Івано-Франківський національний технічний університет нафти і газу

Р.І. Храбатин
В.В. Бандура

ОСНОВИ СКРИПТОВОГО ПРОГРАМУВАННЯ

Навчальний посібник

Для студентів освітньо-професійної програми «Інженерія програмного забезпечення» рівня вищої освіти «бакалавр»

Івано-Франківськ
2022

УДК 004.2+004.4

Рецензент:
Горбійчук М.І. доктор технічних наук, професор, завідувач кафедри автоматизації та комп'ютерно-інтегрованих технологій

Рекомендовано методичною радою університету
(протокол № від 2020р.)

Р.І. Храбатин
Л.В. Саманів
Б Основи скриптового програмування: навчальний посібник. – Івано-Франківськ: ІФНТУНГ, 2022. – 354с.

МВ 02070855 - - 2022

Навчальний посібник містить практичний навчальний матеріал щодо вивчення основ програмування. Може бути використаний студентами денної та заочної, дистанційної форм навчання. Призначено для підготовки фахівців із освітньо-професійної програми «Інженерія програмного забезпечення» рівня вищої освіти «бакалавр»

УДК 004.2+004.4

Завідувач кафедри інженерії програмного забезпечення В.І. Шекета
Член експертно-рецензійної
комісії університету, В.В. Бандура
Нормоконтролер Г. Я. Томашівська
Провідний бібліотекар Г. М. Мацюк

МВ 02070855 - 2022

ЗМІСТ

ВСТУП

Дисципліна «Web-програмування» є однією з базових у підготовці висококваліфікованих спеціалістів у сфері Web-технологій, які займаються створенням, впровадженням, аналізом і супроводженням Web-сайтів і Web-порталів, питаннями Web-дизайну і комп'ютерною графікою.

Отримані знання дозволяють на професійній основі проектувати і програмувати вміст HTML-документів, формувати інтерфейс майбутніх розробок, працювати зі шрифтами, стилями, таблицями, списками, фреймами, картами графічних посилань, зображеннями, відеокліпами, банерами, формувати сценарій на мові скриптів, включати в склад документів Flash-ролик, просувати на ринку спеціалізовані Web-сайти, які мають сучасний графічний користувацький інтерфейс і напрям на ведення електронного бізнесу та проведення маркетингових досліджень.

Дисципліна «Web-програмування» займає важливе місце у навчальному процесі з напряму підготовки 6.030502 «Економічна кібернетика», оскільки дає студентам основи програмування, дозволяє отримати уміння і навички у сфері технології розробки Web-додатків з використанням мови гіпертекстової розмітки документів HTML, мов сценаріїв, інструментальних середовищ типу HomeSite і Flash-технологій.

Навчальний посібник містить лекції, лабораторні роботи, завдання, питання для повторення і самостійної роботи, тестові завдання. Усі розділи і теми послідовні і тісно пов'язані між собою.

У процесі підготовки навчального посібника були використані джерела, перелік яких наведений у літературі.

Призначений для студентів 3 курсу денної форми навчання напряму підготовки 6.030502 «Економічна кібернетика» факультету економіки та оподаткування.

1. HTML

Найпростішим компонентом Web є HTML (Hyper Text Markup Language). Web-сторінка (документ HTML) – текстовий файл мовою HTML формату *.htm або *.html розміщений в World Wide Web (WWW). WWW – всесвітня павутина, розподілена система доступу до гіпертекстових документів, що існує в Інтернеті. Web-сторінка, крім тексту, може містити гіпертекстові посилання, за допомогою яких можна переходити до інших Web-сторінок і переглядати їх, а також містити вставки у вигляді графіки, анімації, відеокліпів і музики. Для перегляду Web-сторінок можна використовувати, наприклад Microsoft Internet Explorer або Netscape Navigator (переглядач або браузер).

Мова HTML дозволяє:

– створювати й редагувати Web-сторінки, у тому числі свою домашню Web-сторінку, яку можна потім розмістити в Інтернеті;

– редагувати документи HTML, отримані з Інтернету, так щоб функціонували всі впроваджені в документ об’єкти (картинки, анімації тощо);

– створювати мультимедійні презентації, слайдшоу, демон- страційні проекти завдяки гіпертекстовим посиланням і можли- вості вставляти в документ HTML малюнки, діаграми, анімації, відеокліпи, музичний і мовний супровід, текстові спецефекти (наприклад рядок, що біжить).

Існують три способи створення Web-сторінок (або документів HTML):

1. Використання текстового редактора *Блокнот* (Notepad), вбудованого в Windows, і перегляд результатів за допомогою браузера. Технологія створення Web-сторінки така:

– у редакторі *Блокнот* створюється файл Web-сторінки, який зберігається з розширенням *.htm. Потім цей файл завантажується і переглядається програмою Internet Explorer. Для виклику редактора *Блокнот* з метою редагування файла Web-сторінки під час її перегляду в Internet Explorer використовується пункт меню «Вид», «Источник». Після збереження файла і виходу із ре-

дактора *Блокнот* для перегляду відредагованої сторінки потрібно натиснути клавішу F5 або кнопку «Обновить» на панелі інструментів Internet Explorer.

2. Використання спеціальних редакторів документів HTML, наприклад Hot Metal Light, Hot Dog Professional, MS Front Page, Htmlpad та ін.

3. Використання Word, де створюється текст документа, який потім конвертується в HTML-формат.

Основні поняття мови HTML.

1. **Елемент** – це конструкція мови HTML або контейнер, що містить дані. Web-сторінка являє собою набір елементів.

2. **Тег** – це стартовий і кінцевий маркери елемента. Теги виз- начають границі дії елементів і відокремлюють елементи один від одного. Практично усі теги (за виключенням деяких) парні, тобто є «відкриваючий» і «закриваючий» теги. У тексті Web-сто- рінки теги записують у кутові дужки, наприклад:

<HTML>. Кінцевий тег завжди задається з косою рискою: </HTML>.

3. **Гіперпосилання** – фрагмент тексту, який є покажчиком на інший файл або об’єкт. Гіперпосилання дозволяють переходити від одного документа до іншого.

4. **Фрейм** – область гіпертекстового документа зі своїми смугами прокручування.

5. **Апплет** – програма, передана на комп’ютер клієнта у виг- ляді окремого файла, яка запускається під час перегляду Web- сторінки.

