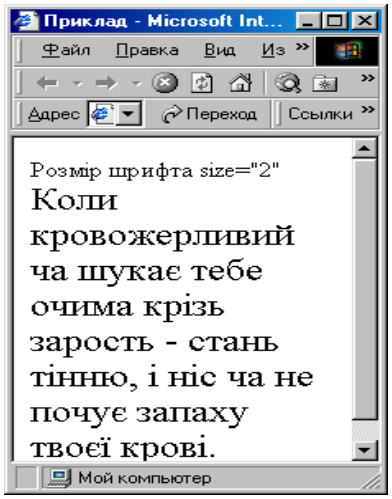


Рис. 3.40. Приклад програми дескриптора шрифтів <FONT> і його атрибута size

Атрибут *color* встановлює колір шрифту. Колір в HTML задається двома способами. Перший спосіб представлення кольору - назви кольору задається англійською мовою, або ж другий спосіб – представлення кольорів у форматі числа RRGGBB (16-ві значення по два розряди на кожен колір в діапазоні 0..255 у відповідно до черговості їх оголошення):

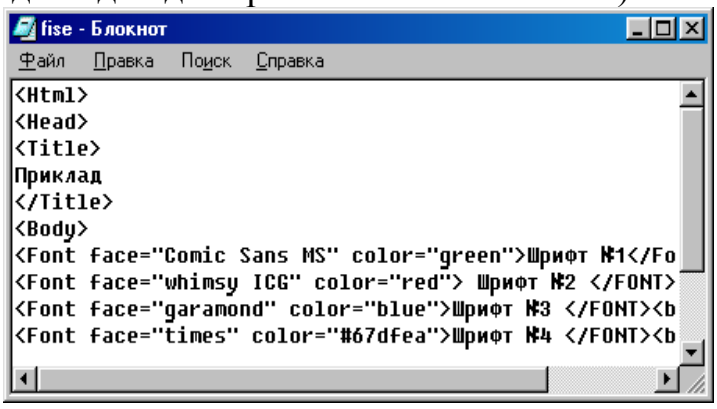


Рис. 3.41. Лістинг програми дескриптора шрифтів <FONT> і його атрибута color

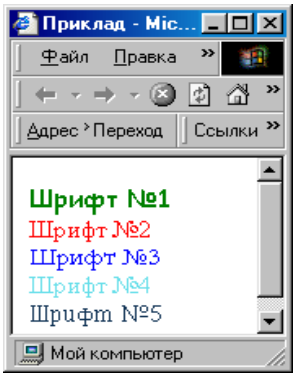


Рис. 3.42. Приклад програми дескриптора шрифтів <FONT> і його атрибута color

4. НАВІГАЦІЯ ПО ВУЗЛУ. ТАБЛИЦІ ТА ФОРМИ

Навігація по вузлі - це механізм, що дозволяє відвідувачу знайти те, що ви хочете йому показати. Від простоти і зрозумілості системи навігації по вузлі, продуманості її елементів залежить, чи досягне відвідувач своєї мети чи покине, так і не розібравшись в нетрях гіпер-документів. Саме зручна навігація, а не красиві картинки, змушує відвідувачів повертатися на ваш вузол, тому що вони знають, що з легкістю знайдуть необхідний матеріал.

Навігація починається з домашньої сторінки вузла і наскрізь пронизує всю структуру вузла, забезпечуючи зручний та швидкий інформаційний “серфінг” користувачу.

Елементи навігації

Вся сутність WEB полягає у зв’язку. Якби його не було, то поняття мережі зводилось би до публікації документів в Internet. Пов’язування ж дозволяє вийти за межі документу і отримати доступ не тільки до інших документів, що відносяться до нього, але й до інших посилань в Internet.

Спочатку цей метод зв’язку називався *гіперзв’язком* технічний прийом зв’язку з іншими документами. На сьогоднішній день в WEB використовують зв’язки не тільки, як гіперзв’язки, а й зв’язок з великою кількістю об’єктів і засобів мультимедіа. Цей новий вид зв’язку називається - *гіпермедіа*.

Існує три основних види гіпертекстових зв’язків:

- абсолютні зв’язки;
- відносні зв’язки;
- локальні зв’язки.

Для зв’язку в нас служить дескриптор якоря *<A>...</A>* з атрибутом *href*. Вся інформація, що записана в дескрипторі якоря *<A>* **тут відображено зв’язок** *</A>*.

Абсолютний зв’язок використовує повну URL адресу. Це означає, що ви використали **абсолютно** всю WEB-адресу, а не її частину. В зв’язку включена інформація про протокол і домен. Ці дані дозволяють перйти на початкову сторінку по замовчуванню цієї WEB-сторінки. Абсолютні зсилки важливі при задані в елементах прив’язки адрес не на вашому вузлі, а на інших вузлах, що знаходяться на інших серверах. Приклад абсолютної зсилки:

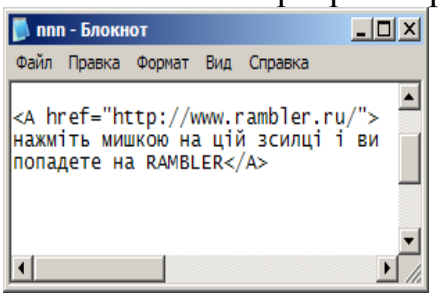


Рис. 4.1. Лістинг програми дескриптора якоря <A> абсолютний лінк

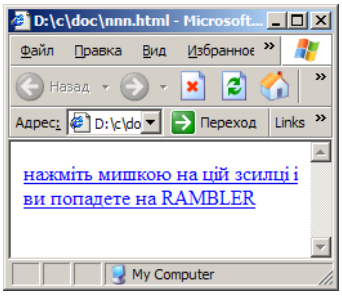


Рис. 4.2. Приклад програми дескриптора якоря <A> абсолютний

лінк

Якщо треба попасти в певний розділ в середині WEB-вузлу тоді потрібно вказувати відповідні каталоги URL. Слід пам'ятати, що URL визначає адресу доступу ззовні, і реальне розташування файлу на диску сервера може бути зовсім іншим. Крім того, файл може зовсім не існувати як такий, а генерувати відповідь на запит.

Хоча для отримання доступу до вузла останньої версії бровзера дозволяють пропустити *http://* і просто виводить www.rambler.ru, це не значить, що в HTML-коді можна виключити з абсолютного URL префікс вказівника протоколу з'єднання. Якщо ви це зробите, тоді отримаєте безкорисний лінк.

Відносний лінк - це лінк, що дозволяє зв'язуватись з файлами , що находяться по тому ж адресу і на тому ж сервері.

Приклад:

<A href="index.html">Початкова сторінка</A>

В вікні бровзера будемо мати:

Початкова сторінка

Іноді задача ускладнюється, якщо напрклад одна HTML-сторінка знаходиться в головній директорії, а інший файл знаходиться в піддиректорії. Тоді у відносній зсилці треба вказувати шлях до того файлу.

Приклад:

*<A href="1/Les1.html">*Початкова сторінка*</A>*

В вікні бровзера будемо мати: Les1.html

Часто при роботі з HTML-сторінками нам потрібно перейти в батьківську директорію чи директорію вище, для цього використовується ".." (дві крапки підряд).

Приклад:

*<A href="../html/ Les2.html">*Початкова сторінка*</A>*

В вікні бровзера будемо мати: Les2.html

Локальні зв'язки - це лінки, котрі дозволяють переміщуватись в межах сторінки в заданому напрямку і в задане місце. Для цього ми використовуємо директиву *name* дескриптора якоря <A>. Внутрішні зсилки є підвидом відносних ссилок.

Приклад:

Місце куди ми переміщаємось: *<A name="up"></A>*

*<A href="#up">*переміщення до потрібного нам місця*</A>*

В вікні бровзера будемо мати:

переміщення до потрібного нам місця

На одній сторінці можна розміщати декілька видів лінків.

Медіалінки. До цього часу розглядали гіперсилки іншими словами *активний текст* тільки з допомогою, але, як сказано вище гіпер, або зв'язуючи можна зробити і медіа зсилки, це стосується малюнків. Використовувати зображення можна таким чином, якщо вставити зображення в середину контейнера, створенного дескриптором якоря, і тоді з'явиться гіперзображення,

закріпленне до відносної чи абсолютної зсилки.

Приклад:

<A href="htm5.html"><img src="1\sample.jpg" alt="бігова доріжка"></A>

В вікні браузера будемо мати відносний лінк на файл htm5.html:

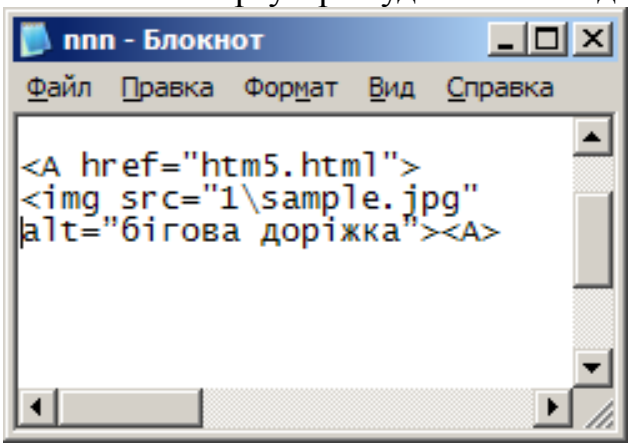


Рис. 4.3. Лістинг програми дескриптора якоря <A> медіа лінка

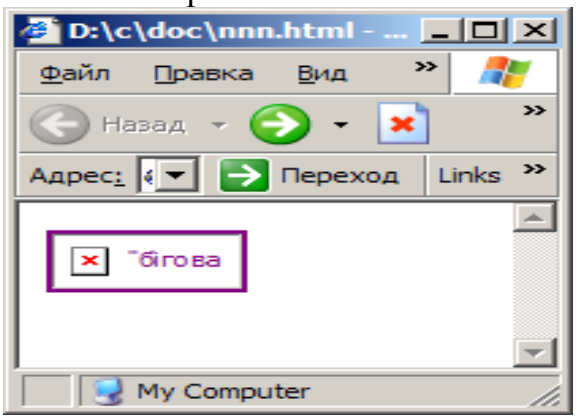


Рис. 4.4. Приклад програми дескриптора якоря <A> медіа лінка

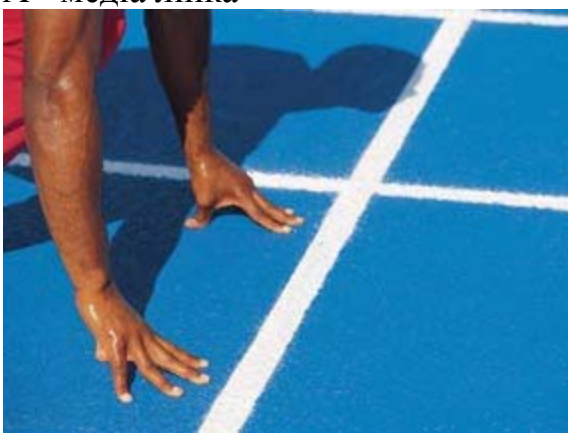


Рис. 4.5. Sample.jpg

Для навігації характерним є оформлення її у вигляді графічної (сенсорної) навігаційної карти, або у вигляді окремих графічних елементів і найпростіший варіант – це марковані списки. Графічну навігаційну карту завжди необхідно продублювати текстовим варіантом глобальної навігації, а для графічних елементів - забезпечити альтернативу текстовими підписами. Це необхідно зробити для користувачів, котрі не відключили у своїх броузерах завантаження графіки. Коли у вас не підключено графіку тоді у вас вікна будуть мати такий вигляд.

Поштові зсилки. Переважно користувачі Internet зв'язуються з вами через вашу WEB-сторінку. Для цього використовується ссилка на адресу електронної пошти. Поштові зсилки можна прив'язувати до зображень, де замість тексту можна розмістити малюнок. Поштові зсилки - дуже ефективна можливість контактувати один з одним, що багато розміщають їх на кожній сторінці свого WEB-вузлу. Рекомендовано розміщувати ссилку електронної пошти в нижньому

колонтитулі.

Приклад:

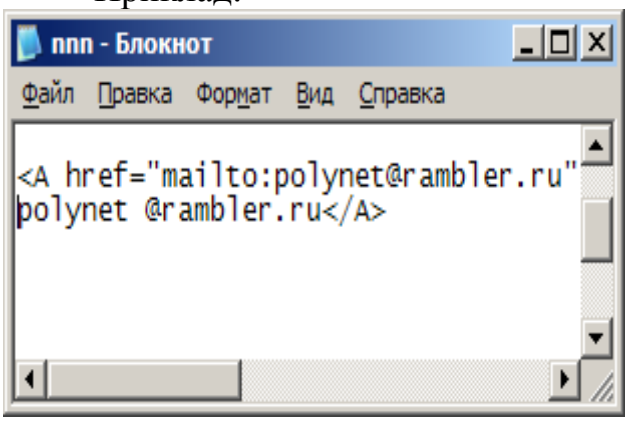


Рис. 4.6. Лістинг програми дескриптора якоря <A> поштовий зв’язок

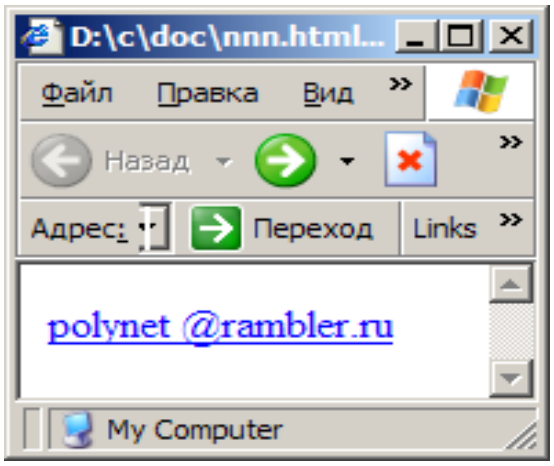


Рис. 4.7. Приклад програми дескриптора якоря <A> поштовий зв’язок

Після того, як ви натисне на поштовій зсилці у вас запуститься поштовий клієнт.

Робота із зображеннями

До цього часу ми ще не розглядали роботу зі зображеннями. На цій сторінці буде представлено роботу з зображення.

Розміщення на WEB-сторінці зображень є першим кроком від простого форматування документів в світ дизайну. Зображення (малюнки, flech, video) надають HTML-сторінкам індивідуальність, колір, форму і зовнішній вигляд. Вони є потужним і важливим елементом WEB. Розміщення зображення на сторінці не підлягають якимось стандартам, а підпорядковуються певним угодам. Є декілька способів це зробити. Наприклад можна використати звичайні дескриптори, таблиці, і на кінець у відповідності зі стандартом HTML 4.0 зображення можна розміщати з допомогою листів стилів.

Для розміщення зображення на HTML-сторінки використовується дескриптор *<IMG>*. Він працює самостійно і йому не потрібний закриваючий дескриптор. Для дескриптора *<IMG>* потрібно вказати URL в атрибуті *src*.

Приклад:

<IMG src="1\mob1.jpg">

В вікні браузера будемо мати відносний зв’язок на початок:

Цієї стрічки достатньо, щоб розмістити рисунок на сторінці, якщо він знаходиться в тому ж місці, що і HTML-сторінка. Як правило розробники WEB-сторінок зберігають свої малюнки в якомусь одному каталозі і дають назви *images* (зображення) або *graphics* (графіка), в залежності від того яке

слово розробнику більше подобається. Деякі розробники створюють лінки на зображення, що находяться за межами WEB-вузлу з використанням абсолютної URL. Це не рекомендується робити по кільком причинам: можливе погане з'єднання з іншим сервером, і малюнок може зовсім не завантажитись. Зображення завжди розміщають в середині розділу дескриптора <*BODY*>.

В дескриптора <*IMG*> є ряд атрибутів для керування зображенням (управління поведінкою, місцем розташування на сторінці, і зовнішнім видом зображення на сторінці).

В дескриптора <*IMG*> є ряд таких атрибутів:

src="x" - джерело зображення (її абсолютний або відносний URL);

width="x" - дозволяє браузеру наперед встановити ширину зображення;

height="x" - це є атрибут висоти і використовують разом з атрибутом *width*, браузер наперед встановлює місце для зображення на сторінці;

border="x" - атрибут, що визначає ширину рамки навколо малюнку;

align="x" - атрибут вирівнювання малюнку на сторінці по горизонталі і вертикалі;

alt="опис" - дозволяє текстовим браузерам виводити га екран замість малюнку його опис, цей опис відображається на екрані будь-якого браузера (не тільки текстового) до тих пір поки не відбудеться загрузка самого зображення. Крім того атрибут *alt* дозволяє виводити на екран елемент підказки з описом, коли вказівник мишки розміщений над зображенням.

hspace="x" - вільний простір по горизонталі; використовується для добавлення вільного місця справа і зліва від зображення. Встановлюється в вигляді числового значення.

vspace="x" - атрибут *vspace* такий самий, як атрибут *hspace*, але для вільного простору по вертикалі.

плаваючі зображення - плаваючі зображення, це обтікання тексту навколо зображення.

Тепер кілька слів про кожен з атрибутів окремо:

Атрибути *width* і *height*

Зокрема, почнемо свій розгляд з *width* і *height*. Ці атрибути обов'язково потрібно включати в дескриптор <*IMG*>, це допоможе браузеру в роботі з зображенням на сторінці. Ні в якому випадку не використовуйте неточні значення атрибутів *width* і *height* (за виключенням GIF-зображень розміром в один піксель). Стандартні значення, відносяться до зображень і завжди повинні бути точними , інакше браузер невиправдано розтягне зображення. Вас іноді може виникнути задача створення фото галереї (розміщення великої кількості рисунків на одній HTML-сторінці) для цього їх з можна масштабувати з допомогою атрибутів *width* і *height*.

Приклад:

<IMG src="1\mob1.jpg">

Розмір рисунку 200x133 пікселі. Зобразимо його зменшеного з розмірами 100x50 пікселів *<IMG src="1\mob1.jpg" width="100" height="50">* і

збільшеного з розмірами 300x250 пікселів *<IMG src="1\mob1.jpg" width="300" height="250">*.

В вікні браузера будемо мати: 

Зменшене зображення з розмірами 100x50 пікселів

Збільшене зображення з розмірами 300x50 пікселів

Для того щоб знайти ширину і висоту малюнку (значення для атрибутів *width* і *height*), необхідно відкрити цей файл в програмі котра призначена для роботи з графікою.

Атрибут *border*

При роботі з WEB-браузерами рамки для рисунків були опцією по замовчуванню, особливо якщо зображення використовується в якості лінка. Зараз по замовчуванню зображення виводиться без рамки. Для того, щоб ваша графіка виводилась без рамок, краще всього явно задати атрибуту *border* значення 0.