6. **Скрипт** – програма, включена до складу Web-сторінки для розширення її можливостей.

7. **Завантаження (Download)** – копіювання документа з Web-сервера на комп’ютер клієнта. Upload – копіювання документа з комп’ютера клієнта на Web-сервер – використовується під час створення власної Web-сторінки.

Загальна структура типового найпростішого документа HTML:

<COMMENT>Коментар</COMMENT>
<HTML>

<HEAD>
<TITLE>Назва документа</TITLE>
</HEAD>
<BODY>
Тут розташований текст самого документа HTML.
</BODY>
</HTML>

Теги структури документа

<HTML> . . . </HTML>	визначає початок і кінець документа
<HEAD> . . . </HEAD>	визначає початок і кінець заголовка документа
<TITLE> . . . </TITLE>	вказується заголовок поточної сторінки
<BODY> . . . </BODY>	визначає початок і кінець тіла документа
Атрибути тегу BODY	
BGCOLOR=« . . . »	визначає колір фона
BACKGROUND=« . . . »	вказує адресу файлу для Background (рекомендується підбирати колір bgcolor, максимально наближений до кольору background)
TEXT=« . . . »	задає колір тексту
LINK=« . . . »	задає колір гіперзв’язків (посилань)
VLINK=« . . . »	задає колір посилань, де користувач уже побував
ALINK=« . . . »	задає колір активного гіперзв’язку
<! . . . >	використовується для вставки коментаря
Приклад використання: <BODY bgcolor=«f0ffff» BACKGROUND=«31.jpg» LINK=«880000» VLINK=«0a00ab» ALINK=«ffff00»>	

Параграфи, роздільники, заголовки

Параграфи і абзаци:	
 	початок нового рядка
<P>	початок нового абзацу, з відступом
Заголовки:	
<H1> . . . </H1>	заголовок першого рівня
<H6> . . . </H6>	заголовок шостого рівня

Роздільники:	
<HR>	горизонтальна лінія, що йде через весь екран
Атрибути тегу HR:	
SIZE=« . . . »	визначення товщини лінії (у пікселях)
WIDTH=« . . . »	визначення довжини лінії (у пікселях або у відсотках від ширини сторінки)
COLOR=« . . . »	визначення кольору лінії
ALIGN=left ALIGN=right ALIGN=center	визначення способу вирівнювання лі- нії (ліворуч, праворуч, по центру)
NOSHADE	без використання тривимірних ефектів
Приклади використання: <HR COLOR=«008000» SIZE=«1» NOSHADE> <HR SIZE=«3» WIDTH=«50» ALIGN=right> <HR SIZE=«5» WIDTH=«50%» ALIGN=center>	

Відображення тексту

Зображення у правому стовпці відображають приклад застосування описуваного тегу, встановлення розміру базового шрифту.

<BASEFONT SIZE= . . . >	Діапазон – від 1 до 7, (за замовчуванням встановлюється 3)
<BIG> . . . </BIG>	текст буде відображатися шрифтом більшого розміру
<SMALL> . . . </SMALL>	текст буде відображатися шрифтом меншого розміру
<B> . . . </B>	напівжирний текст
<I> . . . </I>	текст, виділений курсивом
<U> . . . </U>	підкреслений текст
<STRIKE> . . . </STRIKE>	перекреслений текст
<S> . . . </S>	перекреслений текст
<TT> . . . </TT>	шрифт фіксованої ширини
<BLINK> . . . </BLINK>	миготливий текст
_{. . .}	текст зміщується вниз
^{. . .}	текст зміщується нагору
<FONT> . . . </FONT>	визначення кольору, розміру і гарнітури тексту

Атрибути тегу FONT:	
COLOR=« . . . »	визначення кольору тексту
FACE=гарнітура	визначення типу гарнітури
SIZE=« . . . »	визначення розміру шрифту від 1 до 7 або відносно базового розміру за допомогою значень +n або –n
Приклади використання: <FONT FACE=«arial» SIZE=«2» COLOR=«5e0000»> <FONT SIZE=«+3» COLOR=«red»>	

Відображення інформації

Зображення у правому стовпці дає приклад застосування описуваного тегу.

<ADDRESS> . . . </ADDRESS>	відображає інформацію про адресу
<BLOCKQUOTE> . . . </BLOCKQUOTE>	виділяє текст відступами ліворуч і праворуч. Використовується для цитування
<CITE> . . . </CITE>	використовується для посилань (назви книг, фільмів тощо) і цитат;
<CODE> . . . </CODE>	використовується для частин початкового тексту програм
<SAMP> . . . </SAMP>	використовується для розміщення у документі друку програм тощо
<EM> . . . </EM>	використовується для виділення тексту
<DFN> . . . </DFN>	дає приклад для обговорюваного терміна або поняття
<KBD> . . . </KBD>	використовується для виділення тексту, що набирається користувачем на клавіатурі
<VAR> . . . </VAR>	використовується для позначення змінних, переданих з командами
<DIV> . . . </DIV>	використовується для створення розділу в тексті
Атрибути тегу DIV	
ALIGN=left ALIGN=right	використовується для визначення способу вирівнювання блоку

ALIGN=center	(ліворуч, праворуч, по центру)

Гіпертекстові зв’язки

<A HREF=«Адреса»>гаряча точка</A>	гіпертекстовий зв’язок з іншою сторінкою сервера;
<A HREF=«#Имя»>гаряча точка</A>	гіпертекстовий зв’язок у цьому документі
<A NAME=«Ім’я»>місце переходу</A>	визначає місце переходу по гіперзв’язку в документі
<A HREF=«Адрес#Ім’я»>гаряча_крапка</A>	гіперзв’язок до певної крапки на іншій сторінці сервера
<A HREF=«Адреса»><IMG SRC=«Ім’я»></A>	гіперзв’язок по зображенню з іншою сторінкою сервера (активне зображення)
<A HREF=«file://fname.mov»> гаряча точка </A>	гіперзв’язок до файла зображення, відеофайла, аудиофайлу тощо
Атрибути тегу A HREF:	
<A HREF=«Адреса» TARGET=«_TOP»> гаряча точка </A>	атрибут TARGET указує, що сторінка, задана посиланням, завантажиться у новому екземплярі браузера
<A HREF=«Адреса» TARGET=«_TOP»> гаряча точка </A>	атрибут TARGET вказує, що сторінка, задана посиланням, завантажиться у новому вікні
<A HREF=«Адреса» TARGET=«_BLANK»> гаряча точка </A>	атрибут TARGET указує, що сторінка, задана посиланням, завантажиться у новому порожньому вікні
<A HREF=«Адреса» TARGET=«_SELF»> гаряча точка </A>	атрибут TARGET указує, що сторінка, задана посиланням, завантажиться у вікні, що містить посилання

<A HREF=«Адреса» TARGET=«_PARENT»> гаряча точка </A>	атрибут TARGET указує, що сторінка, задана посиланням, завантажиться у вікні, що є безпосереднім власником набору фреймів
<A HREF=«Адреса» TARGET=«Ім'я_фрейму»> гаряча точка </A>	атрибут TARGET вказує, що сторінка, задана посиланням, завантажиться у зазначеному фреймі

Списки

У правому стовпці дається приклад застосування описуваного тегу.