Приклад:

<IMG src="1\mob1.jpg" width="200" height="100" border="0">

В вікні бровзера будемо мати:

З допомогою коду цієї стрічки у вас буде захист від появи рамки навколо рисунку при його загрузці в старих WEB-браузерах і WEB-браузерах де по замовчуванню виводиться рамка. Якщо навколо рисунку вам потрібна рамка, тоді атрибут *border* приймає значення "4". Атрибут *border* приймає значення від 0 до 4. При значенні 0 рамки нема.

<IMG src="1\mob1.jpg" width="200"

height="100" border="4">

В вікні браузера будемо мати:

Атрибут *align*

Якщо зображення не є лінком, то їх рамки приймають колір тексту, а якщо є - тоді рамка може бути синьою, або фіолетовою (у відповідності з прийнятим в даному браузері по замовчуванню кольору для лінків), або ж приймати задані користувачем в своєму WEB-браузері кольори, або ж використовувати кольори, задані в дескрипторі <*BODY*> з допомогою атрибутів *link* і *vlink*.

Іноді нам потрібно вирівнювати зображення. Це можна здійснювати декількома способами. По горизонталі для об'єктів по замовчуванню передбачено вирівнювання по лівому краю. Атрибуту вирівнювання *align* можна явним чином присвоїти значення *left* (по лівому краю) або *right* (по правому краю).

Приклад: *<IMG src="1\mob1.jpg" width="200" height="100" align="left">*

В вікні браузера будемо мати:

Приклад:

<IMG src="1\mob1.jpg" width="200" height="100"align="right">

В вікні браузера будемо мати:

Такі самі властивості будемо мати для значення атрибуту *align*, як при форматуванні тексту для *center*. Для форматування тексту в атрибуті *align* використовується ці значення, що наведені вище, але для роботи з зображеннями можна використовувати інші значення атрибуту *align*, що взаємодіють форматування тексту і зображення одночасно:

<IMG src="1\mob1.jpg" width="200" height="100" align="value"> Де *value* набирає такі значення:

- *top* - розміщає зображення вздовж самої верхньої частини стрічки, що її розміщує;
- *middle* - зображення вирівнюється по середній або базовій лінії, що вміщує його стрічки;
- *bottom* - зображення вирівнюється по нижній частині що вміщують його стрічки.

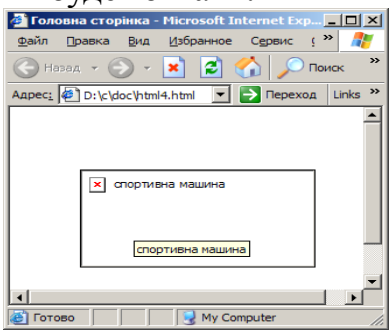
Існує ще декілька значень вирівнювання, специфічних для певних браузерів. Ці значення в себе включають:

- *textop* - вирівнює по верхній частині самого високого тексту або зображення в стрічці;
- *absmiddle* - вирівнювання з абсолютною серединою самого високого оточуючого тексту або зображення;
- *baseline* - вирівнюється по нижній частині;
- *absbottom* - вирівнювання нижньої частини зображення по самому нижньому зображенню або тесту в стрічці.

Атрибут *alt*

Цей важливий атрибут дає можливість забезпечити зображення альтернативним текстовим описом. Наприклад, нам потрібно описати наш малюнок, як "спортивна машина".
Приклад: *<IMG src="1\mob1.jpg" width="200" height="100" border="0" alt="спортивна машина">*

В вікні браузера коли підведемо курсор миші будемо мати:



Для користувачів котрі не працюють з графікою, або ж для тих в кого поганий зір, або обмежений канал зв'язку і вони змушені використовувати текстові браузери, атрибут *alt* надає хороший засіб для опису того, що могло б бути на цьому місці при використані графіки.

Ще одним прикладом того, як можна використовувати атрибут *alt* є те коли браузер завантажує графіку, і під час довгого процесу завантаження вже є опис зображення. Це підвищить інтерес відвідувачів до вашої WEB-сторінки.

Цей атрибут потрібно постійно використовувати, за одним винятком, коли зображення є однопіксельна графіка, що використовується для коректування місця розташування об’єктів на сторінці. В такому випадку атрибут *alt* можна пропускати, або ж задавати без значення.

Приклад: *<IMG src="1\mob1.jpg" width="200" height="100" border="0" alt="">*

Атрибути *hspace* і *vspace*

Значення для атрибутів *hspace* (вільний простір по горизонталі) і *vspace* (вільний простір по вертикалі) є числовими.

Приклад: *<IMG src="1\mob1.jpg" width="200" height="100" border="1" alt="спортивна машина" hspace="50" vspace="50">*

В вікні бровзера будемо мати:

Плаваючі зображення

З допомогою комбінацій атрибутів дескриптора *IMG* можна отримати гарне динамічне розташування графіки і тексту. Хоча таблиці і каскадні листи стилів можуть більш ефективно вирішити цю проблему.

Щоб добитися обтікання рисунку, спочатку треба його вирівняти. Навіть для вирівнювання рисунку по лівому краю (а це завжди задається по замовчуванню) обов'язково використовуйте атрибут **align**. Без цього обтікання не отримаємо. Атрибут *align* має такі значення, як:

top – нижній край тексту вирівнюється по верхньому краю зображення;

middle – нижній край тексту вирівнюється по центру зображення;

bottom – нижній край тексту вирівнюється по нижньому краю зображення.

left – вздовж лівого поля;

right – вздовж правого поля.

Для визначення способу обтікання малюнка текстом використовують дескриптор
 і атрибут *CLEAR*: *<BR CLEAR='параметр'>*. Параметр буває:

left – видаляє текст, що міститься справа від зображення, і розміщує його під малюнком;

right – видаляє текст, що міститься зліва від зображення, і розміщує його під малюнком;

all – видаляє текст, що міститься зліва і справа від зображення, і розміщує його під малюнком;

none – запис з цим параметром є еквівалентним до запису *
*.

Створення таблиць

Таблиці дозволяють більш ефективно розмістити на Web сторінці текст і графіку. Вони використовуються також для представлення інформації у виді логічної схеми, що значно спрощує її пошук. В HTML-таблицях може міститись інформація любого виду. З їх допомогою можна створити чітку структуру матеріалу на Web сторінці. Тому таблиці дуже часто використовуються як інструмент макетування для якісного форматування сторінки а також це практично єдиний засіб замальовки всієї таблиці або її окремих частин кольором. Весь цифровий матеріал, поданий на сторінці, як правило, може бути оформлений у вигляді таблиць. Таблиця складається з рядків і стовпців (див. рис. 4.6).

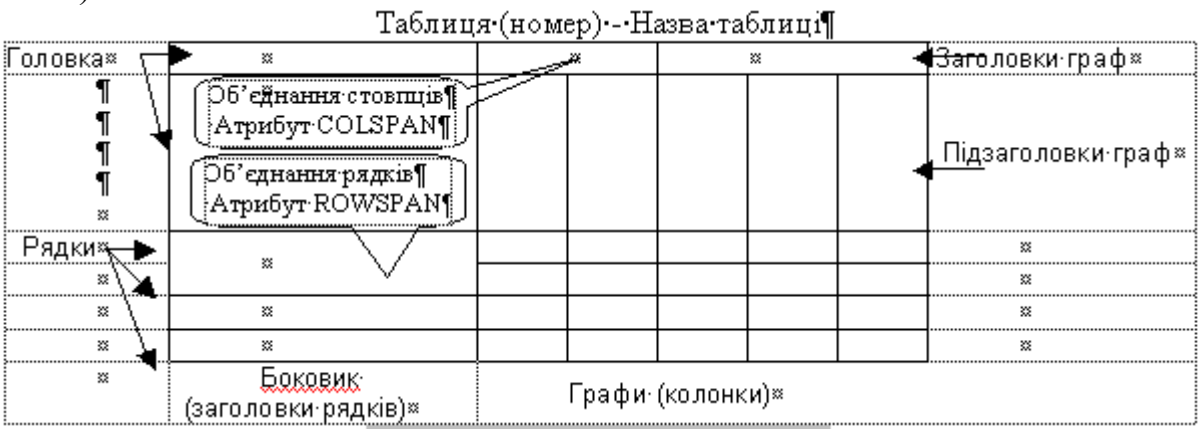


Рис. 4.6. Структура побудови таблиці

Кожна таблиця (якщо вона подана в явному виді, а не як інструмент форматування) повинна мати назву, яку розміщують над таблицею і друкують симетрично до тексту. Назву і слово "Таблиця" починають з великої літери. Назву не підкреслюють. Заголовок граф повинні починатися з великих літер, підзаголовки - з маленьких, якщо вони складають одне речення із заголовком, і з великих, якщо вони є самостійними. Висота рядків повинна бути не меншою 8 мм. Графу з порядковими номерами рядків до таблиці включати не треба.

Код таблиці в HTML міститься в контейнері *<TABLE>*. Дескриптори *<TR>*...</TR> описують один рядок, а за допомогою дескрипторів *<TD>* - комірку таблиці з даними та *<TH>* - комірка заголовку таблиці. Максимальна кількість комірок в таблиці не може перевищувати добутку кількості рядків на кількість стовпців. Закриваючі дескриптори *</TD>* та *</TH>* можуть не використовуватись якщо є в кінці опису рядка є закриваючий дескриптор *</TR>*. Всі рядки повинні містити однакову кількість комірок. Виключенням може бути об'єднання за допомогою атрибуту *COLSPAN=?* декількох комірок сусідніх стовпців в одну, а за допомогою атрибуту *ROWSPAN=?* – декількох комірок сусідніх рядків (де ? – кількість об'єднаних елементів). Ці атрибути описуються всередині дескрипторів *<TD>* та *<TH>*.

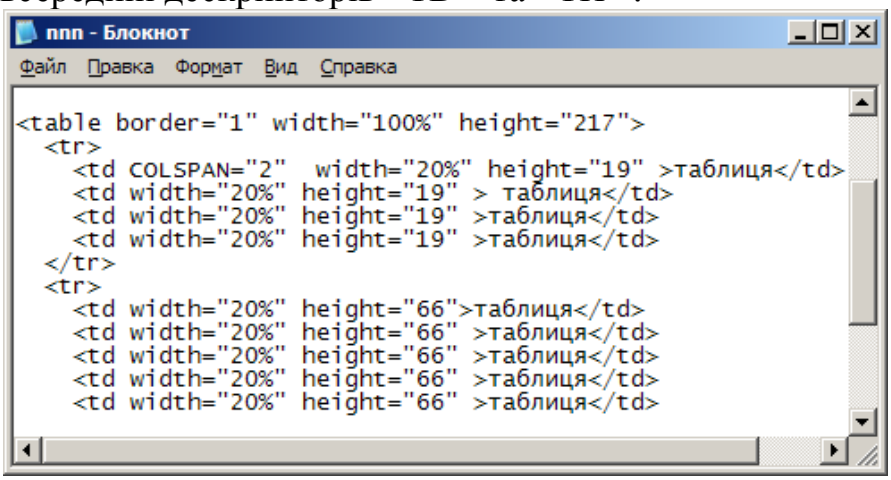


Рис. 4.7. Лістинг програми таблиці з об'єднанням комірок (рядків і стовпців)

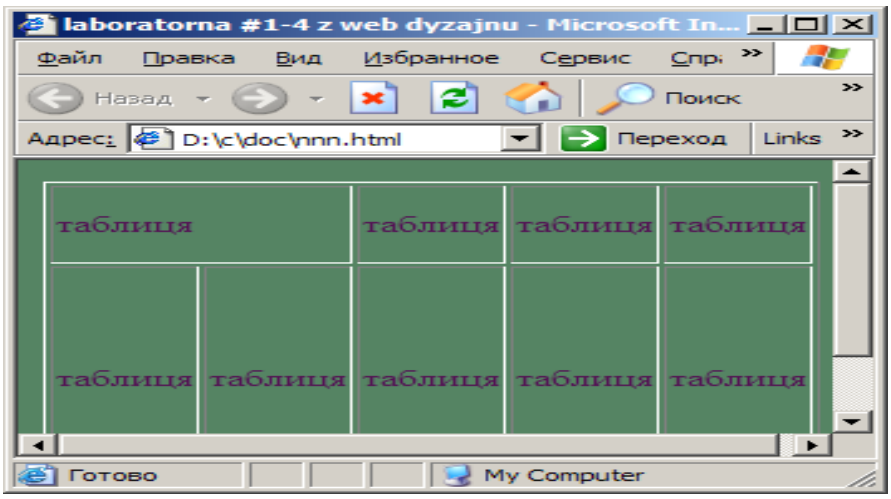


Рис. 4.8. Приклад побудови таблиці з об’єднанням комірок (рядків і стовпців)

З цього прикладу видно, що функція об’єднання комірок описується в описі першої коміри зліва (для рядків) та в першій комірці зверху (для стовпців) а опис використаних для об’єднання інших комірок не проводиться (просто ігнорується). Дескриптор <TH> на відміну від дескриптора <TD> виділяє і центрує вміст контейнера.

Найбільш широко вживані дескриптори і атрибути табличних дескрипторів:

- Визначити таблицю *<TABLE>…</TABLE>*;
- Окантовка (рамка) таблиці *<table border=?>...</TABLE>*;
- Відстань між комірками *<TABLE CELLSPACING=?>*;
- Доповнення комірками *<TABLE CELLPADDING=?>*;
- Бажана ширина *<TABLE WIDTH=?>* (у пікселях - формат абсолютного розміру таблиці);
- Ширина у відсотках *<TABLE WIDTH="%">* (відсотки від ширини сторінки - формат відносного розміру таблиці);
- Рядок таблиці <TR>...</TR>;
- Вирівнювання *<TR ALIGN=LEFT| RIGHT |CENTER| MIDDLE| BOTTOM>*;
- Комірка таблиці *<TD>...</TD>* (повинна бути всередині рядка);
- Заголовок таблиці <TH>...</TH> (як дані, але жирний шрифт і центрування);
- Вирівнювання по горизонталі та по вертикалі *(VALIGN) <TD (<TH) ALIGN=LEFT| RIGHT| JUSTIFY| CENTER| MIDDLE| TOP| BOTTOM| CHAROFF> ALIGN=”CHAROFF” CHAROFF=x* Вирівнювання по зміщенню на “*x*” символів;
- Без переходу на новий рядок *<TD (<TH) NOWRAP>*;
- Розтягування по стовпчику *<TD (<TH) COLSPAN=?>*;
- Розтягування по рядку *<TD (<TH) ROWSPAN=?>*;
- Бажана ширина *<TD (<TH) WIDTH=?>* (у пікселях);
- Ширина у відсотках *<TD (<TH) WIDTH="%">* (відсотки від ширини сторінки);
- Колір рамки *<TABLE BORDERCOLOR=”#$$$$$$” BORDERLIGHT=”#$$$$$$”* Для зміни кольору всередині – *<TD (<TR) BORDERCOLOR=”#$$$$$$”*;
- Заголовок таблиці *<CAPTION>...</CAPTION>*;
- Вирівнювання *<CAPTION ALIGN=TOP| BOTTOM>* (зверху/знизу таблиці);

Вкладені таблиці

Створення вкладених таблиць – це приклад творчого підходу до питань макетування Web сторінки, це популярний метод використовується для організації на всій площині сторінки великої кількості різноманітних елементів (графіка, текст, списки і т.д.). Для цього необхідно визначити таблицю для

всього вмісту сторінки з атрибутом *WIDTH=100%* без рамки в одній з комірок якої розмістити, наприклад, карту посилань, а в іншій комірці – вкладену таблицю з рамками для розміщення, наприклад, новин. Вкладена таблиця розміщується в середині контейнера <*TD*>:

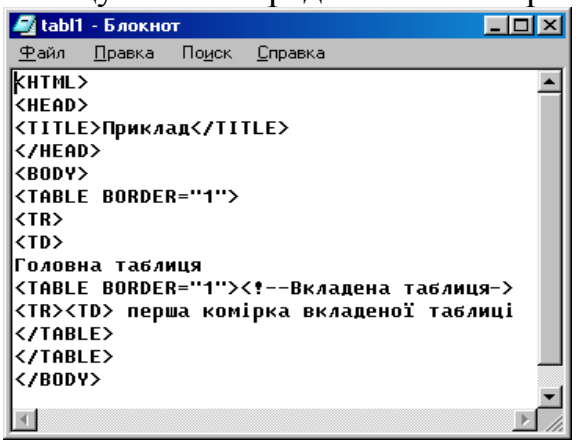


Рис. 4.9. Лістинг програми вкладеної таблиці

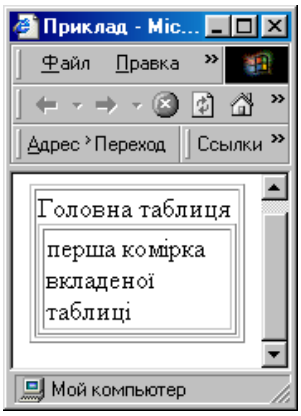


Рис. 4.10. Приклад програми вкладеної таблиці

Але не забувайте, що і в одній таблиці можна розмістити різноманітні елементи Web сторінки в різних місцях робочого поля.

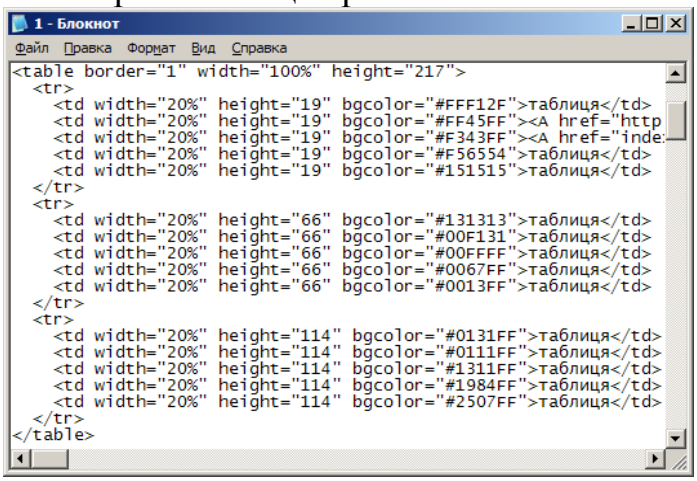


Рис. 4.11. Лістинг програми побудови таблиці з заданням кольору комірок

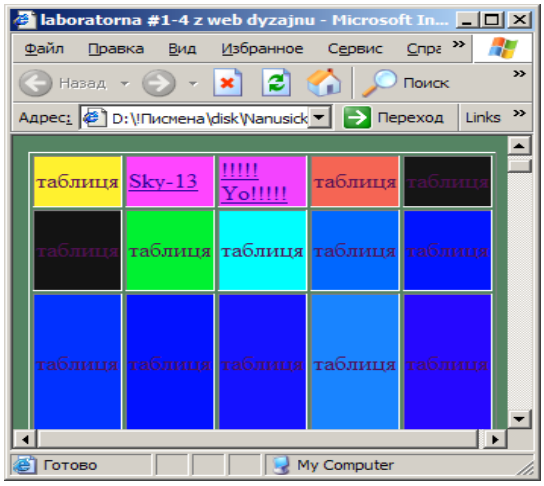


Рис. 4.12. Приклад побудови таблиці із заданням кольору комірок

Завдяки великій кількості властивостей комірок, можна змінювати вигляд і оформлення таблиць.

Робота з фреймами

Фрейми — вічне джерело захоплення та страждань як для Web-дизайнерів, так і для відвідувачів вузлів. Фрейми відбирають і без того малу площу екрану для свого обрамлення. У зв'язку з цим, а також з огляду на відвідувачів, що має обмеження по зору, їхнє використання вимагає акуратного програмування. Крім того, застосування фреймів змушує дизайнера писати більше коду. Тому фрейми, як частина своїх розробок використовують тільки досвідчені дизайнери — та й то нечасто, побоюючись відлякати відвідувачів вузла.

З іншого боку, застосування фреймів дозволяє дизайнеру обновляти тільки частина сторінки, фіксуючи, наприклад, в одному фреймі систему навігації по вузлі і залишивши другий фрейм для показу викликаної сторінки. Але саме цікаве полягає в тому, що фрейми, особливо без обрамлення, дозволяють дизайнеру створити сіткову систему на сторінці. Тим самим реальне застосування фреймів йде від свого первісного призначення — використовуватися винятково як інструмент для форматування сторінки і гнучкого керування нею. У цьому фрейми подібні з таблицями (правда, слід зазначити, що використання таблиць можливо тільки в межах сторінки).

Web-дизайнер повинен добре уявляти собі призначення фреймів. Чи доводиться застосування фреймів до створення розкреслених сторінок із привабливим чи інтерфейсом складних макетів без рамок — у будь-якому випадку в руках конструктора є могутній інструмент.

Структура фреймів.

Перш ніж приступити до практичних занять по оформленню сторінки, розглянемо кілька прикладів застосування фреймів. Подібно до таблиць, вони використовують ті ж ключові елементи — рядки та стовпці. Нагадаємо, що стовпці орієнтовані по вертикалі, рядки — по горизонталі.

Фрейми мають дуже простий синтаксис. Для оголошення рядків використовується атрибут rows, для оголошення стовпців — *cols*. Значення обох елементів встановлюються в пікселях чи відсотках. Так, запис *cols="240,*"* оголошує два стовпці; ширина лівого складає 240 пікселі, а правий – отримує динамічний залишок, позначений символом *, займає ширину вікна, що залишилася.

Щоб організувати більшу кількість фреймів, потрібно описати ширину кожного з них. Наприклад, для оголошення чотирьох вертикальних фреймів, що мають одинакову ширину і займають весь доступний екран, варто записати *cols="25%,25%,25%,25%"*.

Точно так само для оголошення пари горизонтальних фреймів запишемо *rows="240,* "*, у результаті чого, верхній фрейм займе 240 пікселів висоти вікна,

а розташований знизу — висоту, що залишилася. Для оголошення чотирьох горизонтально розташованих фреймів, що мають рівну висоту і займають весь доступний екран, запишемо *rows="25% ,25%,25%,25%"*.

Набір фреймів

Як і у випадку таблиць, для побудови сторінки з фреймами вимагаються три основних елементи. Звичайно, вони складніші від таблиць, але в основі своєї схожі з ними.

Ефектним є вузол Sizzling HTML Jalfrezi, що пропонує останню інформацію з HTML, включаючи безліч зведень про фрейми:

http://vzone.virgin.net/sizzling.jalfrezi/iniframe.htm.

Будь-якій сторінці з фреймами потрібно керуючий HTML-документ, що називається набором фреймів (*frameset*). Для кожного фрейму з набору потрібна своя HTML-сторінка. В сумі, скільки ж сторінок потрібно? Відповідь проста: одна сторінка плюс загальна кількість фреймів.

Набір фреймів — це контрольна сторінка, у якій оголошуються рядки і стовпці фреймів. Крім того, у наборі оголошуються HTML-сторінки, що будуть вставлені в отримані фрейми. З цією задачею справляються два основних дескриптори.

• *<FRAMESET>* — дескриптор оголошення набору фрейму, основні атрибути якого визначають рядки і стовпці. Оголошення набору фреймів завершується дескриптором *</FRAMESET>*.

• *<FRAME>* — дескриптор оголошення окремих фреймів у межах набору. Він містить адресу HTML-документу, що уставляється у фрейм (для цього використовується атрибут *src="x"*, де замість *"x"* записується значення відносного чи абсолютного URL сторінки). Інші атрибути дескриптора *<FRAME>* розглядаються далі.

Важливо пам'ятати, що дескриптор *<FRAMESET>* у HTML-сторінці заміняє дескриптор *<BODY>*. Тому в сторінці з набором фреймів дескриптори *<BODY>* відсутні.

Побудова сторінки з фреймами

Спробуємо створити сторінку з двох вертикальних фреймів, причому розташований ліворуч буде призначений для меню навігації по вузлі.

1. У HTML-редакторі введіть наступний код:

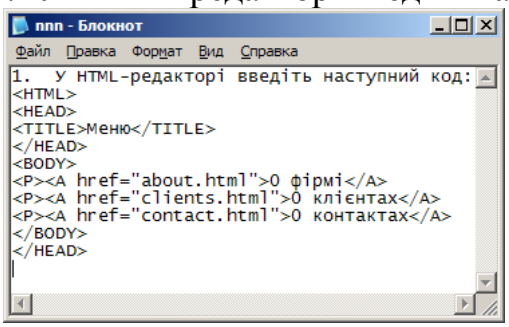


Рис. 4.13. Лістинг програми.

Рис. 4.14. Вигляд програми в броузері.

2. Збережіть код у файлі menu.html.
3. Відкрийте файл у вікні броузера для перегляду роботи коду.
4. Створіть головну HTML-сторінку:

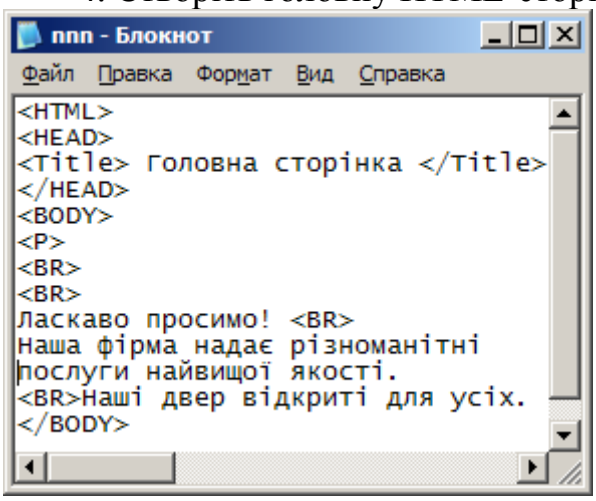


Рис. 4.15. Лістинг програми main.

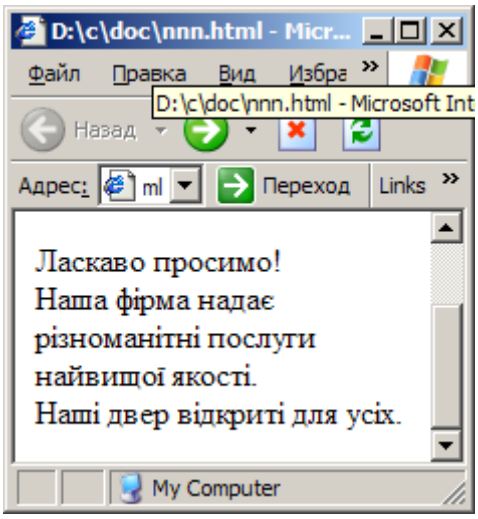


Рис. 4.16 Вигляд програми main в броузері.

5. Збережіть код у файлі main.html.
6. Відкрийте файл у вікні броузера і переконаєтеся, що усе введено правильно.

А тепер створимо саме 2 вертикальні фрейми.

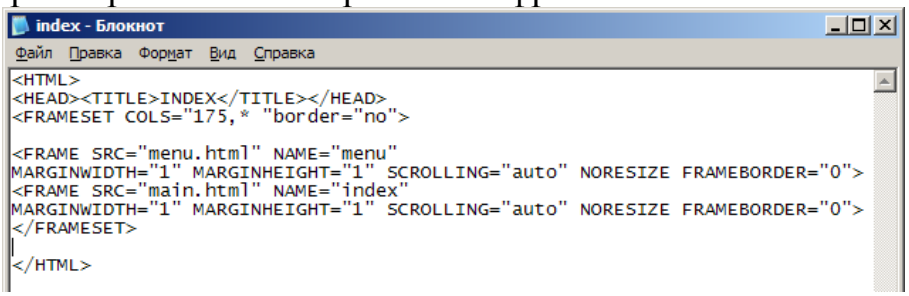


Рис. 4.17. Лістинг програми frame.

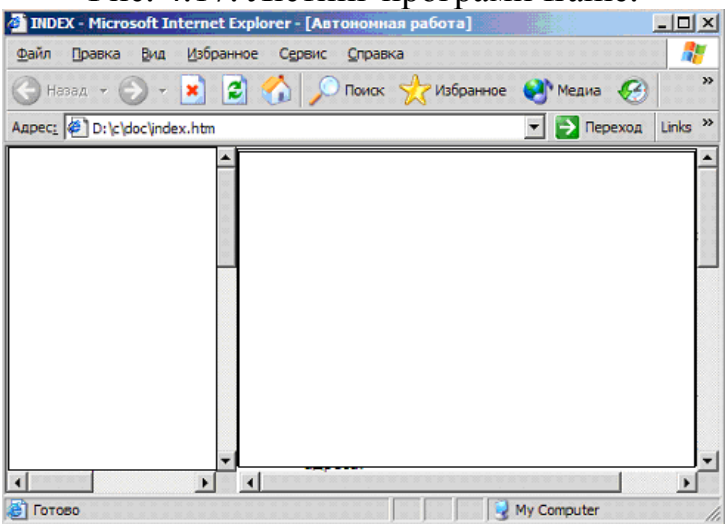


Рис. 4.18. Вигляд програми frame в броузері.

Індивідуальні фрейми з відповідними їм HTML-сторінками додаються за допомогою дескриптора FRAME:

Збережете код у файлі frame.html.

Завантажите файл у вікно броузера. Якщо результати збігаються з мал. , виходить, задача виконана.

Дескриптори <*FRAMESET*> і <*FRAME*>.

Для розширення можливостей дескрипторів <*FRAMESET*> і <*FRAME*> застосовується кілька наступних атрибутів:

•*cols="x"* — як уже відзначалося, цей атрибут описує вертикальні фрейми. Значення "x" задається для кожного фрейму сторінки або в пикселях, або у відсотках. Допускається комбінація одиниць виміру, а також використання символу *, що створює динамічний фрейм.

•*rows="x"* — використовується для створення горизонтальних фреймів. Правила запису ті ж, що й у випадку вертикальних фреймів.

•*border="x"* — використовується броузерами Netscape Navigator 3.0 і більш пізніми для установки ширини границь фрейму в пикселях.

•*frameborder="x"* — використовується броузером Internet Explorer для установки ширини границь фрейму в пікселях. Netscape Navigator 3.0 і більш пізні використовують логічні значення — yes чи no.

•*framespacing="x"* — цей атрибут, котрий використовується броузером Internet Explorer повідомляє ширину границь фрейму.

Далі приведені атрибути, котрі використовуються в дескрипторі <*FRAME*>.

•*frameborder*="x" — керування рамками окремих фреймів.

Браузер Netscape Navigator допускає логічні значення по і yes, а Internet Explorer - ширину границь у пікселях.

•*marginheight*="x" — значення в пікселях, що керує висотою відступу фрейму.

•*marginwidth*="x" — значення в пікселях, що керує шириною відступу фрейму.

•*name*="x" — це дуже важливий атрибут, дозволяє іменувати фрейми для організації зв'язків між HTML-сторінками. Імена повинні починатися з букви , або цифри.

•*noresize* — даний атрибут, якщо не приймає ніяких значень, забороняє зміну встановлених розмірів фрейму.

•*scrolling="x"* — установкою значення *"no"*, *"yes"* чи *"auto"* можна контролювати наявність смуг прокручування. Значення "yes" автоматично встановлює смугу у фреймі, а значення *"no"* забороняє її. Значенням *"auto"* керування смугою прокручування передається браузеру, що розмішає її у своєму вікні при необхідності.

•*src="x"* — значення атрибута представляє відносний чи абсолютний URL HTML-сторінки, що розміщається у фреймі.

Такий багатий вибір атрибутів дозволяє використовувати широку гаму

можливостей при роботі з фреймами.

Приклад керування відступами, розмірами і смугою прокручування

В лістингу приведений код сторінки з фреймами й атрибутами *marginheight*, *marginwidth*, *noresize*, *scrolling* (див. далі).

Приклад набору фреймів

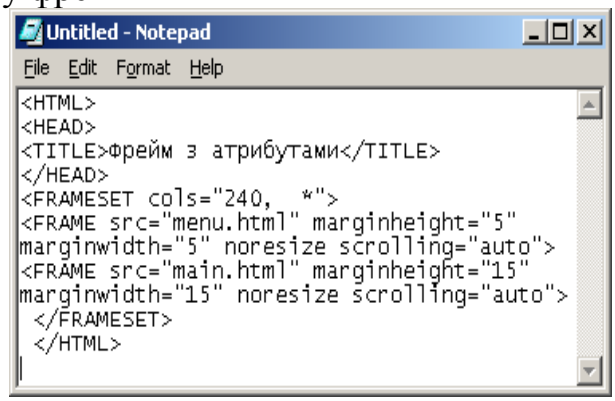


Рис.4.18. Лістинг програми використання атрибутів marginheight, noresize, scrolling

Перша особливість коду з набором фреймів — відсутність дескрипторів *<BODY>*. Тут використовується пара дескрипторів *<FRAMESET>* і *</FRAMESET>*.

Дескриптор *<FRAMESET>* доповнений описом фрейму шириною 240 пікселів і динамічного фрейму (з використанням значення *). За цим оголошенням випливають опису окремих фреймів. Спочатку описується фрейм, розташований ліворуч, а потім — фрейм із правої сторони вікна.

Перший фрейм доповнений оголошенням відступів рівної ширини і висоти в 5 пікселів, що дає небагато вільного простору навколо тексту, розташованого у фреймі. Результатом роботи атрибутів *noresize* і *scrolling* стає поява смуги прокручування в браузері відвідувача при низькому дозволі його екрана.

Використання значення атрибуту *scrolling="yes*" бажаним є тільки у випадках, коли фрейм містить довгий документ. Значення "no" більш прийнятно для фреймів з фіксованою шириною. Якщо ви абсолютно упевнені в тім, що при довільному значенні роздільчої значенні екрану, відвідувач буде бачити весь вміст фрейму, використовуйте значення *"no"*. Оптимальним же варіантом є установка значення *"auto",* що передає керування смугою прокручування браузеру. Значення "*auto*" варто також використовувати у випадку динамічних фреймів.

Ідея заборони зміни розмірів фрейму проста, хоча доцільність її використання дуже спірна. Будемо виходити з того, що атрибут «*noresize*» позбавить відвідувача можливості спотворити вид документа, що міститься у фреймі.

Другий фрейм в лістингу описаний точно так само, як і перший, з однією відмінністю — великими відступами.

Вікна призначення

Для того, щоб ефективно використовувати фрейми, дизайнер повинен вирішити питання завантаження сторінок. Наприклад, у розробленій вище сторінці з фреймами ви вирішили розмістити систему меню з лівої сторони і вікно для результатів повідомлень — із правої. Це традиційне оформлення сторінки з використанням фреймів.

Існує два способи зв'язування HTML-сторінок з визначеними вікнами:

• комбінуючи атрибути *target* і *name* для визначення цільових вікон;

• використовуючи спеціальні цільові імена.

Атрибути *target* і *name* дозволяють відкривати нові HTML-сторінки в наявних фреймах, визначаючи в посиланні цільовий фрейм для сторінки.

Атрибути *target* і *name*

Необхідно завжди присвоювати ім.'я цільовому фрейму. Використовуючи набір фреймів з лістингу, дамо ім'я правому фрейму:

```
<HTML>
<HEAD>
<TITLE>Фрейм з атрибутами target і name</TITLE>
</HEAD>
<FRAMESET cols="240, *">
<FRAME src="menu.html" marginheight="5" marginwidth="5"
noresize scrolling="auto">
<FRAME src="main.html" name="right" marginheight="15"
marginwidth="15" noresize scrolling="auto">
</FRAMESET>
</HTML>
```

Тепер, коли фрейму дане ім'я (*right*), в оголошення зв'язку повинне бути додане ім'я цільового фрейму. Внесемо наступні зміни у файл menu.html:

```
<HTML>
<HEAD>
<TITLE>Меню</TITLE>
</HEAD>
<BODY>
<P><A hef="about.html" target="right">про фірму</A>
<P><A href="clients.html" target="right">про клієнтів</A>
<P><A href="contact.html" target="right">про контакти</A>
</BODY>
</HTML>
```

Якщо ви хочете, щоб усі сторінки вузла завантажувалися в те саме вікно, тоді вам необхідно скористатись атрибутами *name* - *target*. У коді кожного документа, що ви будете завантажувати в це вікно, використовуйте дескриптор *<BASE>* у розділі *<HEAD>*: *<BASE target="right">*.

Спеціальні цільові імена

Існує кілька визначених цільових імен фреймів.

• *target="_blank"* — документ відкривається в новому вікні браузера.

• *target="_self"* — документ завантажується в те ж вікно, з якого він викликаний.

• *target="_parent"* — документ завантажується в батьківський набір фреймів.

• *target="_top"* — документ завантажується в повне вікно браузера, ігноруючи всі набори фреймів.

Пам’ятайте, що спеціальні цільові імена, по-перше, починаються із символу підкреслення, а по-друге, щоб уникнути непередбачених проблем беруться в лапки.

При використанні спеціальних цільових імен варто врахувати ряд обставин.

Необхідно уникати появи в звичайних іменах яких-небудь символів, крім алфавітно-цифрових. Так, підкреслення в іменах будуть зігноровані.

Значення *" blank"* завжди провокує відкриття нового вікна браузера. Використовуйте це значення тільки тоді, коли нове вікно дійсне необхідно. В противному випадку ви ризикуєте відлякати відвідувачів вузла появою численних ресурсномістких вікон браузерів.

Значення *"_top"* бажано застосовувати у випадках, коли посилання відправляє відвідувача на інший вузол.. Уникайте відкривати чужі сторінки за будь-яку ціну.