Елементи списку	
<LI>aaa<LI>bbb<LI>ccc	для того щоб задати список, кожний елемент його позначається (можна в рядок)
Атрибути тегу LI (значення зазначені нижче):	
TYPE=«CIRCLE/DISK/SQUARE»	тип мітки для цього і наступних елементів у неупорядкованому списку
TYPE=«A/a/I/i/1»	тип нумерації для цього і наступних елементів;
VALUE=«n»	вказується номер, з якого починати відлік
Неупорядковані списки	
<UL> . . . </UL>	неупорядкований список
Атрибути тегу UL:	
<UL COMPACT> . . . </UL>	атрибут COMPACT указує, що список буде представлений у більш компактному вигляді
<UL TYPE=«CIRCLE»> . . . </UL>	атрибут TYPE=«CIRCLE» указує, що мітка буде у вигляді кружка
<UL TYPE=«DISK»> . . . </UL>	атрибут TYPE=«DISK» вказує, що мітка буде у вигляді диску (за замовчуванням)

<UL TYPE=«SQUARE»> . . . </UL>	атрибут TYPE=«SQUARE» вказує, що мітка буде у вигляді квадрата
Упорядковані списки:	
<OL> . . . </OL>	упорядкований список
Атрибути тегу OL:	
<OL COMPACT> . . . </OL>	атрибут COMPACT вказує, що список буде представлений у більш компактному вигляді
<OL TYPE=«A»> . . . </OL>	атрибут TYPE=«A» указує, що мітки будуть у вигляді прописних букв
<OL TYPE=«a»> . . . </OL>	атрибут TYPE=«a» указує, що мітки будуть у вигляді малих букв
<OL TYPE=«I»> . . . </OL>	атрибут TYPE=«I» вказує, що мітки будуть у вигляді великих римських цифр
<OL TYPE=«i»> . . . </OL>	атрибут TYPE=«i» указує, що мітки будуть у вигляді маленьких римських цифр
<OL TYPE=«1»> . . . </OL>	атрибут TYPE=«1» указує, що мітки будуть у вигляді арабських цифр (за замовчуванням)
<OL START=«3»> . . . </OL>	атрибут START=«3» визначає, що список буде пронумерований із зазначеної цифри
Меню:	
<MENU> . . . </MENU>	список у вигляді меню
<MENU COMPACT> . . . </MENU>	атрибут COMPACT указує, що список буде представлений у більш компактному вигляді
Списки визначень	
<DL> . . . </DL>	список визначень (словник термінів)
<DT> . . .	цей тег стоїть перед терміном;
<DD> . . .	цей тег стоїть перед визначенням зазначеного вище Терміна

Приклад використання:

```
<DL>
<DT>Internet
<DD>Всесвітня комп'ютерна мережа мереж,
що надає доступ до інформації з усього світу.
<DT>Гіпертекст.
<DD>Система гіпертексту встановлює зв'язки між невеликими
фрагментами інформації, спрощуючи пошук необхідних даних.
<DL>
```

Internet
Всесвітня комп'ютерна мережа мереж, що надає доступ до інформації з усього світу.
Гіпертекст
Система гіпертексту встановлює зв'язки між невеликими фрагментами інформації, спрощуючи пошук необхідних даних

Таблиці

<TABLE> . . . </TABLE>	визначає початок і кінець таблиці
Атрибути тегу TABLE:	
BGCOLOR=« . . . »	визначає колір фона всієї таблиці
BACKGROUND=« . . . »	указує адресу файла для Background (ре- комендується підбирати колір bgcolor, максимально наближений до кольору background)
BORDER=«n»	задає обрамлення, утворене лініями шириною n пік селів
WIDTH=« . . . »	задає ширину таблиці (у пікселях або відсотках);
ALIGN=left ALIGN=right ALIGN=center	задає спосіб вирівнювання таблиці по ширині сторінки (ліворуч, праворуч, по центру)
CELLSPACING=« . . . »	задає у пікселях відстань між границями комірки та її вмістом;
CELLPADDING=« . . . »	задає у пікселях інтервал між комірками

Приклад використання:

```
< TABLE CELLSPACING=2 CELLPADDING=3 BORDER=
```

1 ALIGN=«Center» BGCOLOR=«Gray» BACKGROUND=«Bk10.gif» WIDTH=100>	
<TR> . . . </TR>	визначає початок і кінець рядка в таблиці
Атрибути тегу TR:	
BGCOLOR=« . . . »	визначає колір фону даного рядка
BACKGROUND=« . . . »	указує адресу файла для Background даного рядка
BORDER=«n»	задає обрамлення, утворене лініями шириною n пік селів
ALIGN=left ALIGN=right ALIGN=center	задає спосіб вирівнювання вмісту комірок у даному рядку (ліворуч, праворуч, по центру)
VALIGN=top VALIGN=bottom VALIGN=center VALIGN=baseline	задає спосіб розміщення вмісту комірок у даному рядку по вертикалі (зверху, знизу, по центру, по базовій лінії)
Приклад використання: < TR ALIGN=«center» VALIGN=«middle»>	
<TD> . . . </TD>	визначає початок і кінець комірки в таблиці
Атрибути тегу TD:	
BGCOLOR=« . . . »	визначає колір фону даної комірки
BACKGROUND=« . . . »	указує адресу файла для Background даної комірки
WIDTH=« . . . »	задає ширину комірки (у пікселях або відсотках від ширини таблиці);
ALIGN=left ALIGN=right ALIGN=center	задає спосіб вирівнювання вмісту комірки (ліворуч, праворуч, по центру)
VALIGN=top VALIGN=bottom VALIGN=center VALIGN=baseline	задає спосіб розміщення вмісту комірки по вертикалі (зверху, знизу, по центру, по базовій лінії);
ROWSPAN=«n»	вказує, що дана комірка охоплює n сусідніх рядків