Приклад 1. Створення двох фреймів

```
<frameset cols="200,*">
<frame src=menu.html name=MENU>
<frame src=content.html name=CONTENT>
</frameset>
```

Вікно браузера розбивається на дві колонки, ліва займає 200 пікселов, а права - простір, що залишилося, був заданим символом зірочки. Ширину або висоту фреймів можна також задавати в процентному відношенні, на зразок таблиць.

В дискрипторі *<FRAME>* задається ім'я HTML-файла, котрий завантажується у вказану область за допомогою параметра src. В ліве вікно буде завантажений файл, названий menu.html, а в праве - content.html. Кожному фрейму бажано задати його унікальне ім'я, щоб документи можна було завантажувати в указане вікно.

Приклад 2. Створення трьох фреймів

```
<frameset rows="10%,90%">
<frame src="top.html" name="TOP">
<frameset cols="200,*">
<frame src="menu.html" name="MENU">
<frame src="content.html" name="CONTENT">
</frameset>
```

</frameset>

Ми розглянули тільки кілька способів кодування фреймів. Допускаються й інші варіації використання перерахованих атрибутів. Якщо вас зацікавили фрейми, експериментуйте з ними і переконаєтеся в їхніх величезних можливостях.

Фіксовані і динамічні фрейми

Для керування розмірами фреймів на сторінці можна використовувати динамічні фрейми і фрейми фіксованих розмірів. Приклад використання фіксованих фреймів приведений у лістингу 4.

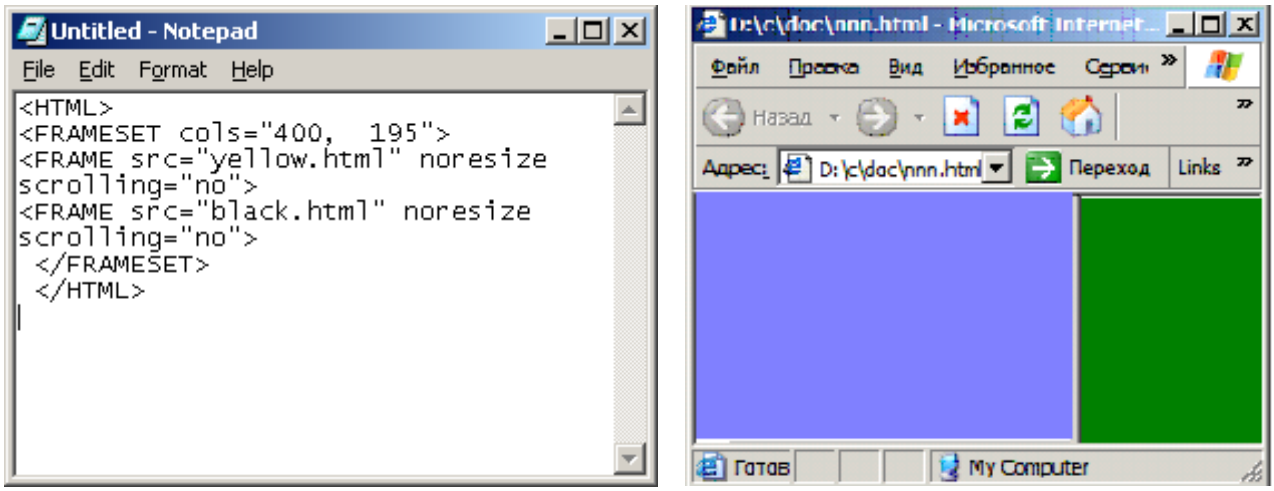


Рис. 4.19 Сторінка з фіксованими фреймами

Не забувайте, що ширина фреймів задається в атрибуті *cols*, а висота — в атрибуті *rows*.

Розміри фреймів, як і розміри таблиць, можна робити динамічними, що змінюються зі зміною розміру вікна браузера. Для оголошення динамічних фреймів використовуються значення розмірів у відсотках.

<FRAMESET rows="50%, 25%, 25%">
<FRAME src="red.html" noresize scrolling="no">
<FRAME src="black.html" noresize scrolling="no">
<FRAME src="yellow.html" noresize scrolling="no">
</FRAMESET>

Створюючи комбінацію з фіксованого і динамічного фреймів, як значення розміру можна використовувати символ *. Він означає, що броузер автоматично віддасть частину екрана, що залишилася після розміщення фіксованих фреймів, динамічним фреймам, наприклад:

*<FRAMESET rows="150, *">*
<FRAME src="red.html" noresize scrolling="no">
<FRAME src="black.html" noresize scrolling="no">
</FRAMESET>

Плаваючі фрейми

Уперше випробувані в Internet Explorer 3.0 фрейми, що плавають, (I-Frames від Inline Frames, вони ж - floating frames) офіційно прийняті стандартом HTML 4.0. Браузером Netscape Navigator 4.61 фрейми, що плавають, не підтримуються.

Фрейми, що плавають, відрізняються від звичайних. Вони не вимагають оголошення набору фреймів — замість цього інформація про фрейми, що плавають, розташовується безпосередньо в коді стандартної HTML-сторінки.

От фрагмент коду з оголошенням фрейму, що плаває:

<IFRAME width="350" height="200" src="text.html"> ... </IFRAME>

Код виглядає дуже схожим на запис дескриптора *<OBJECT>* чи *<IMG>* — з оголошенням ширини і висоти поля. В дескрипторі *<IFRAME>... </IFRAME>* атрибут *src=”x”*, де ”x” джерело ресурсу Фрейду. А „**...**” це текст, що записується в контейнерний дескриптор *<IFRAME>* він виводиться на екран, якщо ваш бровзер не підтримує фрейми. Дескриптору *<IFRAME>* властиві всі атрибути, що й для дескриптора *<FRAME>*.

Можна також вирівнювати текст, розташований у межах фреймів, що плавають:

<IFRAME width="350" height="200" src="text.html"
scrolling="no" frameborder="0" align="right" hspace="10" vspace="10">
<FRAME width="350" height="200" src="text.html"
scrolling="no" frameborder="0" align="right" hspace="10" vspace="10">
</IFRAME>

Плаваючі фрейми підтримують атрибут name, а також спеціальні імена.

Дескриптор <*NOFRAMES*>

Однієї з глобальних проблем для фрейми вузлів, що використовує, як уже згадувалося, є відсутність гарантій підтримки фреймів у браузерах користувачів.

З огляду на описані проблеми сторінок із фреймами, спробуємо організувати доступ для відвідувачів, що надають перевагу текстовим браузерам. Досягти цього у вузлі з фреймами можна, використовуючи дескриптор *<NOFRAMES>*. Він розміщується в сторінці з набором фреймів, і між відкриваючим і закриваючим дескрипторами міститься вся інформація, що буде виведена браузером, не підтримуючі фрейми. Браузер з підтримкою фреймів проігнорує все, що знаходиться між дескрипторами *<NOFRAMES>* і *</NOFRAMES>*.

Створення форм засобами HTML

Форма (form) – перевірений часом гнучкий метод взаємодії користувачів з Web-вузлом. Це один з видів інтерфейсу взаємодії з аудиторією. На відміну від статичних методів HTML, форма зв’язує не тільки користувачів і сторінки вузла, але і сценарії, що виконуються на WEB-сервері.

Відношення між формою і програмою, що обробляє її дані регулюються стандартом CGI (*Common Gateway Interface* – інтерфейс спільного доступу). CGI працює в якості каналу, по котрому передається інформація із форми процесу, що її обробляє. Форми можуть виконувати широкий ряд задач – від звичайного отримання даних про користувачів вузла (його ім'я, адресу та ін.) до створення ігр.

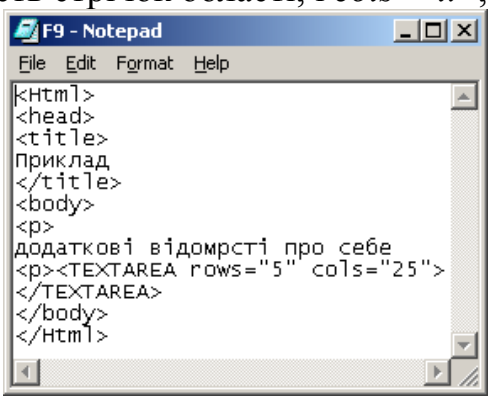
В будь-якій формі існують дескриптори і деякі спеціальні елементи, які називаються ***елементами управління***. Існує декілька ключових елементів форм і ряд пов'язаних з ними атрибутів котрі необхідно знати для створення форми.

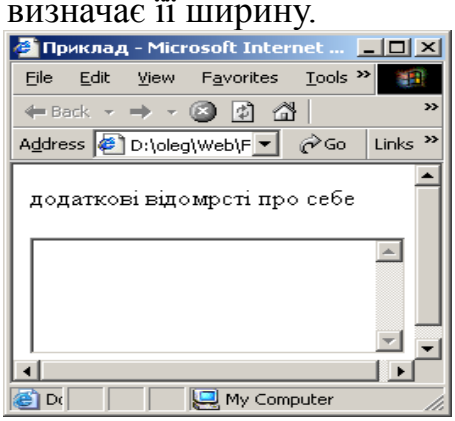
<FORM>...</FORM> - це фундаментальний елемент всіх форм, дескриптори, відкриваючі і закриваючі форму. Форма може мати декілька атрибутів. Основні з них – action, котрий в якості значення приймає URL для передачі на вузол інформації з форми, і *metod*, що приймає значення *get* або *post*. Ці значення визначають спосіб передачі даних форми вузлу.

<INPUT> - це дескриптор відповідає за створення елементів управління, розміщених у формі. Це дескриптор одиночного типу. З цим дескриптором можуть використовуватись наступні атрибути:

- *type="x"* – визначає елемент управління;
- *name="x"* – ім'я елемента управління;
- *value="x"* – значення елемента управління; його атрибут є необов'язковим для всіх елементів управління, крім перемикача (*radio*);
- *size="x"* – ширина поля елемента управління в пікселях; для елементів *text* і *password* ширину поля визначають в символах;
- *maxlength="x"* – максимальна кількість символів, котрі приймає елемент управління;
- *checked="x"* – відмічений прапорець (перемикач);
- *src="x"* – вказівник на рисунок, що використовується в формі в якості графічної кнопки.

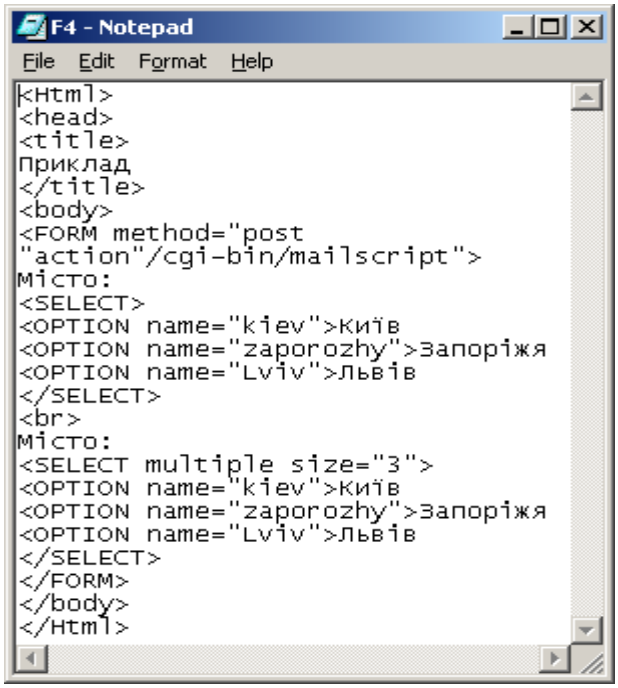
<TEXTAREA>...</TEXTAREA> - цей елемент створює текстову область з розширеними можливостями. Він схожий до елементів управління, але, крім атрибутів, ми можемо управляти розміщений між дескрипторами текст. Крім вже відомого атрибута *name* приймає атрибути *rows="x"*, де *"x"* визначає кількість стрічок області, і *cols="x"*, де *"x"* визначає її ширину.

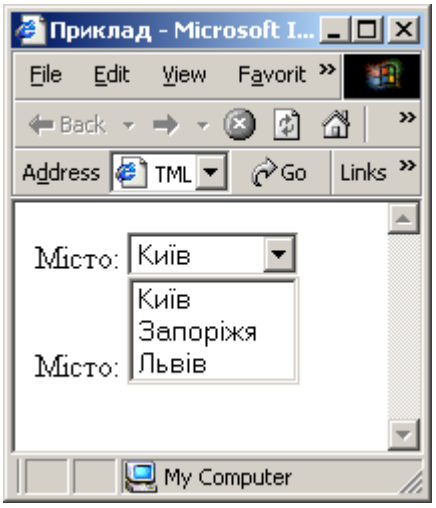




<SELECT>...</SELECT> - це елемент створює список. В залежності від

атрибутів, що використовують атрибути, список може бути випадаючим і представлений у вигляді списку-меню. Якщо потрібно представити користувачу вузла можливість вибору декількох елементів, використовують атрибут *multiple*. Перший спосіб можна використати як звичайне випадаюче меню яке представлено на рисунку нижче. Другий спосіб представляє собою список видимих елементів який програмується атрибутом *multiple*.

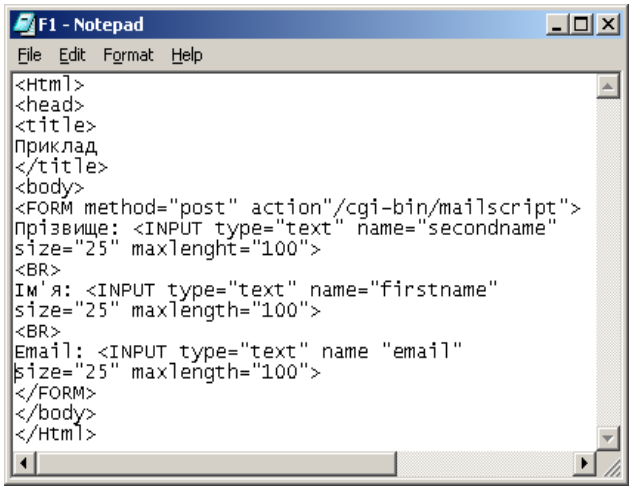


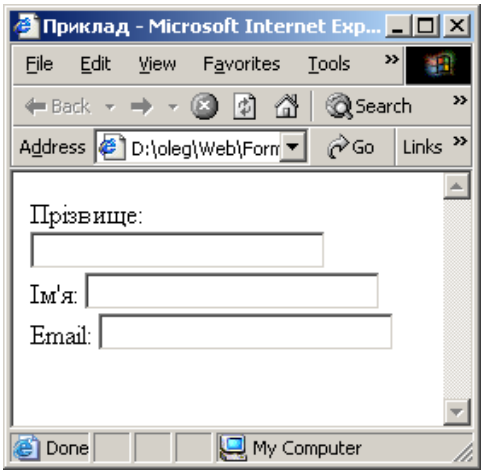


<OPTION>…</OPTION>- цей дескриптор визначає кожний елемент списку-меню окремо. Використовуються два дескриптора - відкриваючий і закриваючий. Встановлена кількість стрічок і стовпців в текстовій області не обмежує гранично допустиму довжину стрічки і текст в цілому. Наприклад, при ширині поля 40 символів і висоті в 20 символів в нього можна ввести 800 символів. Якщо ми вийшли за даний розмір 800 символів в нас з’являються полоси прокрутки.

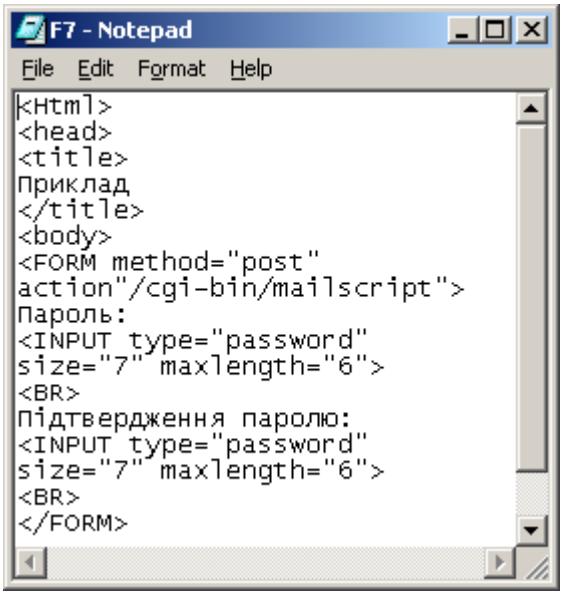
У формах існують управляючі елементи, що визначають спосіб вводу інформації. Синтаксичні елементи представляють собою значення атрибуту *type* дескриптора *<INPUT>*.

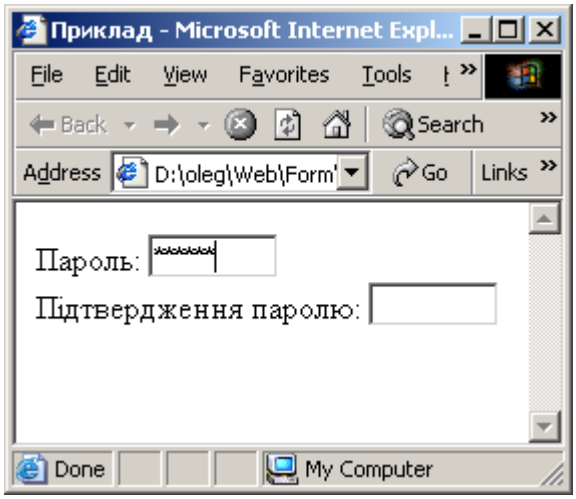
text - поле вводу тексту, складаються з однієї стрічки. Ширина поля визначає значення атрибута *size*;



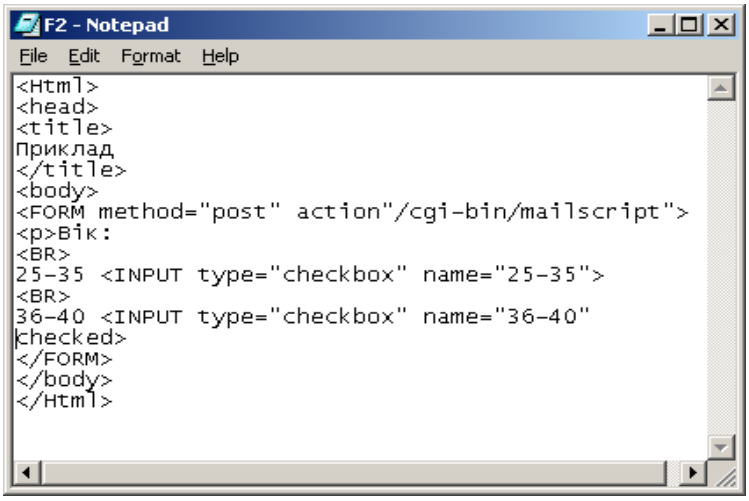


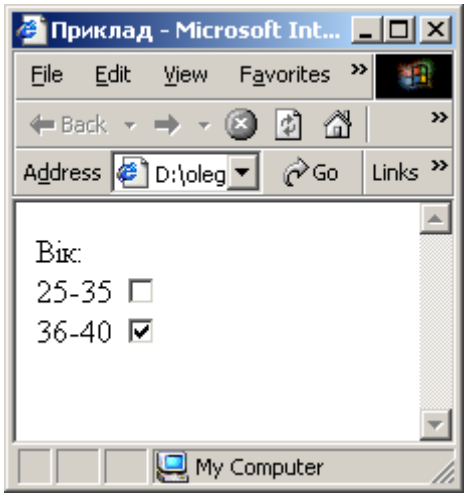
password - поле, подібне до поля вводу тексту *text*. Різниця полягає в тому, що символи, що вводяться користувачем вузла відображаються в вигляді символів *;



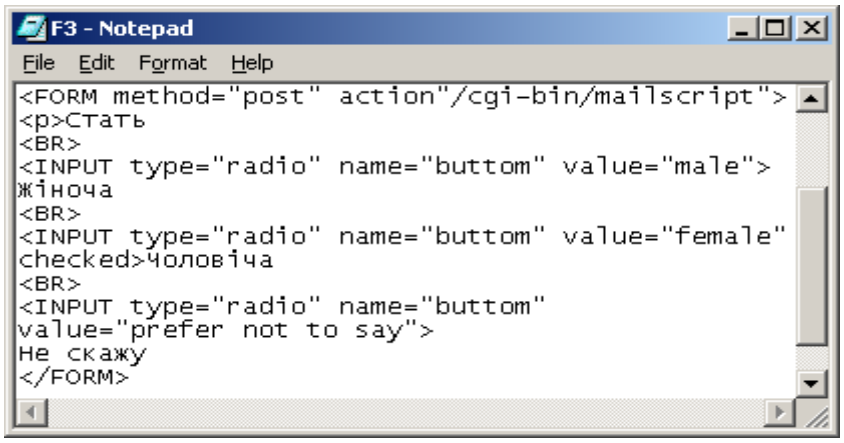


checkbox – прапорець, що приймає два логічних значення. Можна створити групу з декількох прапорців, всі вони при цьому будуть відмічені;





radio – перемикач. На відміну від прапорця, форма складається з декількох перемикачів, що утворюють групу, може бути вибраний тільки один;



Button – кнопка. Для роботи цього елементу необхідно зв'язати її з сценарієм. Надпис на кнопці визначається встановленням її значення *value*;

submit – кнопка, натиск котрої визначають підтвердження вибору і відправки інформації на WEB-вузол. Надпис на кнопці визначається встановленням її значення *value*;

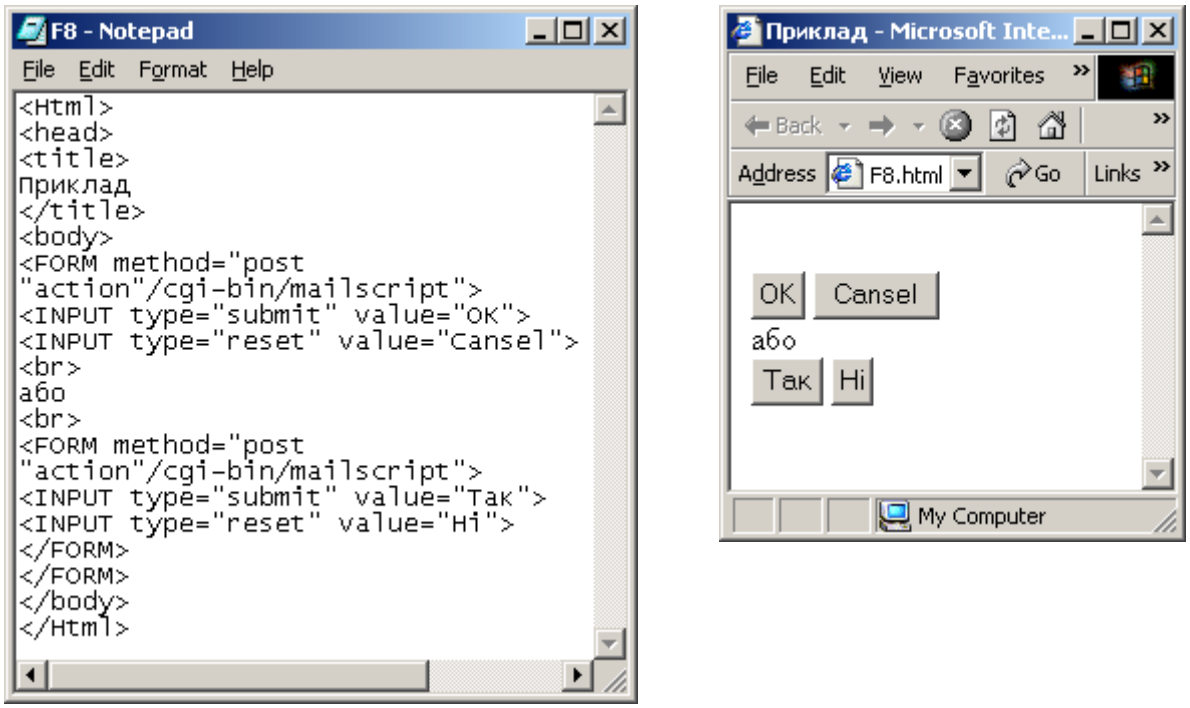
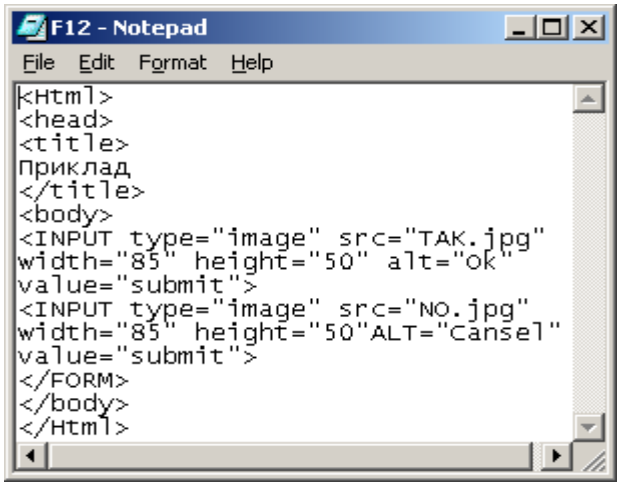


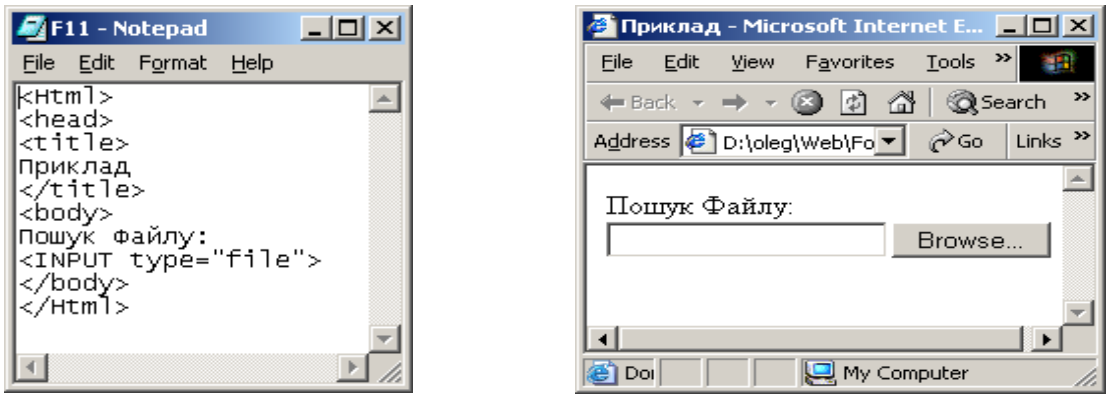
image – елемент, котрий дозволяє вставити зображення і використовувати його для підтвердження вибору або скиду кнопок *submit* і *reset*.

Побудову форми можна починати в стандартній сторінці HTML в тілі документу дескриптором *<FORM>* і відповідно закриваючий дескриптор *</FORM>*.



reset – кнопка, натиск котрої означає скидання поточних значень елементів управління і встановлення значення по замовчуванню;

file – дозволяє створювати кнопку виклику діалогового вікна вибору файлу;



hidden – прихований елемент управління, котрий не представлений будь-яким зображенням на формі. Переважно сховані елементи використовують з метою автоматичного вводу додаткових даних для передачі WEB-вузлу;

fieldset – цей елемент управління дозволяє групувати логічно зв'язані елементи управління і надписів. Він здійснює навколо групи елементів управління рамку забезпечує перехід фокуса вводу в межах групи за допомогою атрибута *tabindex*. Застосування цього елемента управління підвищує доступність форми різним користувацьким пристроям;

legend – з допомогою елемента *legent* автор може зробити заголовок елемента управління.

5. ТЕХНОЛОГІЯ CSS ТА ЇЇ ПІДТРИМКА БРАУЗЕРАМИ

Ви вже розглянули можливості форматування вмісту веб-сторінок за допомогою засобів мови HTML та візуального редактора веб-сай-тів. Проте є й інші засоби оформлення текстів і створення зовнішніх ефектів, які можуть прикрасити веб-сторінку. Одним з них є таблиці стилів. Загальний принцип використання стилів на веб-сторінках той самий, що і для документів, створених у середовищі текстового процесора: користувач визначає набори правил форматування, які потім застосовуються до елементів документа. Проте у веб-дизайні способи застосування стилів різноманітніші.

Поняття про таблиці каскадних стилів

Стиль — це набір правил оформлення та форматування, який можна застосувати до різних елементів веб-документа. У разі використання стандартної мови HTML для надання кільком елементам певних властивостей (наприклад, призначення стилю шрифту) доводиться задавати ці властивості для кожного елемента. Використання таблиць стилів дає змогу уникнути цього, оптимізувавши розробку веб-сайтів. Стилі дають змогу позиціювати елементи сторінки (наприклад, тексти і графіку), задаючи координати. Крім того, таблиці стилів часто використовують під час створення так званих динамічних сторінок.

Таблиці каскадних стилів (CSS, Cascading Style Sheets) містять параметри форматування частини або всього тексту веб-сторінки. Якщо таку таблицю підключено, то у тегах можна просто вказувати посилання на неї, а не задавати велику кількість атрибутів. У цьому випадку стилі названі каскадними тому, що в одному документі їх можна описати кілька, і браузер використовуватиме їх каскадом відповідно до їхнього пріоритету.

Таблиці каскадних стилів — це передусім набори параметрів, що змінюють властивості тегів HTML. Такі набори називають ще визначеннями тегів. Наприклад:

```
P {font-size: 4Opt; color: green; font-family: "Comic Sans MS"}
```

Тут задано параметри для тегу абзацу <P>, які встановлюють розмір шрифту 40 пунктів, колір шрифту — зелений, гарнітуру — Comic Sans MS. У документі достатньо ввести теги <P>...</P> із текстом абзацу, щоб автоматично надати йому зазначеного оформлення.

Таблиці каскадних стилів дають змогу отримати результати, яких неможливо досягнути звичайними засобами HTML. У наведеному прикладі встановлено розмір шрифту 40 пунктів, хоча в мові HTML за допомогою атрибута SIZE можна задати для шрифту максимальний розмір 7, що відповідає 36 пунктам.

Крім того, таблиці каскадних стилів дають змогу визначити єдиний стиль

оформлення для різних сторінок документа і швидко модифікувати його зміною відповідного параметра у таблиці стилів.
Параметрів форматування, які можна задавати за допомогою стилів досить багато:

- background - колір тла;
- font- - family• -стиль шрифту (гарнітура);
- font- -size -розмір шрифту;
- font- -weight - жирність шрифту;
- color - колір шрифту;
- text- -decoration - оздоблення тексту;
- text- -align - вирівнювання тексту;
- margin-top - відступ від верхнього рядка абзацу;
- line- -height - міжрядкова відстань.

Застосування каскадних стилів у HTML-документах

Є три способи зв'язку каскадних стилів із HTML-документом: підключення зовнішньої таблиці стилів; розташування опису стилів у розділі HEAD документа; задавання властивостей стилів безпосередньо в тегах абзаців чи заголовків.

Підключення зовнішньої таблиці стилів

Зовнішня таблиця стилів (External Style Sheet) — це текстовий файл із розширенням .ess. Його підключають до HTML-документа за допомогою тегу <LINK>, який записують у розділі <HEAD>, наприклад:
<LINK REL="stylesheet" TYPE="text/ess" HREF="mystyle.css">
Атрибути REL та TYPE вказують браузеру на те, що сторінка використовує таблиці каскадних стилів. Атрибут HREF задає адресу файлу (mystyle.css),

Оформимо веб-сторінку за допомогою зовнішньої таблиці стилів.
1. Відкрийте текстовий редактор Блокнот і введіть таке визначення тегу <P>:

P {font-size: 40pt; color: green; font-family: "Comic Sans MS"}

Збережіть файл з іменем mystyle.css.
Створіть у Блокноті файл такого змісту:

```
<HTML>
<HEAD>
<LINK REL="stylesheet" TYPE="text/ess"
HREF="mystyle.css">
<TITLE>Приклад використання CSS</TITLE>
</HEAD>
<BODY>
<P>Цей текст оформлено відповідно до таблиці стилів, яка
```

міститься у файлі mystyle.css: розмір шрифту — 40 пунктів,
колір шрифту — зелений, а гарнітура — Comic Sans MS.</P>
</B0DY>
</HTML>

Збережіть цей файл з іменем pryklad1.html у тій самій папці, що і mystyle.css.

Відкрийте файл pryklad1.html у вікні браузера (рис. 5.1). Як бачите, хоча в цьому документі не задано жодних параметрів для тегу <P>...</P>, текст абзацу набув нових властивостей, які описані в зовнішній таблиці стилів mystyle.css.

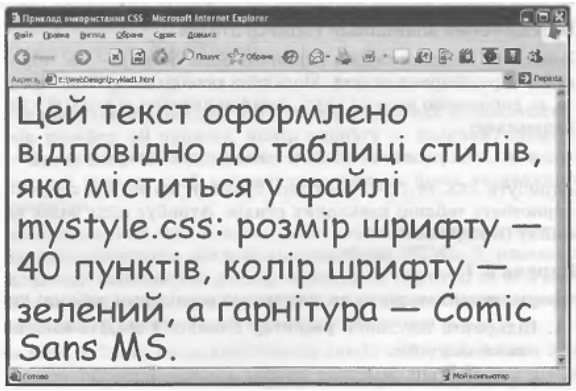


Рис. 5.1 Результат застосування таблиці каскадних стилів

Перевага такого методу підключення CSS полягає у тому, що одну таблицю стилів можна підключати до багатьох документів, і всі вони матимуть єдине стильове оформлення.

Використання стилів внутрішньої таблиці

Внутрішню таблицю стилів (Embedded Style Sheet) розміщують безпосередньо в розділі HEAD, у блоці, який обмежений тегами
<STYLE> та </STYLE>.

```
<HEAD>
<STYLE>
Тег1 {властивістьі: значенняіі; властивість 12: значенняі2; ...;
властивістьіп: значенняіп}
Тег2 {властивість21: значення21; властивість22: значення22; ...;
властивість2ш: значення2т}
…
</STYLE> </HEAD>
```

Як видно з коду, найпростіша внутрішня таблиця стилів — це послідовність визначень тегів, кожне з яких записується, як правило, з нового рядка. Визначення тегу містить його ім'я без кутових дужок, за яким у фігурних дужках через крапку з комою перелічують властивості тегів та їхні значення, розділені двокрапками. Розглянемо приклад.

Визначимо стилі заголовків першого та другого рівнів із використанням

тегу <STYLE>.

```
<HTML>
<HEAD>
<STYLE>
HI {font-size: 48pt; color: red}
H2 {font-size: 20pt; color: blue}
</STYLE>
<TITLE>Приклад використання CSS</TITLE>
</HEAD>
<BODY>

<H1>Для заголовка першого рівня визначено розмір 48 pt,
а колір тексту — червоний </H1>
<H2>Для заголовка другого рівня визначено розмір 20 pt,
а колір тексту — синій </H2>
<P>Для цього абзацу стиль не застосовано, для оформлення
тексту використано атрибути за умовчанням.
</BODY>
</HTML>
```

У цьому описі стилю змінено оформлення заголовків HI та H2. Зокрема, для першого призначено розмір 48 пунктів і червоний колір, а для другого — розмір 20 пунктів і синій колір. У вікні браузера цей документ матиме вигляд, як на рис. 5.2.

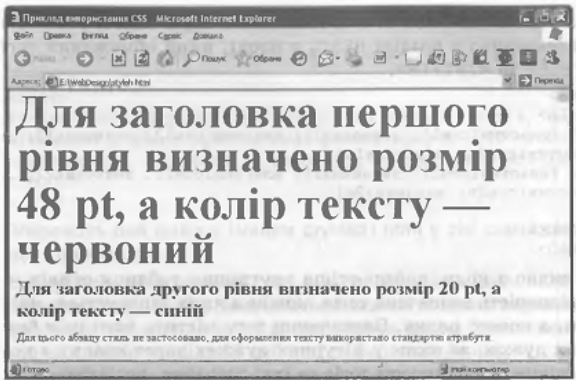


Рис. 5.2. Стилі заголовків першого та другого рівнів

У таблицях можна створювати нові стилі, надаючи їм імена з крапкою перед першим символом. У тегах звертаються до такого стилю за іменем, використовуючи атрибут CLASS=ім’я_стилю, де ім'я записують вже без крапки. Для одного тегу можна використовувати декілька стилів форматування.

Наприклад, застосовуючи атрибут CLASS тегу </P>, задамо для двох абзаців різні стилі: першому надамо розмір шрифту 38 пунктів, білий колір символів і оливковий колір тла, другому — розмір шрифту 46 пунктів, фіолетовий колір символів і рожевий колір тла.

```
<HTML>
<HEAD>
<STYLE>
.stylel {font-size: 38pt; color: white; background-color: olive}
.style2 {font-size: 46pt; color: magenta; background-color: mistyrose}
</STYLE>

<TITLE>Приклад використання CSS</TITLE>
</HEAD>
<BODY>
<P CLASS-stylel>До цього абзацу застосовано стиль stylel </P>
<P CLASS=style2>До цього абзацу застосовано стиль style2 </P>
</BODY>
</HTML>
```

Який вигляд має цей документ у вікні браузера, показано на рис. 5.3.