COLSPAN=«n»	указує, що дана комірка охоплює n сусідніх стовпців
NOWRAP	вказує, що у даній комірці відключа- ється режим автоматичного розподілу тексту по всій комірці. Тобто буде відображатися лише та частина тексту, яка вміщається у комірці по довжині
Приклад використання: <TD ALIGN=«Center» VALIGN=«Baseline» COLSPAN=«2» WIDTH=«30» HEIGHT=«30» NOWRAP>	
<TH> . . . </TP>	визначає початок і кінець комірки із заголовком у таблиці
Атрибути тегу TH:	
BGCOLOR=« . . . »	визначає колір фону даної комірки
BACKGROUND=« . . . »	указує адресу файла для Background даної комірки
WIDTH=« . . . »	задає ширину комірки (у пікселях або відсотках від ширини таблиці)
ALIGN=left ALIGN=right ALIGN=center	задає спосіб вирівнювання вмісту комірки (ліворуч, праворуч, по центру)
VALIGN=top VALIGN=bottom VALIGN=center VALIGN=baseline	задає спосіб розміщення вмісту комірки по вертикалі (зверху, знизу, по центру, по базовій лінії)
ROWSPAN=«n»	указує, що дана комірка охоплює n сусідніх рядків
COLSPAN=«n»	указує, що дана комірка охоплює n сусідніх стовпців
NOWRAP	указує, що у даній комірці відключається режим автоматичного розподілу тексту по всій комірці. Тобто буде відображатися лише та частина тексту, яка вміщається у комірці по довжині

Приклад використання:
< TH ALIGN=«Center» VALIGN=«Baseline» COLSPAN=«4»
WIDTH=«50»>
Приклад взаємного розташування тегів при описі таблиці:
<TABLE BORDER=0>
<TR>
<TD></TD>
</TR>
</TABLE>

Фрейми

<FRAMESET> . . . </FRAMESET>	визначає початок і кінець документа із фреймами. Використовується замість тегу BODY
Атрибути тегу FRAMESET:	
BORDER=«n»	задає обрамлення фреймів, утворене лініями шириною n пік селів
BORDERCOLOR=« . . . »	задає колір обрамлення
COLS=« . . . , . . . , . . . »	задає горизонтальне розташування фреймів. У списку вказується ширина кожного фрейму в пікселях або відсотках. Ширину останнього фрейму можна вказати як *, тобто увесь інший простір
ROWS=« . . . , . . . , . . . »	задає вертикальне розташування фреймів. У списку вказується висота кожного фрейму в пікселях або відсотках. Висоту останнього фрейму можна вказати як *, тобто увесь інший простір
Приклад використання: <FRAMESET COLS=«179, *»>	
<FRAME>	визначає фрейм у наборі фреймів
Атрибути тегу FRAME:	
BORDER=«n»	задає обрамлення фрейму, утвореного лінією шириною n пік селів
BORDERCOLOR=« . . . »	задає колір обрамлення
SRC=« . . . »	задає адресу (URL) файла, яка буде відображатися у цьому фреймі
NAME=« . . . »	задає ім’я цього фрейму

SCROLLING=yes SCROLLING=no SCROLLING=auto	описує наявність і тип лінії прокручування (є завжди, ніколи ні, додається при необхідності)

NORESIZE	забороняє змінювати розміри фреймів
MARGINHEIGHT=« . . . »	задає розміщення над фреймом і під ним областям вільного простору висотою по n пікселів
MARGINWIDTH=« . . . »	задає розміщення ліворуч і праворуч від фрейму областей вільного простору висотою по n пікселів

Приклад використання:
<FRAME NAME=«Framea» SRC=«Begin.htm» MARGINHEIGHT=1 MARGINWIDTH=1 SCROLLING=«Auto» NORESIZE>

Приклад взаємного розташування тегів при описі набору фреймів:

```
<FRAMESET ROWS=«200,*»>
 <FRAME NAME=«Frame 1» SRC=«A1.htm» SCROLLING=«AUTO»>
 <FRAME NAME=«Frame 2» SRC=«B1.htm» SCROLLING=«AUTO»>
</FRAMESET>
```

Створення текстового рядка, що біжить (тільки для Internet Explorer):

<MARQUEE>ТЕКСТ</MARQUEE> – прапорець, що створює рядок.
<MARQUEE DIRECTION=left>ТЕКСТ</MARQUEE> – якщо рядок, що біжить, потрібно направити зправа наліво.
<MARQUEE DIRECTION=right>ТЕКСТ</MARQUEE> – рух зліва направо.
scroll – стандартний рух від правого краю до лівого.
slide – напис один раз пробігає від правого краю до лівого, де і залишається.
.alternate – рух від правого краю сторінки до лівого і назад. Нескінченний цикл.
<MARQUEE LOOP=n BEHAVIOR=scroll>ТЕКСТ</MARQUEE> – обмеження числа циклів. Значення n оператора LOOP указує число повторень циклу.
<MARQUEE WIDHT=n>ТЕКСТ</MARQUEE> – указати ширину ділянки, займаної рядком, що біжить, де n – ширина тієї частини сторінки, на якій розташований рядок, що біжить. Значення n вказується як у пікселях, так і у відсотках від загальної ширини видимої частини сторінки.
<MARQUEE scrollamount=n>ТЕКСТ</MARQUEE> – регулювання руху напису по екрану. Тут n – число пікселів.
<MARQUEE scrolldelay=t>ТЕКСТ</MARQUEE> – у даному випадку змінна величина – час t – вимірюється у мілісекундах. Метод задання швидкості полягає у вказівці часу, через який текст буде перемальований на екрані заново.
<FONTSIZE=n><MARQUEE> ТЕКСТ</MARQUEE></FONT> – можливість указувати величину шрифту тексту в рядку.