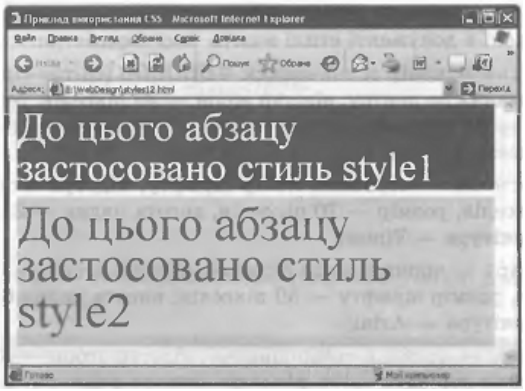


Рис. 5.3. Застосування різних стилів форматування для тегу абзацу

Стилі дають змогу сформувати сторінку з цікавими ефектами, які інакше можна створити лише за допомогою графіки. Це, наприклад, ефект об'ємного тексту або накладання (часткового перекриття) фрагментів сторінки.

Створення текстових ефектів за допомогою стилів

Визначимо для тегу <BODY> такий стиль: шрифт Arial чорного кольору розміром 16 пунктів. Внаслідок цього весь текст, що міститься між тегами <BODY> та </BODY>, буде автоматично відформатований відповідно до цих властивостей. Тепер визначимо стилі з іменами тінь, основа, шарі і шар2. Текст запишемо в шарах, які накладаються. Спочатку відображатиметься шар тінь, а на нього накладатимуться шари основа, шарі та шар2. Порядок накладання задано розташуванням фрагментів тексту в HTML-документі.

У прикладі буде застосовано такі властивості:

margin-top — відступ згори (за від'ємних значень можна забезпечити накладання фрагментів);

color — колір;

font-size — розмір шрифту;
font-family — сімейство шрифтів (гарнітура);
line-height — висота рядка.

Визначені в документі стилі мають такі параметри:
.тінь — колір зі значенням #DBDBDB (світло-сірий), вирівнювання по центру, відступ згори — ЗО пікселів, розмір шрифту — 80 пікселів, висота рядка — 270 пікселів, гарнітура — Times;
.основа — червоний колір шрифту, відступ згори — -230 пікселів, розмір — 70 пікселів, висота рядка — 250 пікселів, гарнітура — Times;
.шарі — чорний колір шрифту, відступ згори — -100 пікселів, розмір шрифту — 50 пікселів, висота рядка 65 пікселів, гарнітура — Arial;
.шар2 — зелений колір шрифту, відступ згори — ЗО пікселів, розмір шрифту — 35 пікселів, висота рядка — 45 пікселів, гарнітура — Arial.

Для виділення частини HTML-документа використовують тег <DIV>. Він нічого не форматує, а лише відзначає фрагмент тексту, який виступає як окремий об'єкт. Атрибут CLASS цього тегу дає змогу посилатися на стилі внутрішньої таблиці й тим самим задавати стиль подання тексту, розташованого між тегами <DIV CLASS...> та </DIV>.

```
<HTML>
<HEAD>
<TITLE>Приклад використання CSS</TITLE>
<STYLE>
BODY {color: black; font-size: 16px; font-family: Arial}
.тінь {color: #DBDBDB; text-align: center;; margin-top: 30px;
font-size: 80px; line-height: 270px; font-family: Times}
.основа {color: red; margin-top: -230px; font-size: 70px;
line-height: 250px; font-family: Times}
.шарі {color: black; margin-top: -lOOpx; medium; font-size:
50px; line-height: 65px; font-family: Arial}
.шар2 {color: green; margin-top: ЗОpx; font-size: 35px;
line-height: 45px; font-family: Arial}
</STYLE>
</HEAD>
<BODY>
```

Приклад використання каскадних стилів

```
<CENTER>
<TABLE WIDTH=500 CELLPADING=0 CELLSPASING=0 BORDER=0>
<TR>
<TD ALIGN=CENTER VALIGN=TOP>
<DIV CLASS=TiHb>TeKCT із t:lhhk></DIV>
<DIV CLASS-oCHOBa>TeKCT із t:Lhhk></DIV>
<DIV CLASS=uiapl>Приклад тексту із to.hhio</DIV>
<DIV CLASS=uiap2>IJ,e приклад використання каскадних стилів</OIУ>
```

```
</TD> </TR>
</TABLE>
</CENTER>
</BODY>
</HTML>
```

Вигляд цього документа після відкриття його у вікні браузера, показано на рис. 5.4.

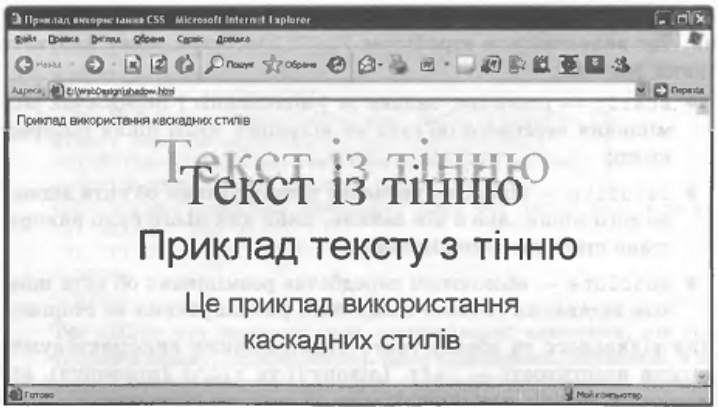


Рис. 5.4. Приклад накладання рядків тексту

Вбудовані стилі

Вбудовані стилі (Inline Styles) вставляють у теги заголовків <H1>... <H6>, абзацу <P>, тіла <BODY>, а також у теги <DIV>, <IMG> тощо за допомогою атрибута STYLE, в якому перелічують властивості та їх значення. Наприклад:
<P STYLE="font-size: 48pt; color: yellow">

Визначені у такий спосіб властивості мають найвищий пріоритет порівняно з іншими, оскільки вони визначені безпосередньо у те-гу. Цей підхід використовують для оформлення невеликої кількості елементів.
Приклад використання атрибута STYLE для форматування заголовка другого рівня:

<H2 STYLE-"font-size: 48pt; font-family: Aria1">Текст...</H2>

Позиціювання елементів веб-сторінки за допомогою стилів

Розміщувати елементи на сторінці можна не лише за допомогою фреймів і таблиць — каскадні стилі надають для цього додаткові цікаві можливості.

Зокрема, можна обирати один із трьох типів позиціювання елементів на сторінці: статичне, відносне та абсолютне позиціювання. Тег визначається атрибутом POSITION, який може набувати таких значень:

static — статичне; задане за умовчанням і передбачає розміщення чергового об'єкта на вільному місці після попереднього;

relative — відносне; визначає розташування об'єкта відносно того місця, яке б він зайняв, якби для нього було використане статичне позиціювання;

absolute — абсолютне; передбачає розміщення об'єкта шляхом задавання точного місця його розташування на сторінці.

Для відносного та абсолютного позиціювання використовують чотири властивості — left (ліворуч) та right (праворуч), які задають відступи для розміщення об'єкта по горизонталі, і top (згори) та bottom (знизу) — по вертикалі. Необхідно задавати по одному значенню позиції по горизонталі та вертикалі. При цьому елементи можуть накладатися один на інший: нижче буде розташований об'єкт, описаний першим, а зверху — той, який описали останнім.

Якщо потрібна зміна стандартного порядку накладання, застосовують шари. Шар об'єкта задають за допомогою властивості z-index, значенням якої можуть бути лише ціле число та auto (за умовчанням). Об'єкт із більшим значенням z-index розміститься поверх об'єкта з меншим значенням цієї властивості, перекриваючи його. Якщо ж об'єкти матимуть однакові її значення, то зверху розташується об'єкт, описаний нижче за текстом у HTML-документі.

Розглянемо, як позиціювати три об'єкти-зображення і два фрагменти тексту. Координати спеціально вибрані так, щоб об'єкти перекривалися. Зауважте, що слід використовувати атрибут STYLE, а не тег <STYLE>.

1. У програмі Блокнот введіть такий текст веб-сторінки:

```
<HTML>
<HEAD>
<TITLE>Позиціювання</TITLE>
</HEAD>
<BODY>
<DIV STYLE="position: absolute; top: 0; left: 70; width: 50; height: 100">
<IMG SRC="fly. jpg"></DIV>
<DIV STYLE="position: absolute; top: 10; left: 15; width: 600; height: 100">
<H1 STYLE="color: ye11ою">Повій, вітре, до схід сонця, до схід сонця, край віконця.</H1></DIV>
<DIV STYLE="position: absolute; top: 60; left: 400; width:50; height: 100">
<H1 STYLE="color: blue">Чайка</H1></DIV>
</BODY>
</HTML>
```

Тег <DIV> тут виконує роль контейнера: елементи, що містяться в ньому, успадковують його властивості, зазначені в атрибуті STYLE. Перший контейнер містить зображення, тип розташування якого є абсолютним, з нульовим відступом згори, 70 пікселів — відступ зліва; його ширина становить 50, висота — 100 пікселів. Другий контейнер містить текст «Повій, вітре, до схід сонця, до схід сонця, край віконця», він зміщений на 10 пікселів від початку сторінки вниз та на 15 вліво, ширина контейнера — 600, висота — 100 пікселів; завдяки використанню

тегу H1 із атрибутом STYLE текст має відповідний до заголовка першого рівня розмір та жовтий колір. Третій контейнер зміщений на 60 пікселів униз та 400 ліворуч, ширина контейнера — 50, висота — 100 пікселів. Слово «Чайка» оформлене як заголовок першого рівня синього кольору.

2. Збережіть цей документ у файлі з іменем pryklad2.html. Скопіюйте у папку з цим документом зображення fly.jpg (можна використати інший файл, але його назва має збігатися з назвою в HTML-документі).

3. Відкрийте файл pryklad2.html у браузері (рис. 5.5) — контейнери перекриваються згідно з розташуванням у документі.

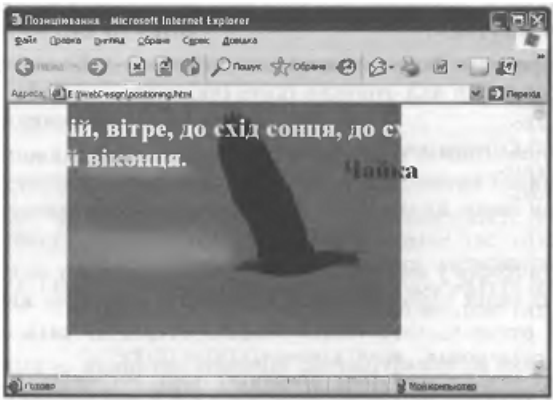


Рис. 5.5. Позиціювання фрагментів тексту і зображення за допомогою стилів

4. Внесіть зміни у текст веб-сторінки, який стосується першого контейнера (із зображенням). Для цього клацніть правою кнопкою миші вільне місце сторінки у вікні браузера і виберіть пункт Перегляд HTML-коду (View Source). У вікні редактора, що з'явиться, внесіть такі зміни:

<DIV STYLE="position: absolute; top: 0; left: 70; width: 50; height: 100; z-index: 2"> <IMG SRC="fly. jpg">< /DIV>

Решту документа залиште без змін. Хоча в тексті HTML-документа зображення описане першим, завдяки тому, що йому присвоєно індекс із більшим номером, воно відобразиться поверх інших об'єктів.

5. Збережіть внесені зміни (командою Файл ► Зберегти), закрийте текстовий редактор і оновіть веб-сторінку за допомогою команди Вигляд ► Оновити або функціональної клавіші F5. Вигляд цього документа у вікні браузера показаний на рис. 5.6.

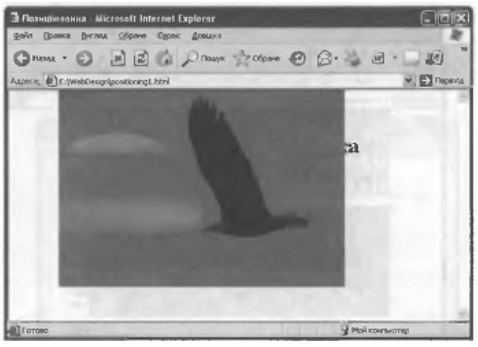


Рис. 5.6. Використання властивості z-index

Отже, використання властивості z-index дає змогу відображати об'єкти незалежно від порядку їх розташування в тексті HTML-документа.

6. Під час відображення сторінки може статися, що розміри елемента перевищують розміри наданого йому місця. Наприклад, текст і малюнок не вміщуються у виділений для них прямокутник. У таких випадках використовують властивість overflow (переповнення). Вона може мати три значення:

- none — якщо елемент вийде за межі наданого місця, він все одно буде показаний;
- clip — об'єкти, що виходять за межі, будуть обрізані;
- scroll — буде використано прокручування.

Відкрийте HTML-код сторінки і внесіть у другий контейнер, що містить текст «Повій, вітре, до схід сонця, до схід сонця, край віконця», такі зміни:

<DIV STYLE="position: absolute; top: 10; left: 15; width: 220; height: 120; overflow: scroll">

<H1 STYLE="color: ye11cж,,>Повій, вітре, до схід сонця, до схід сонця, край віконця </Hl></DIV>

7. Збережіть зміни та оновіть веб-сторінку в браузері. Який вигляд вона матиме, показано на рис. 5.7.

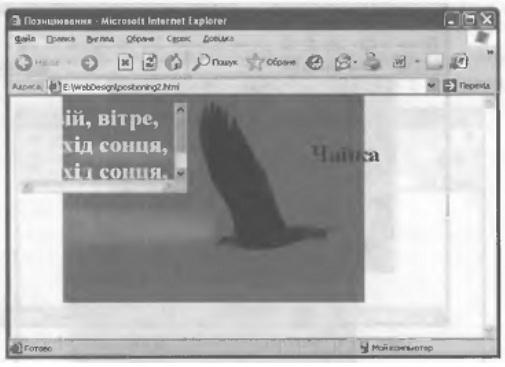


Рис. 5.7. Використання властивості overflow для реалізації прокручування

Поняття об'єктної моделі

З появою таблиць каскадних стилів у HTML з'явилася можливість будувати логічну структуру документа, а потім визначати формат її відображення. Цей підхід змінив усю технологію проектування сторінок сайту. Тепер можна визначити спочатку типи сторінок, потім логічні структури сторінок для кожного типу і, нарешті, для кожного логічного елемента, його склад і зовнішній вигляд.

Розглянемо поняття об'єктної моделі як способу взаємодії між HTML-кодом веб-сторінки та браузером. Об'єктна модель документа (Document Object Model, DOM) — це засіб для роботи зі структурою документа, а також з елементами сторінки в кодах HTML та у сценаріях. Вона забезпечує реалізацію технології динамічної HTML, яка ґрунтується на класичній HTML і використовує таблиці каскадних стилів та мови сценаріїв. Об'єктна модель документа є основою для того, щоб зробити елементи сторінки динамічно керованими під час її відтворення у вікні браузера.

Об'єктна модель описує кожний HTML-документ як набір окремих об'єктів — зображень, абзаців, списків і т. д. до найнижчого рівня, навіть до окремих символів. Кожний об'єкт може мати властивості, визначені у вигляді атрибутів. Наприклад, абзац <P> має атрибут вирівнювання <ALIGN>, який може набувати значень left, right або center. В об'єктній моделі атрибут називають властивістю об'єкта. Об'єкт має також свої методи і події, які можуть відбуватися з ним і впливати на нього. Наприклад, зображення <IMG> має подію OnMouseOver, яка відбувається тоді, коли користувач розміщує над ним вказівник миші. Можна керувати станом об'єктів, використовуючи методи з деякого набору стандартних методів. Все це й складає концепцію DOM як платформо-незалежного програмного інтерфейсу, який дає змогу програмам та скриптам керувати вмістом HTML-документів, змінювати їх структуру та оформлення.

Ми розглянули об'єкти HTML-документа, зокрема теги з текстовим наповненням. Проте браузер як програма також має свою об'єктну модель, при цьому моделі різних браузерів суттєво відрізняються. Браузер і документи, завантажені в нього, створюють ієрархічно організований набір об'єктів.

Сьогодні є можливість керувати як вмістом HTML-доку ментів, так і браузером. Наприклад, для браузера Internet Explorer за допомогою об'єктно-орієнтованих мов JScript та VBScript можна писати програми, які називають сценаріями (скриптами), і вставляти їх у HTML-код. Такі сценарії розміщують у спеціальних те-гах <SCRIPT> і </SCRIPT>.

Властивості, методи та події

Об'єкти мають фіксовані імена і певні властивості. Наприклад, вікну браузера відповідає об'єкт Window, а HTML-документу, завантаженому в браузер, — об'єкт Document. Звичайні властивості — це змінні з фіксованими

іменами, які мають певні значення. Одні властивості можна лише переглядати, інші можна змінювати. Для доступу до властивості об'єкта у мовах сценаріїв використовують такий синтаксис:

об'єкт.властивість

Наприклад, значенням властивості Document. Location є URL-адреса HTML-документа.

Властивістю об'єкта може бути інший об'єкт. При цьому перший об'єкт називають також батьківським (parent), а другий — нащадком (child). Якщо ми хочемо звернутися до властивості або методу об'єкта Object2, який міститься в об'єкті Objectl, то слід записати:

Objectl.0bject2.властивість
0bjectl.0bject2 .метод ()

Наприклад, об'єкт Document є нащадком об'єкта Window. Якщо ми хочемо щось записати в документ, завантажений у поточне вікно, то можемо скористатися для цього методом Write (). Наприклад:
Window.Document.Write ("Текст")

Методи — це пов'язані з об'єктами дії, які мають фіксовані імена, можуть мати параметри і повертати значення. Синтаксис застосування методу такий:

об'єкт.метод (список_параметрів)

Наприклад, метод Window. Open ("www. protvaryn. narod. ru") відкриває нове вікно браузера і завантажує у нього сторінку, розташовану за вказаною адресою.

Крім властивостей і методів для кожного об'єкта існує набір подій. Події мають наперед визначені назви: натискання кнопки миші позначають OnClick, відпускання кнопки миші — OnMouseUp, а завантаження документа в браузер — OnLoad тощо. Настанням події можна скористатися для ініціювання певних дій.

Окрім об'єктів, у модель входять колекції — структури, що складаються з однотипних елементів. Колекція — це групування об'єктів для спрощеного доступу до них за допомогою програмного коду. Наприклад, об'єкт Document містить колекцію зображень Images. Ми можемо звернутися до зображення або за його іменем, або за порядковим номером:

Document.Images ("my_image")
Document.Images (0)

Індекс (порядковий номер) елемента в колекції залежить від його розміщення у тексті HTML-документа. У колекціях об'єктної моделі перший елемент має нульовий індекс.

Колекціями також є:

All — всі теги та елементи, що розташовані на веб-сторінці;
Frames — фрейми;
Images — зображення;
Links — посилання та карти посилань.

Приклад 5.2. Об'єктна модель HTML-документа Розглянемо приклад HTML-документа.

```
<HTML>
<TITLE>Приклад</TITLE>
<BODY>
<P><IMG SRC="fly. jpg"  NAME="image1">
<A HREF="l.html"  NAME="link1">Чайка</AX/P>
<P> <IMG SRC="ptahy.jpg"  NAME="image2">
<A HREF="2.html"  HAME="link2">Фламінго</A></P>
</BODY>
</HTML>
```

На рис. 5.13 показано, який вигляд має цей документ у вікні браузера.

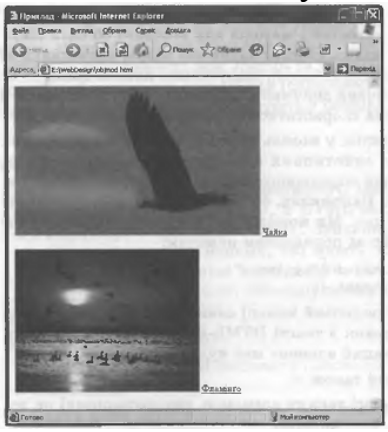


Рис. 5.13. Приклад веб-сторінки

На цій сторінці розташовано два малюнки та два посилання. На рис. 5.14 наведена ієрархічна схема об'єктного подання HTML-документа з цього прикладу

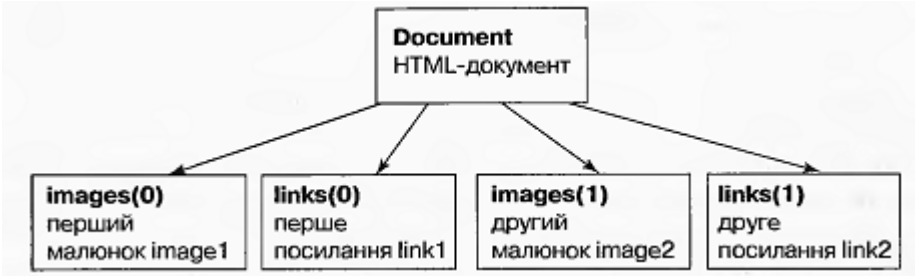


Рис. 5.14. Ієрархічна схема об'єктного подання HTML-документа

Головний об'єкт у цій структурі — Document. Перший елемент HTML-документа — зображення з іменем Imagel (це малюнок із файлу fly. jpg) — воно

подане як об'єкт колекції Images (0) (нумерація елементів колекції починається з нуля, тому перший елемент має індекс 0). Другий елемент — посилання з іменем Linkl на файл 1. html — нульовий об'єкт колекції Links тощо.

У HTML-документі імена елементів було визначено за допомогою атрибута NAME, і, хоча це не обов'язково, у нашому прикладі таке позначення виявилося корисним, оскільки тепер до елементів можна звертатися за іменами. Наприклад, для доступу до першого елемента сторінки можна записати Document. Images (0) або Document. Imagel. Це особливо зручно тоді, коли на сторінці розташовано багато елементів, і важко відстежити, який індекс має той чи інший із них.

Об'єктна модель браузера Internet Explorer

Основні об'єкти моделі браузера Internet Explorer такі:

- Window — об'єкт найвищого рівня, це вікно браузера Internet Explorer;
- Frame — фрейм; усі фрейми утворюють колекцію Frames;
- History — об'єкт-історія, призначений для навігації за списком переглянутих у цьому сеансі сторінок;
- Location — об'єкт-розташування, це URL-адреса поточної
- сторінки;
- Event — об'єкт-подія, надає інформацію, що пов'язана з деякою подією у сценарії;
- Parent — батьківське вікно;
- Document — документ, область, в яку сценарій виводить інформацію.

Об'єкт Window

Об'єкт window займає в наборі об'єктів особливе, привілейоване місце, оскільки він є основним контейнером, в якому розміщується все те, чим можна керувати за допомогою браузера. Упродовж усього часу, поки вікно браузера відкрите, навіть якщо в ньому не завантажено жодного документа, об'єкт window буде визначений у поточній об'єктній моделі, що зберігається в пам'яті.

Діапазон використання об'єкта Window є широким — від керування вмістом вікна до настроювання його розмірів. Розміри вікна визначають область, де також розміщені смуги прокручування, панелі інструментів, рядок стану і меню — усе, що належить до атрибутів вікна. Кожен фрейм розглядають як окремий об'єкт Window.

У сценарії посилання на властивості та методи об'єкта Window задають безпосереднім звертанням до нього (квадратні дужки у записі означають, що параметри можуть бути відсутні):

Window.властивість Window.метод([параметри])

Оскільки об'єкт Window існує завжди, то в посиланні на об'єкти всередині поточного вікна його назву можна не зазначати:

властивість

метод([параметри])

Розглянемо деякі властивості та методи об'єкта Window.

Властивість Status

Рядок стану в нижній частині вікна браузера після наведення вказівника миші на будь-яке гіперпосилання зазвичай відображає URL-адресу посилання. Проте можна зробити так, що у певні моменти в рядку стану будуть показані спеціальні повідомлення, що надають корисні для користувача відомості. Наприклад, замість того, щоб відображати адресу посилання, можна вивести короткий опис сторінки, присвоївши його як значення властивості Status.

Метод open()

Метод, за допомогою якого можна генерувати нові вікна, — Window.Open (). У ньому використовують три параметри, що визначають такі характеристики, як URL-адресу завантажуваного документа, назву та розмір вікна.

Наведений нижче рядок сценарію відкриває нове вікно abc заданого розміру (висота — 500 пікселів, ширина — 250) із HTML-документом 1.html, розташованим у поточній папці:

```
NewWindow=
Window.Open ("1.html","abc","HEIGHT=500, WIDTH=250")
```

Після цього змінну NewWindow можна використовувати як посилання на це вікно. Наприклад, щоб закрити його, можна скористатися методом Close ():

```
NewWindow.Close()
```

Метод alert()

Цей метод генерує діалогове вікно-попередження, що відображає текст, заданий як параметр методу. Єдина кнопка ОК, напис якої не можна змінити, призначена для того, щоб користувач міг підтвердити, що він прочитав попередження.

Метод confirm()

У діалоговому вікні, яке відображує цей метод, є дві кнопки та текст, заданий як параметр методу. Для більшості версій браузерів і платформ це кнопки ОК і Cancel (Скасувати). Таке вікно називають діалоговим вікном підтвердження.

Метод Confirm () повертає значення true (так), якщо користувач клацає кнопку ОК, і false (ні) — якщо кнопку Cancel (Скасувати). Це діалогове вікно і значення, яке воно повертає, можна використовувати для надання користувачу можливості керувати подальшими діями сценарію.

Метод prompt()

Цей метод генерує діалогове вікно запиту. Воно містить повідомлення,

задане як перший параметр методу, і текстове поле для введення відповіді з підказкою, заданою другим параметром. Дві кнопки, наявні в діалоговому вікні, — ОК і Cancel — дають змогу користувачу закрити діалогове вікно, повернувши у сценарій значення текстового поля (кнопкою ОК) або спеціальне значення null (кнопкою Cancel).

З інформацією, отриманою внаслідок виконання всіх цих методів, можуть далі працювати сценарії: наприклад обробляти відповіді користувача, перевіряти їх правильність, долучати їх до баз даних тощо.

Наведемо приклади ще деяких властивостей, методів та подій об'єкта Window.

Властивості об'єкта Window

- Parent — повертає батьківське вікно;
- Self — повертає посилання на поточне вікно;
- Top — повертає посилання на головне вікно;
- Name — назва вікна.

Методи об'єкта Window

- Open — відкриває нове вікно браузера;
- Close — закриває поточне вікно браузера;
- Focus — робить вікно активним;
- SetInterval — вказівка процедурі виконуватися періодично через задану кількість мілісекунд;
- SetTimeOut — запускає програму через задану кількість мілісекунд після завантаження сторінки.

Події об'єкта Window

- OnFocus — активізація вікна;
- OnResize — змінення користувачем розмірів вікна;
- OnScroll — прокручування вікна користувачем;
- OnLoad — повне завантаження сторінки.

Об'єкт Window має кілька об'єктів-нащадків: Document, History, Navigator, Location, Event і Screen.

Об'єкт Document

Цей об'єкт є центральним в ієрархічній об'єктній моделі й надає всю інформацію про HTML-документ, а також методи та події для роботи з документами. В ньому зберігається весь вміст сторінки. Властивості та методи об'єкта Document впливають здебільшого на вигляд сторінки у вікні. Проте метод Write дає змогу динамічно змінювати вміст у процесі завантаження документа.

Доступ до властивостей і методів об'єкта Document одержують так:

[Window.]Document.властивість [Window.]Document.метод([параметри])

Посилання на вікно Window для доступу до його об'єкта Document є необов'язковим. Далі подано деякі властивості, методи та події цього об'єкта.

Властивості об'єкта Document

- ALinkColor — колір активних посилань на сторінці;
- BgColor — колір тла;
- LastModif ied — дата останнього змінення сторінки, доступна як текстовий рядок;
- LinkColor — колір ще не відвіданих гіперпосилань на сторінці;
- Location — повна URL-адреса документа;
- Ref erer — URL-адреса сторінки, що викликала поточну;
- Vlink — колір відвіданих посилань на сторінці.

Методи об'єкта Document

- Clear — очищає виділений фрагмент;
- Close — завершує сеанс запису в поточний документ (для подальших операцій запису документ буде очищено);
- Write — записує текст у документ, який міститься в поточному вікні;
- WriteLn — записує текст у документ, що міститься в поточному вікні, з переведенням курсору на наступний рядок;
- Open — відкриває зазначений як параметр документ.

Події об'єкта Document

- Onclick — відбувається, коли користувач клацне кнопкою миші на документі;
- OnMouseDown — відбувається, коли користувач натискає кнопку миші і не відпускає її;
- OnMouseOver — відбувається, коли вказівник миші розташований на елементі документа;
- OnMouseMove — відбувається, коли користувач переміщує
- мишу;
- OnDragStart — відбувається, коли користувач починає перетягувати об'єкт за допомогою миші;
- OnError — відбувається, якщо сталася помилка;
- OnKeyDown — виникає під час натискання клавіші;
- OnKeyPress — виникає, коли користувач натиснув клавішу й утримує її;
- OnKeyUp — виникає, коли користувач відпускає клавішу;
- OnLoad — виникає після повного завантаження документа.

Якщо якийсь елемент входить у колекцію документа, то звернутися до нього можна, зазначивши його ім'я або номер у колекції через крапку після імені об'єкта. Наприклад, запис Document. Images (i) . Src="l.gif" означає, що

властивості Src елемента з номером і з колекції Images об'єкта Document потрібно присвоїти значення 1 .gif; іншими словами, і-тий малюнок на цій веб-сто-рінці буде завантажено з файлу 1.gif.

6. СЦЕНАРІЇ

Щоб веб-сторінка була інтерактивною, тобто могла взаємодіяти з користувачем, і динамічною, необхідно використовувати скрипти, або сценарії. Сценарій (script, скрипт) — це програма, написана спеціальною мовою програмування і вбудована в HTML-документ. Сценарії описують усі можливі дії над елементами HTML-документа під час взаємодії з користувачем (наприклад, реакцію на натискання кнопки миші, зміну вмісту сторінки залежно від певних дій користувача тощо).

Мова програмування JavaScript

Стандартною мовою для веб-скриптів є JavaScript — мова програмування, яка дає змогу вбудовувати виконувані модулі в документи, написані в кодах HTML. Програму, створену мовою JavaScript, інтерпретує браузер під час завантаження документа, в який вміщено її код. Проте різні браузери сприймають різні її варіанти. Версія мови JavaScript від корпорації Маикрософт, що має назву JScript, є найближчою до стандарту. Браузер Microsoft Internet Explorer підтримує не лише JScript, а й ще одну мову скрип-тів — Visual Basic Script (VBScript).

За допомогою мови JavaScript, можна, наприклад, зробити так, щоб після клацання зображення лівою кнопкою миші воно змінювало свій вигляд. її засобами можна реалізувати й складнішу поведінку елементів сторінки, скажімо, змусити їх пересуватися з необхідною швидкістю і за бажаною траєкторією.

За допомогою веб-сценаріїв можна створити принципово новий інтерфейс користувача для своєї сторінки. Всі події, генеровані браузером, такі як клацання кнопок, модифікація полів форм і переміщення між сторінками, можна перехопити й обробити засобами JavaScript. Ця мова придатна для розв'язування рутинних завдань, таких як перевірка достовірності даних, опрацювання форм, виконання дій над текстовими і числовими значеннями, тобто тих завдань, які не можна розв'язати за допомогою стандартних засобів мови HTML.

Основні області застосування мови JavaScript:

- динамічне створення документа HTML за допомогою скриптів;
- перевірка достовірності полів форм HTML до передавання їх на сервер;
- локальне введення інформації для керування програмою;
- надання користувачу можливості вибору операцій, виконуваних браузером;
- виведення повідомлень для користувача у діалогових вікнах;
- локальне опрацювання форм, введення інформації користувачем.

Щоб використовувати мову скриптів ефективно, необхідно орієнтуватися

в об'єктній моделі HTML-документа.

Програмний код JavaScript можна помістити в документ HTML у три способи:

- окремі скрипти розмістити в тілі документа, там, де в їхньому використанні є потреба;
- скрипти (функції, оголошення об'єктів) розмістити у заголовній частині документа між тегами <HEAD>...</HEAD>, а використовувати їх у тілі документа;
- зберегти скрипт у файлі (зазвичай із розширенням .js), а в документі дати посилання на нього.

У першому випадку для того, щоб повідомити браузер про використання JavaScript, у тіло HTML-документа потрібно вставити парний тег <SCRIPT> з атрибутом LANGUGE="JavaScript":

<SCRIPT LANGUGE=,,JavaScript"> програма на JavaScript </SCRIPT>

Оскільки браузер Internet Explorer здатний розпізнавати програму на JavaScript, вміщену між тегами <SCRIPT>...</SCRIPT>, для нього зазначений атрибут задавати необов'язково.

Для відвідувачів сторінки, у яких встановлений браузер, що не підтримує JavaScript, після тегів <SCRIPT>...</SCRIPT> вміщують теги <NOSCRIPT>...</NOSCRIPT>, які описують вміст та вигляд без-скриптового варіанта сторінки.

JavaScript, як і будь-яка мова програмування, має набір інструкцій, що описують виконання тих чи інших дій. Синтаксис цих інструкцій схожий на синтаксис операторів у мові Java.

Створимо просту веб-сторінку зі сценарієм. Він виводитиме на екран вікно з повідомленням «!!!». Для цього використаємо метод alert ().

```
<HTML>
<HEAD>
<TITLE>приклад</TITLE>
</HEAD>
<BODY>
<SCRIPT>
alert ("!!!");
</SCRIPT>
</BODY>
</HTML>
```

Рис. 5.15. Найпростіша веб-сторінка зі сценарієм

Слід зазначити, що з міркувань безпеки браузер Internet Explorer за умовчанням блокує всі активні елементи веб-сторінок, тому для коректної роботи з ними необхідно додатково підтверджувати запуск скрипту. Коли браузер завантажує сторінку зі скрип-том, вгорі вікна з'являється панель безпеки, після клацання на якій необхідно вибрати команду Дозволити заблокований вміст. У вікні, що відкриється (рис. 5.16), слід дозволити браузеру запускати активний вміст. Після цього скрипт буде виконуватися без обмежень.

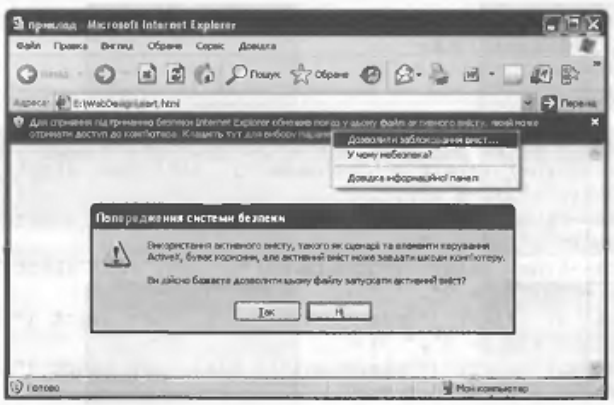


Рис. 5.16. Панель безпеки браузера Internet Explorer та діалогове вікно, що дає змогу запускати активний

Розглянемо простий тест для перевірки навичок із додавання, реалізований за допомогою вбудованого сценарію JavaScript. У ньому використано методи Alert (генерує діалогове вікно-попередження для виведення результатів тесту), Confirm (повертає значення true (істина), якщо користувач клацає кнопку ОК, і false (хибність), якщо користувач клацає кнопку Cancel — це буде використано для перевірки правильності виконання завдань) та Prompt, в якому відображається запитання і надається текстове поле для введення відповіді користувачем. У змінній і накопичуватиметься кількість правильних відповідей.

У цьому скрипті використано команди розгалуження, в яких перевіряються умови s==true або s==false — правильно чи неправильно дано відповідь на запитання тесту. Користувач отримує повідомлення про це, і якщо відповідь була вірною, до змінної і додається 1 (i++).

Ще одне розгалуження із вкладеними розгалуженнями аналізує кількість

правильних відповідей користувача (що міститься у змінній i). Залежно від її числового значення за допомогою методу Alert буде виведена словесна оцінка — «чудово», «добре» або «погано». Останнє повідомлення показує суму балів.

1. У редакторі Блокнот наберіть код веб-сторінки:

```
<HTML>
<TITLE>test</TITLE>
<BODY>
<SCRIPT>
var s, i=0
s=confirm ("2+2=4 ?") ;
if (s==true) {alert ("Правильно!"); i++} else alert ("Hi!")
s=confirm ("2+3=6 ?");
if (s==false) {alert ("Правильно!"); i++} else alert ("Hi!")
s=confirm ("2+4=6 ?");
if (s==true) {alert ("Правильно!"); i++} else alert ("Hi!")
s=prompt ("2+2= ?", " ");
if (s==4) {alert ("Правильно!"); i++} else alert ("Hi!")
s=prompt ("2+3= ?"," ");
if (s==5) {alert ("Правильно!"); i++} else alert ("Hi!")
s=prompt ("2+4= ?"," ");
if (s==6) {alert ("Правильно!"); i++} else alert ("Hi!")
if (i>=5) alert ("чудово")
else if (i>=3) alert ("добре")
else alert ("погано")
alert ("Сума балів="+i)
</SCRIPT>
</BODY>
</HTML>
```

2. Збережіть документ у файлі test.html.
3. Відкрийте веб-сторінку test.html у вікні браузера. У разі потреби розблокуйте активний вміст документа.
4. Дайте відповіді на запитання тесту. Деякі етапи тестування показані на рис. 5.17. Під час тестування викликаються такі методи:
   - перше запитання (2+2=4?) — метод Confirm;
   - повідомлення про правильну відповідь праворуч — метод Alert;
   - четверте запитання (2+2=?) — метод Prompt;
   - повідомлення про неправильну відповідь праворуч — метод Alert;
   - виведення словесної оцінки — метод Alert;
   - повідомлення про кількість балів праворуч — метод Alert.
5. Доповніть тест, щоб у ньому було 8 запитань.