<MARQUEE BGCOLOR=n> ТЕКСТ </MARQUEE> – зафарбувати поверхню рядка, що біжить, у який-небудь колір, де n можна задати у вигляді шістнадцятирічного числа або написавши його назву
<MARQUEE HEIGHT=n>ТЕКСТ</MARQUEE> – указати висоту рядка, що біжить, задаючи величину n у пікселях.

Відтворення звуку

Для відтворення звуку (файл *.mid) після завантаження документа HTML у браузер (тобто у фоновому режимі) потрібно записати наступну команду: <bgsound src=«/windows/canyon.mid» loop=1>. Можна також використовувати файл формату *.wav. Число відтворень музики loop можна збільшити з 1 до n

Список кольорів символів HTML

16 базових кольорів:
aqua – бірюзовий; black – чорний; blue – синій; gray – сірий; green – зелений; lime – яскраво-зелений; maroon – темно-червоний; white – білий; navy – темно-синій; olive – маслиновий; purple – фіолетовий; red – червоний; silver – яскраво-сірий; teal – яскраво-блакитний; yellow – жовтий; fuchsia – яскраво-фіолетовий.

Назва	Колір	Rgb-коди
aquamarine	Аквамарин	#7FFFD4
beige	Бежевий	#F5F5DC
black	Чорний	#000000
blue	Синій	#0000FF
brown	Коричневий	#A52A2A
chocolate	Шоколадний	#D2691E
coral	Кораловий	#FF7F50
crimson	Малиновий	#DC143C
cyan	Ціан	#00FFFF
gold	Золотий	#FFD700
gray	Сірий	#808080
green	Зелений	#00FF00
indigo	Індиго	#4B0082
khaki	Хакі	#F0E68C
magenta	Ліловий (Фуксін)	#FF00FF
orange	Оранжевий	#FFA500
pink	Розовий	#FFC0CB

purple	Пурпурний	#800080
Red	Червоний	#FF0000

seagreen	Морської хвилі	#2E8B57
silver	Срібний	#C0C0C0
turquoise	Бірюзовий	#40E0D0
violet	Фіолетовий	#EE82EE
white	Білий	#FFFFFF
yellow	Жовтий	#FFFF00

Міняючи Rgb-коди, можна підбирати бажані кольори тексту і фону. Для базових кольорів можна використовувати приставку dark для задавання темних кольорів: darkblue, darkred і т.п.; або light – для світлих: lightgreen, lightyellow

Приклад

Перегляньте цей html-документ за допомогою браузера

```
<html>
<head>
<title>bgcolor</title>
</head>
<body text=000000 bgcolor=ffffff>
<table border=0 align=center>
<tr><td><form>
    <input type=button value=«червоний»
    onClick=«document.bgColor=‘ff0000’» >
    <input type=button value=«жовтий»
    onClick=«document.bgColor=‘ffff00’»>
    <input type=button value=«синій»
    onClick=«document.bgColor=‘0000ff’»
    >
    <input type=button value=«голубий»
    onClick=«document.bgColor=‘87ceeb’»
    >
    <input type=button value=«кораловий»
    onClick=«document.bgColor=‘ff7f50’» >
</form></td>
</table>
```

</body>
</html>

Форми

Практично всі сучасні WWW-браузери дозволяють користувачеві, заповнивши спеціальну форму, виконувати деякі дії на вашому Web-сервері. Коли форма інтерпретується Web-браузером, створюються спеціальні екранні елементи GUI, такі як поля введення, кнопки підтвердження (check-boxes), кнопки вибору альтернатив (radio-buttons), меню, що випадають, «скроллируючі» списки, кнопки тощо. Коли користувач, заповнивши форму, натискає кнопку «Підтвердження» (SUBMIT – спеціальний тип кнопки, який задається при описі документа), інформація, введена користувачем у форму, посилається Web-серверу для обробки й передачі іншим програмам, що працюють під сервером, відповідно до CGI (Common Gateway Interface) інтерфейсом. За допомогою мови написання сценаріїв Javascript (Vbscript) можна перевірити правильність заповнення форми до її відправлення серверу, обробити натискання інших (відмінних від SUBMIT) кнопок і багато інших подій.

При написанні сценарію використовуються мови VBScript та JavaScript (JScript). Написання сценарію здійснюється таким чином:

На мові VBScript	**На мові JavaScript**
<SCRIPT language= «VBScript» type= «text/vbscript»> …текст сценарію на мові VBScript </SCRIPT>	<SCRIPT language= «JavaScript» type= «text/javascript»> …текст сценарію на мові JavaScript </SCRIPT>

Для забезпечення сумісності з усіма броузерами необхідне використання атрибуту «language» і атрибуту «type».

Для того щоб старі браузери, які не підтримують сценаріїв, не відображали код програми, необхідно текст сценарію узяти в

тег <!-- --> як це показано нижче.

На мові VBScript	На мові JavaScript
<SCRIPT language= «VBScript» type= «text/vbscript»> <!-- ‘це коментар на мові VBScript MsgBox «Привіт!» --> </SCRIPT>	<SCRIPT language= «JavaScript» type= «text/javascript»> <!-- //це коментар на мові JavaScript /*Цей коментар займає декілька строк*/ alert(«Привіт!»); --> </SCRIPT>

При створенні сторінки тег <SCRIPT> можна розміщувати у будь-якому місці сторінки. Стандарти HTML рекомендують розміщувати тег <SCRIPT> у заголовку сторінки, тобто між тегами <HEAD> та </HEAD>.

Для браузерів, які не підтримують сценаріїв, або при відключенні підтримки сценаріїв, можна скористатися тегом
<noscript>.

```
<NOSCRIPT>
<B>Ваш браузер не підтримує JavaScript
або його підтримка відключена</B>
</NOSCRIPT>
```

Роздільником операторів у VBScript є символ «:», а в JavaScript – «;». Наприклад:

На мові VBScript	На мові JavaScript
<SCRIPT language= «VBScript» type= «text/vbscript»> MsgBox «Привіт!»: MsgBox «Ще привіт!» </SCRIPT>	<SCRIPT language= «JavaScript» type=”text/javascript”> alert(«Привіт!»); alert(«Ще привіт!»); </SCRIPT>

Якщо текст сценарію досить великий, його можна помістити в окремий текстовий файл.

Підключення файла сценарію до web-сторінки слід здійснювати за допомогою атрибуту «src» тегу <SCRIPT>.

Підключення файла сценарію «myvb.vbs», написаного на мові VBScript	Підключення файла сценарію «myjs.js», написаного на мові JavaScript
<SCRIPT src= «myvb.vbs» language= «VBScript» type= «text/vbscript»> </SCRIPT>	<SCRIPT src= «myjs.js» language= «JavaScript» type= «text/javascript»> </SCRIPT>

Слід зазначити, що в такому випадку недоречно писати текст сценарію між тегами <SCRIPT> та </SCRIPT>, тому що браузер у такому випадку проігнорує код програми.

У наведених прикладах файли із текстами сценаріїв повинні знаходитись у тій же папці, що і сторінка. Якщо, наприклад, файл знаходиться у підпапці «script», то використовується такий код:
<SCRIPT src= «script/myjs.js» language= «JavaScript» type= «text/javascript»></SCRIPT>.

Можна задати також файл, що знаходиться на іншому вузлі. Наприклад:
<SCRIPT src= «http://www.host.com/myscripts.js» language= «JavaScript» type= «text/javascript»></SCRIPT>
Тег FORM.

Усі форми починаються тегом <FORM> і закінчуються тегом </FORM>. Між цими тегами й розташовуються елементи фор- ми, такі як кнопки, поля введення тощо. Крім елементів форм, там можна розташовувати будь-які тексти й майже будь-які теги HTML.

Синтаксис тегу FORM:
<FORM
ACTION=
«URL»
METHOD= «get | post»
NAME= «ім’я»
ENCTYPE= «application/x-www-form-urlencoded | multipart/form-data»
TARGET= «вікно, за замовчуванням _sefl»
TITLE= «текст підказки»
LANG= «мова (ru, en, de, і т.д.)»

LANGUAGE= «Javascript | Jscript | Vbscript | VBS»
CLASS= «ім’я класу CSS»

STYLE= «безпосередній стиль»
ID= «ідентифікатор»
ONRESET= «оброблювач події»
ONSUBMIT= «оброблювач події»

□

□

…….....
Елементи форми й інші елементи HTML
…….....
</FORM>

Тут, як і далі, підкреслені ті значення, які беруться за замовчуванням.

Жоден з перерахованих параметрів тегу FORM не є обов'язковим. Розглянемо параметри:

ACTION

Указує URL, куди слід передати інформацію, занесену у фор- му. Звичайно, це адреса CGI програми або e-mail, оформлений через **mailto**:. Однак досить часто можна зустріти й інші речі, ти- пу **javascript**:0, коли форма обробляється сценарієм. Якщо ACTION не зазначений, то інформація передається тому доку- менту, у якому перебуває форма.

METHOD

При використанні методу GET (за замовчуванням), інформація з форми додається у кінець URL, який був зазначений в ACTION. Оскільки максимальна довжина URL обмежена, цей метод варто використовувати тільки для найпростіших форм. Метод POST кращий, тому що даний метод передає всю інформацію про форму негайно після звертання до зазначеного URL. CGI-програма одержує дані із форми в стандартний потік вводу.

NAME

Задає ім’я форми для використання у програмах-сценаріях.

ENCTYPE

Завжди слід використовувати значення за замовчуванням (просто опускати цей параметр), крім тих випадків, коли потрібно передати файл від клієнта до сервера *(див. нижче)*.

TARGET

Також як і в тегу <A…>, указує вікно, в яке потрапить уся вихідна інформація форми. За замовчуванням – у те ж вікно, де й сама форма.

TITLE

Підказка, яка буде з'являтися, коли мишка вказує на форму (MSIE 4.0+).

LANG

Мова – «ru» – російська, «de» – німецька і т.д.

LANGUAGE

Мова, яка використовується для інтерпретації елементів сценарію для даної форми (таких, як оброблювачі подій).

CLASS

Ім'я CSS-стилю.

STYLE

Пряме включення стилю.

ID

Ідентифікатор, за яким можна посилатися на форму з мови написання сценаріїв тощо.

Елементи форми: TEXTAREA.

Тег <TEXTAREA> використовується для того, щоб дозволити користувачеві вводити більше одного рядка інформації (вільний текст). Область введення починається тегом <TEXTAREA> і закінчується тегом </TEXTAREA>.

<TEXTAREA
ROWS=«число
»
COLS=«число»
WRAP=«OFF | VIRTUAL | PHYSICAL»
ACCESSKEY=«символ»
ALIGN=«absbottom|absmiddle|baseline|bottom|left|middle|right|texttop|top»
TABINDEX=«число»
DISABLED
READONLY
NAME=«ім’я»
TITLE=«текст підказки»
LANG=«мова (ru, en, de, і т.д.)»
LANGUAGE=«Javascript | Jscript | Vbscript |
VBS» CLASS=«ім’я класу CSS»
STYLE=«безпосередній стиль»
ID=«ідентифікатор»
ONBLUR=«оброблювач події»
ONCHANGE=«оброблювач події»
ONFOCUS=«оброблювач події»
ONKEYDOWN=«оброблювач події»
ONKEYPRESS=«оброблювач події»
ONKEYUP=«оброблювач події»
ONSELECT=«оброблювач події»

□

….....
Текст, який буде показаний постійно, за замовчуванням
….....
</TEXTAREA>

Параметри тут також не є обов’язковими, але все-таки слід

указати хоча б NAME, COLS і ROWS.

Параметри:

ROWS

Кількість рядків у полі.

COLS

Кількість стовпців у полі.

WRAP

Чи розривати довгі рядки. OFF – не розривати. VIRTUAL – розривати на екрані, але всередині, залишати їх цілими. PHYSICAL – «чесно» розривати.

ACCESSKEY

Задає клавішу, яку потрібно нажати разом з Alt, щоб відразу потрапити в це поле.

ALIGN

Задає вирівнювання елемента щодо навколишнього тексту.

TABINDEX

Задає номер даного елемента в послідовності проходження елементів форми за допомогою клавіші Tab.

DISABLED

Поле не одержує користувацького введення й ні на що не реагує. Служить тільки для показу початкової (за замовчуванням) інформації.

READONLY

Схоже на попереднє, але поле одержує фокус і в ньому може перебувати курсор. Але його заборонено редагувати.

Інші параметри аналогічні відповідним параметрам тегу FORM.