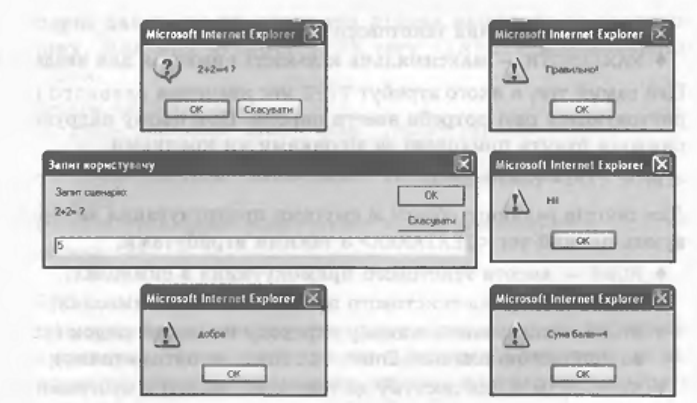


Рис. 5.17. Етапи тестування

Використання форм

Введення інформації можна організувати не лише через діалогові вікна, а й за допомогою форм — наборів елементів керування які забезпечують взаємодію людини з програмою. Користувач може вводити інформацію з клавіатури, а також вибираючи потрібні перемикачі чи прапорці. Його дії потім опрацьовує програма-сценарій.

Елементи форми

Форма може містити поля для введення текстової інформації, списки для вибору заздалегідь визначених відповідей, прапорці, перемикачі, кнопки та інші елементи керування. Розглянемо докладніше деякі з них.

Текстові поля

Для введення текстових даних у спеціальні поля використовують тег <INPUT>, в якого атрибут TYPE має значення text:

<INPUT TYPE="text">

Інші атрибути тегу <INPUT>:

- NAME — ім'я змінної, в якій зберігається введене значення;
- VALUE — початкове значення;
- SIZE — довжина текстового поля;
- MAX LENGTH — максимальна кількість символів для введення.

Цей самий тег, в якого атрибут TYPE має значення password використовують у разі потреби ввести пароль. При цьому надруковані символи будуть приховані за зірочками чи крапками.

<INPUT TYPE="password">

Для текстів великого обсягу зі смугами прокручування використовують парний тег <TEXTAREA> з такими атрибутами:

- ROWS — висота текстового прямокутника в символах;
- COLS — ширина текстового прямокутника в символах;
- WRAP — визначення способу переходу на інший рядок (off — за

допомогою клавіші Enter, virtual — автоматично);

- NAME — ім'я для доступу до текстової області з програми-сценарію.

Перемикачі

Щоб дізнатися про думку відвідувача з того чи іншого приводу, не змушуючи його вводити інформацію, використовують перемикачі з варіантами відповідей. Відвідувач переглядає їх і вибирає потрібний. Перемикачі дають змогу вибрати лише один із запропонованих варіантів.

Для створення перемикачів використовують тег <INPUT>, в якого атрибут TYPE (тип) має значення radio. Групі перемикачів, що стосуються одного питання, обов'язково присвоюють однакове ім'я (NAME). Атрибут VALUE позначає відповідний перемикачу варіант відповіді для розробника форми, тоді як напис після тегу <INPUT> — для користувача. Для вибору одного з перемикачів за умовчанням використовують атрибут CHECKED.

```
<H4>Вибери мову:</H4>
<INPUT TYPE="radio" NAME="lang" VALUE = "українська" CHECKED>
Українська <BR>
<INPUT TYPE="radio" NAME="lang" VALUE ="російська">
Російська <BR>
<INPUT TYPE=r,radio" NAME^'lang" VALUE="англійська">
Англійська <BR>
```

Прапорці

Прапорці дають змогу вибирати кілька варіантів із запропонованих. Для них атрибут TYPE тегу <INPUT> має значення checkbox.

```
<H4>Вибери екзамен:</H4>
<INPUT TYPE="checkbox" NAME=иєхат" УАЬиЕ="українська">
Українська <BR>
<INPUT TYPE="checkbox" NAME»иехати VALUE»ифізикап>
Фізика <BR>
<INPUT TYPE="checkbox" NAME=„exam" VALUE= "математика">
Математика <BR>
```

Списки

Списки, які можна розміщувати у формі так само, як перемикачі та прапорці, також позбавляють відвідувачів веб-сторінки від необхідності вводити інформацію вручну, даючи змогу вибрати відповідь із запропонованих варіантів. Список розміщують між тегами <SELECT> та </SELECT>, а його елементи визначають за
допомогою тегу <OPTION>. Наприклад:

```
Район:
<SELECT>
<OPTION NAME="frank">Франківський
<OPTION NAME="syh">CMxiBCbKMft
<OPTION NAME=„gal„>Галицький
```

</SELECT>

Якщо замість <SELECT> записати <SELECT MULTIPLE>, користувач отримає можливість вибрати зі списку кілька варіантів. Атрибут SIZE=N тегу <SELECT> обмежує кількість показаних елементів списку числом N, після чого буде використано прокручування.

Кнопки

Заповнивши форму текстом та вибравши потрібні елементи керування, користувач повинен мати можливість підтвердити свої дії або скасувати помилково введені дані.

Для підтвердження правильності введення використовують кнопку, яку створює тег <INPUT> з атрибутом TYPE="submit". На цій кнопці буде напис, заданий атрибутом VALUE. Аналогічно створюють кнопку скасування дії: атрибут TYPE у цьому випадку повинен мати значення reset. Звичайна кнопка, з якою можна зв'язати будь-яку дію, має атрибут TYPE="button".

```
<INPUT TYPE="submit" VALUE="Відіслати">
<INPUT TYPE="reset" VALUE="Обчислити форму">
<INPUT TYPE="button" VALUE="Обчислити">
```

Для оригінального оформлення кнопки в неї можна вставити малюнок. Для цього використовують такий синтаксис:

```
<INPUT TYPE="image"  SRC="..." WIDTH=...  HEIGHT=... ALT="..."
                                    VALUE=...>
```

Надсилання форми

Отже, розробник веб-сторінки має змогу отримати відповіді користувача на поставлені запитання, не обмежуючись діалоговими вікнами методів Confirm та Prompt. Форми можна проектувати відповідно до своїх потреб, додавати до них зображення та інші елементи. Після заповнення форми користувач надсилає дані на подальше опрацювання. Включені в документ HTML елементи на зразок полів введення даних, перемикачів, прапорців та кнопок вміщують у тег форми <FORM>. . .</FORM>. Цей тег повинен мати певні атрибути, наприклад:

```
<FORM METHOD="post"  ACTION="/bin/serv" ENCTYPE="text/plain">
```

Атрибут METHOD може мати значення post або get, які визначають різні методи передавання інформації з форми на URL-адресу сценарію-обробника, що зазначена в атрибуті ACTION. У свого провайдера потрібно уточнити, який із методів слід використовувати. Необхідно також знати місце розташування доступних сценаріїв та їхні імена.

Дані форми можна пересилати на певну адресу електронної пошти, якщо задати атрибут ACTION=mailto:адреса_пошти.

Атрибут ENCTYPE визначає, в який спосіб дані форми потрібно кодувати

перед надсиланням на сервер. За умовчанням використовується кодування, під час якого символи замінюються комбінацією символу «% » та шістнадцяткового коду символу тексту в ASCII-таблиці. Щоб дані форми не перетворювати на шістнадцяткові числа, слід задавати значення text /plain.

Наведемо приклад, де користувачу запропоновано заповнити форму, дані якої будуть використані для формування статистичної інформації про вступні іспити у деякий навчальний заклад. Запрошення для заповнення форми записане як заголовок четвертого рівня. Для надсилання форми обрано метод post та зазначено адресу електронної пошти.

Спочатку відвідувачу пропонують ввести своє прізвище у текстове поле (використано підказку у вигляді спливаючого повідомлення). Після цього користувачу потрібно вибрати район, в якому він проживає, у відповідному списку, що міститься між тегами <SELECT> та </SELECT>. Для визначення мови, якою відвідувач бажає здавати іспити, використовуються перемикачі. Іспити вибирають за допомогою прапорців, кілька з яких або навіть усі можна встановити одночасно.

Використання кількох груп варіантів відповідей для різних запитань потребує, щоб елементи кожної такої групи мали однакове значення атрибута NAME у тегах INPUT, але воно було різним для різних груп.

Для надсилання форми передбачена стандартна кнопка з написом Відіслати. Для очищення полів форми від введеної інформації створимо кнопку Оновити форму. Розглянемо наступний приклад:

1. У текстовому редакторі Блокнот наберіть текст HTML-документа. Замість електронної адреси test@tneu.edu.ua у тегу <FORM> введіть власну адресу електронної пошти.

```
<HTML>
<TITLE>Форми</TITLE>
<BODY>
<H4>Заповни форму для складання іспитів</H4>
<FORM METHOD="post" ACTION="miailto:test@tneu.edu.ua "
ENCTYPE="text/plain">
Прізвище: <INPUT TYPE="text" NAME="family" size=25 TITLE="Введи
тут своє прізвище українською мовою"><BR><BR>
З цього списку вибери район, де ти проживаєш: <BR>
<SELECT>
<OPTION NAME="frank">Франківський
<OPTION NAME="syhn">Сихівський
<OPTION NAME="gal">Галицький
</SELECT>
<BR><BR>
Зазнач мову,  якою ти  складатимеш іспити:  <BR>
<INPUT TYPE"radio"  NAME="lang"
VALUE="українська">Українська  <BR>
```

```
<INPUT TYPE="radio"  NAME="lang"
VALUE ="російська">Російська <BR>
<INPUT TYPE="radio"  NAME="lang"
VALUE ="англійська">Англійська <BR><BR>
Вибери іспити:<BR>
<INPUT TYPE="checkbox" NAME="subj" VALUE ="українська">
Українська <BR>
<INPUT TYPE="checkbox" NAME="subj" VALUE ="фізика"> Фізика
<BR>
<INPUT TYPE="checkbox" NAME="subj" VALUE="математика">
Математика <BR>
<INPUT TYPE="subinit"  VALUE ="Відіслатии">
<INPUT  TYPE="reset"  VALUE ="Оновити форму">
</FORM>
</BODY>
</HTML>
```

2. Збережіть документ із назвою anketa1.html.

3. Відкрийте його у вікні браузера.

4. Заповніть форму, давши відповідь на всі запитання. Після цього веб-сторінка матиме приблизно такий вигляд, як на рис. 5.18. Клацніть кнопку Відіслати.

5. Відкрийте свою електронну скриньку. Перевірте інформацію, що надійшла на неї з форми.

Зауважте, що надіслати інформацію можна лише за умови, що на комп'ютері настроєно поштовий клієнт (наприклад, Microsoft Outlook).

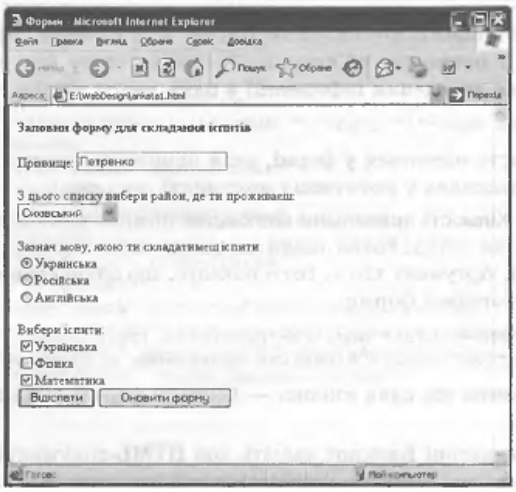


Рис. 5.18. Веб-сторінка із введеною у поля форми інформацією

Розглянемо невелике тестування із вбудованою функцією perevirka, параметром якої є форма з відповідями користувача на запитання. Використано три запитання, для першого правильна відповідь № 2, для другого — № 1, для

третього — № 3.

Відповіді на тестові запитання вибираються у групах перемикачів. Для кожного запитання є своє значення атрибута NAME у тегу INPUT — для першого NAME="one", для другого — NAME="two", для третього — NAME="three". Кожна така група перемикачів інтерпретується у функції перевірки як масив значень.

Функція виконує перевірки такого типу:

```
if  (tests.one[l].Checked)     {i++;}
```

Цей рядок коду означає, що якщо у формі з іменем tests у групі кнопок з іменем one вибране друге значення (елементи масиву нумеруються від 0), тобто вибрана правильна відповідь, то до значення змінної і додається одиниця.

Вираз document.tests.druk.Value = і означає, що значення змінної і присвоєне об'єкту druk — таку назву має текстова область для виведення інформації в один рядок та два стовпця:

```
<TEXTAREA ROWS="1"  COLS="2"  NAME="druk" </TEXTAREA>
```

Ця область міститься у формі, якій присвоєно назву tests. Форма розташована у поточному документі document.

Кнопка "Кількість правильних відповідей" ініціює виконання функції perevirka (this.form) після настання події OnClick (клацання кнопки). Аргумент this. form показує, що обчислення застосовують до поточної форми.

```
<INPUT NAME="check" onclick="perevirka (this.form)" TYPE="button"
                VALUE="Кількість правильних відповідей">
```

Передбачена ще одна кнопка — Оновити форму — для очищення форми.

1. У редакторі Блокнот введіть код HTML-документа.

```
<HTML>
<HEAD>
<TITLE>Тестування</TITLE>
<BODY>
<H4>Тестування</H4>
<SCRIPT>
function perevirka(tests){
var i=0;
if (tests.one[1].checked) {i++;}
if {tests.two[0].checked) {i++;}
if {tests.three[2].checked) {i++;}
document.tests.druk.value = i;
}
</SCRIPT>
```

```
</HEAD>
<FORM METHOD="post" NAME="tests">
2+2=? <BR><BR>
<INPUT TYPE="radiO" NAME="one"
VALUE="1">5 <BR>
<INPUT TYPE="radio" NAME="one" VALUE="2">4 <BR>
<INPUT TYPE="radio" NAME="one" VALUE=" 3 " > 3<BR><BR>
3+3=? <BR><BR>
<INPUT TYPE="radio" NAME="two" VALUE="1">6<BR>
<INPUT  TYPE="radio"  NAME="two" VALUE="2">5<BR>
<INPUT  TYPE="radio"  NAME="two" VALUE="3">7<BR><BR>
4+4=?  <BR><BR>
<INPUT TYPE="radio"  NAME="three" VALUE="1">7<BR>
<INPUT TYPE="radio"  NAME="three" VALUE="2">9<BR>
<INPUT TYPE="radio"  NAME="three" VALUE=" 3 " >8<BR><BR>
<INPUT NAME="check"  onclick="perevirka  (this.form)"
TYPE="button"  VALUE="Кількість  правильних відповідей">
<TEXTAREA  Cols="2"  NAME="druk" ROWS="l" </TEXTAREA>
<INPUT TYPE="reset"  VALUE="Оновити форму">
</FORM>
</BODY>
</HTML>
```

2. Збережіть документ у файлі test.html.
3. Відкрийте збережений документ у браузері. У разі потреби розблокуйте активний вміст веб-сторінки (див. рис. 5.16).
4. Заповніть форму, давши відповіді на всі запитання. Клацніть кнопку Кількість правильних відповідей. Після цього веб-сторінка матиме приблизно такий вигляд, як показано на рис. 5.19.

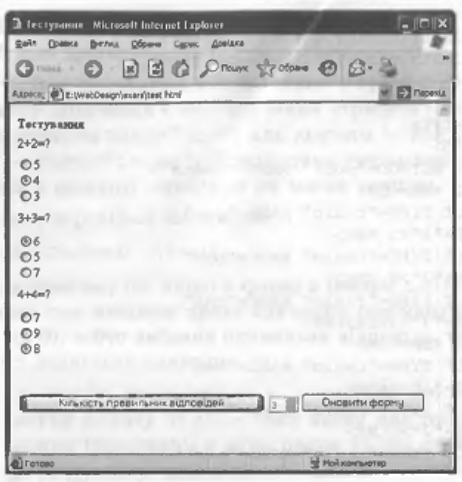


Рис. 5.19. Тестування з використанням форми

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Антоненко В.М., Терейковський І.А., Терейковська Л.О. Основи Web-дизайну. Конспект лекцій. Навч. посібник. К.: МГІ КСУ, 2005. 116 с.
2. Антоненко В. М., Рогушина Ю.В. Сучасні інформаційні системи і технології. Навчальний посібник. - К.: КСУ МГІ, 2005. – 131 с.
3. Вандер Вер Э. JavaScript для чайников. К.: Издательский дом «Вильямс», 2001. – 304 с.
4. Гарнаев А., Гарнаев С. Web программирование на Java и JavaScript. - Санкт-Петербург: БХВ-Петербург, 2002.– 1040 с.
5. Кастаньето Д. Профессиональное PHP программирование. СПб.: Символ-Плюс, 2001. - 912 с.
6. Колбери Р. Освой самостоятельно CGI за 24 часа. — М.: Издательский дом "Вильямс", 2001.— 368 с.
7. Костарев А. PHP в Web-дизайне. С.-Петербург.: BHV, 2002. – 592 стр.
8. Матросов А., Сергеев А., Чаунин М. HTML 4.0. – Санкт-Петербург: БХВ-Петербург, 1999. - 672 с.
9. Паттерсон Л. и др. Использование HTML 4.0. - К.-М.-СПб.: Издат. Дом “Вильямс”. – 1998 – 384с.
10. Пасічник О.Г., Пасічник О.В., Стеценко І.В.. Основи веб-дизайну. Київ. Видавнича група BHV. 2009
11. Терейковський І.А. Підвищення ефективності функціонування корпоративних Web – сайтів. К.: Вісник КНУТД №4 2004 с.41-46.
12. Холл М., Браун Л. Программирование для Web. К.: Издательский дом «Вильямс», 2002. – 1264 с.
13. http://www.postroika.ru
14. http://www.siteforum.ru

ДЛЯ ПРИМІТОК

Підписано до друку 22.06.2012р.
Формат 60х84/16. Папір офсетний.
Друк на дублікаторі. Зам. № 6-1984
Умов.-друк. арк. 3,67. Обл.-вид. арк. 3,92.
Тираж 50 прим.

Віддруковано ФО-П Шпак В.Б.
Свідоцтво про державну реєстрацію: В02 № 924434 від 11.12.2006 р.
Свідоцтво платника податку: Серія Е № 897220
м. Тернопіль, вул. Просвіти, 6
тел. 097 299 38 99, 063 300 86 72
E-mail: tooums@ukr.net