Приклад використання тегу <TEXTAREA>:

```
<TEXTAREA NAME= «addr» ROWS=6 COLS=32>
Національний університет
  державної податкової
  служби України.
Кафедра ЕК.
</TEXTAREA>
```

Зверніть увагу, що усередині цього тегу усі переводи рядків викликають переводи рядків і при виведенні на екран. Для деяких браузерів (наприклад, MSIE 3.0) це правильно й для першого рядка. Тег </TEXTAREA> необхідний навіть тоді, коли поле введення спочатку порожнє.

Елементи форми: SELECT.

Тег <SELECT> використовується для організації вибору з фіксованого числа заздалегідь певних значень. За допомогою цього тегу можна представити список, список з можливістю вибрати більш, ніж один варіант і спадаюче меню. Синтаксис тегу:

<SELECT
SIZE=«число
» MULTIPLE
SINGLE
ALIGN=«absbottom|absmiddle|baseline|bottom|left|middle|right|textt
o p|top
«
ACCESSKEY=«символ»
TABINDEX=«число»
DISABLED
NAME=«ім’я»
TITLE=«текст підказки»
LANG=«мова (ru, en, de, і т.д.)»
LANGUAGE=«Javascript | Jscript | Vbscript |
VBS» CLASS=«ім’я класу CSS»
STYLE=«безпосередній стиль»
ID=«ідентифікатор»

ONBLUR=«оброблювач події»
ONCHANGE=«оброблювач події»
ONFOCUS=«оброблювач події»

□

…….....
один або кілька елементів <OPTION>
…….....
</SELECT>

Як завжди, параметри необов'язкові. Майже всі параметри мають такий зміст, як раніше. Ми розглянемо тільки нові:

SIZE

Кількість елементів списку видимих на екрані. Якщо цей параметр більше 1, то маємо справу зі списком, інакше – зі спадаючим меню.

MULTIPLE

Якщо зазначене, то користувач може вибирати кілька елементів списку одночасно.

SINGLE

Якщо зазначене, то користувач не може вибирати кілька елементів списку одночасно. Якщо не зазначене ні SINGLE, ні MULTIPLE, то мається на увазі SINGLE.

Елемент <OPTION>, згаданий при описі синтаксису, задає один елемент списку. Його формат:

<OPTION
VALUE=«значення»
SELECTED
NAME=«ім'я»
TITLE=«текст підказки»
LANG=«мова (ru, en, de, і т.д.)»
LANGUAGE=«Javascript | Jscript | Vbscript | VBS»
CLASS=«ім'я класу CSS»

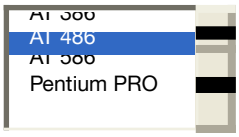
STYLE=«безпосередній стиль»
ID=«ідентифікатор»
 текст
</OPTION>

Текст, що знаходиться між <OPTION …> і </OPTION>, буде показаний користувачеві як значення даного елемента списку.

Якщо зазначений параметр SELECTED, то цей елемент буде спочатку «обраний». Інші параметри використовуються досить рідко. Параметр VALUE задає значення для даного поля «за замовчуванням». Це має сенс, якщо ви збираєтеся міняти поля за допомогою програм сценаріїв. Усі інші параметри вже були описані раніше.

Приклади різних елементів **SELECT**.

<SELECT MULTIPLE NAME=group SIZE=4>

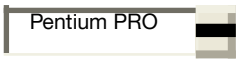


<OPTION> AT 386 </OPTION>
<OPTION SELECTED> AT 486 </OPTION>
<OPTION> AT 586 </OPTION>
<OPTION> Pentium PRO </OPTION>
<OPTION> Pentium II </OPTION>
</SELECT>

<SELECT NAME=group SIZE=4>

<OPTION> AT 386 </OPTION>
<OPTION> AT 486 </OPTION>
<OPTION> AT 586 </OPTION>
<OPTION SELECTED> Pentium PRO </OPTION>
<OPTION> Pentium II </OPTION>
</SELECT>

<SELECT NAME=group>
<OPTION> AT 386 </OPTION>



<OPTION> AT 486 </OPTION>
<OPTION> AT 586 </OPTION>
<OPTION SELECTED> Pentium PRO </OPTION>
<OPTION> Pentium II </OPTION>
</SELECT>

Елементи форми: INPUT

Тег <INPUT> визначається його параметром TYPE.

<INPUT

TYPE= «button|checkbox|hidden|image|password|
radio|reset|submit|text|textarea|file»
SIZE= «число»
MAXLENGTH= «число»
SRC= «URL»
BORDER=
«число»
CHECKED
VALUE= «значення»
ALIGN=
«absbottom|absmiddle|baseline|bottom|left|middle|right|texttop|top»
ACCESSKEY= «символ»
TABINDEX= «число»

DISABLED

NAME= «ім’я»
TITLE= «текст підказки»
LANG= «мова (ru, en, de, і т.д.)»
LANGUAGE= «Javascript | Jscript | Vbscript | VBS»
CLASS= «ім’я класу CSS»
STYLE= «безпосередній стиль»
ID= «ідентифікатор» >

TYPE

Визначає, що являє собою даний елемент. Можливі такі значення:

BUTTON – кнопка. Має сенс тільки, якщо ви обробляєте її натискання з програми сценарію	
CHECKBOX – елемент, який можна включити й виключити	Checkbox ☐

HIDDEN – Схований елемент. Його не видно, але він має значення. У такий спосіб часто передають параметри CGI програмі	Він десь тут!
IMAGE – це не просте зображення. Воно працює як кнопка SUBMIT, тобто, якщо навести на зображення мишу й нажати кнопку, інформація форми негайно передається за адресою зазначеному в ACTION. При цьому передаються й координати того місця, де була миша	
PASSWORD – поле введення пароля, символи, що вводяться, не відображаються	Пароль
RADIO – радіокнопка. Її, звичайно, поєднують у групи (група – кілька кнопок з тим самим NAME). У включеному стані завжди перебуває тільки одна із кнопок групи	Intel 80386 Intel 80486
RESET – кнопка скидання. Очищає форму	Сброс
SUBMIT – кнопка, при натисканні на яку інформація з форми передається за адресою, зазначеною в ACTION, для обробки	Готово
TEXT – просте поле для введення одного рядка тексту	
FILE – дає можливість користувачеві вибрати файл, що перебуває на його локальному диску. Надалі цей файл можна переслати на сервер. Ця можливість з’явилася в NN з версії 3, а в MSIE – з версії 4.0	

SIZE

Задає розмір видимого поля там, де це має сенс. Наприклад, для полів типу TEXT, PASSWORD, FILE тощо.

MALENGTH

Задає максимальний розмір поля там, де це має сенс. Визначає кількість символів, яку користувачі можуть ввести в поле введення. При перевищенні кількості припустимих символів брау- зер реагує на спробу введення нового символу звуковим сигна- лом і не дає його ввести. Не плутати з атрибутом SIZE. Якщо MAXLENGTH більше ніж SIZE, то в полі здійснюється скролінг. За замовчуванням значення MAXLENGTH дорівнює не скін- ченності.

SRC

Для типу IMAGE задає URL файла зображення.

BORDER

Для типу IMAGE задає розмір бордюру навколо зображення.

CHECKED

Для типів CHECKBOX і RADIO вказує, що даний елемент з самого початку «включений».

VALUE

Значення елемента. Звичайно, це текст на кнопці або початковий текст у полі введення.

ALIGN

Для типу IMAGE повністю аналогічно ALIGN у тегу <IMG …>.

Відправлення файлів на сервер

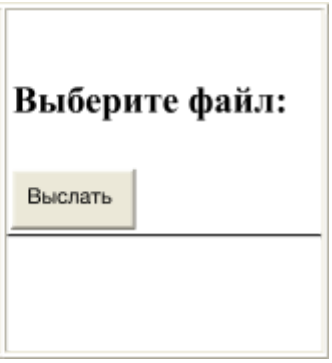
Форми можна також використовувати для відправлення файлів на сервер. Усе, що потрібно від форми, це:

- вказати ENCTYPE=«multipart/form-data» у тегу FORM;
- вказати METHOD=«POST» у тегу FORM;

- використовувати <INPUT TYPE=«FILE» ...> для введення імені файла.

Наприклад:

```
<FORM ACTION=«myURL»
ENCTYPE=«multipart/form-data»
METHOD=«POST»>
Выберите файл:<BR>
<INPUT NAME=«fn» «TYPE=«FILE»><P>
<INPUT TYPE=«SUBMIT» VALUE=«Выслать»>
</FORM>
```



Елементи діалогу (кнопки, області для введення тексту).

```
<HTML>
<HEAD>
<TITLE>Форми</TITLE></HEAD>
<BASE>
<BODY bgcolor=«silver»>
<FORM>
<CENTER><FONT size=6>Елементи діалогу</font></center>
<HR color=«blue»>
<H2>Елемент ISINDEX</h2>
<ISINDEX prompt=«Строка для введення критерію пошуку»>
<HR color=«blue»>
<H2>Елементи INPUT</h2>
<H3> Уведення текстового рядка </h3>
<INPUT type=«text» size=50>
<H3> Уведення пароля </h3>
<INPUT type=«password»>
<H3> Прапорці </h3>
<INPUT type=«checkbox» name=«F001» checked>
<INPUT type=«checkbox» name=«F001» checked>
<H3> Перемикачі </h3>
<INPUT type=«radio» name=«S001» value=«Перший»>
<INPUT type=«radio» name=«S001» value=«Другий»>
<INPUT type=«radio» name=«S001» value=«Третій» checked>
<H3> Кнопка підтвердження введення </h3>
```

<INPUT type=«submit» value=«Підтвердження»>
<H3> Кнопка із зображенням </h3>
<INPUT type=«image» src=«lycos.gif»>
<H3> Кнопка очищення форми </h3>
<INPUT type=«reset» value=«Очищення»>
<H3> Файл </h3>
<INPUT type=«file» name=«photo» accept=«image/*»>
<HR color=«blue»>
<H2>Елемент SELECT
<SELECT multiple>
<OPTION value=a>Перший
<OPTION value=Ь>Другий
<OPTION value=з>Третій
<OPTION value=d>Четвертий
</select></h2>
<HR color=«blue»>
<H2>Елемент TEXTAREA
<TEXTAREA rows=5 cols=30>

Область для введення тексту

</textarea></h2>
<HR color=«blue»>
</FORM>
</BODY></HTML>

2. CSS

Технологія каскадних таблиць стилів (Cascading Style Sheets, CSS) була розроблена як доповнення до HTML з метою визначення зовнішнього вигляду документів і збереження за HTML тільки функції структурної розмітки документів. Система CSS незалежна від HTML, має інший синтаксис і дозволяє задавати параметри графічного (також текстового і звукового) подання дескрипторів HTML.

Таблиці стилів – це набір елементів оформлення, які застосовуються до різних частин документа й описують спосіб їх подання на екрані. Таблиці стилів реалізовані у всіх функціонально-розвинених текстових процесорах.

За допомогою таблиць стилів можна форматувати текст, використовуючи методи, наближені до методів форматування звичайних друкованих сторінок. Створені сторінки будуть коректно функціонувати, враховуючи деякі обмеження, пов’язані із платформами, браузерами, різними розмірами екранів і поданнями моніторів.

Можна виділити три основні переваги застосування таблиці стилів:

1. Можна створити таблицю стилів і застосувати її до одно- го або декількох документів. Ця універсальність надає змогу змі- нювати вигляд усіх сторінок, змінюючи лише таблицю стилів у файлі.

2. Надання більшого контролю керування текстом: з’явили- ся нові властивості, за допомогою яких можна створювати спли- ваючі меню, розміщати один текст над іншим (причому незалеж- но від їх послідовності у коді HTML), створювати ефекти тіні для тексту та ін.

3. Таблиці стилів на відміну від інших методів управління зовнішнім виглядом дозволяють зберігати в різних місцях текст та інформацію про те, як він повинен виглядати, що дозволяє зменшити розміри файлів документів.

Прийняття в 1996 році Консорціумом W3C CSS першого рівня як стандарту дозволило відокремити зміст Web-сторінки

(текст, графічні зображення тощо) від її оформлення (макет сторінки й характеристики тексту, наприклад шрифти тощо). Після цього мова HTML стала функціонально-орієнтованою. Стандарт CSS2, прийнятий в 1998 році, заснований на CSS першого рівня й дозволяє розроблювачам здійснювати контроль над Web-сторінками на більш високому рівні. Він включає деякі нові функції, зокрема, можливість точно розташовувати елементи й об'єкти Web-сторінки, застосовувати шрифти, що завантажуються, або використовувати звукові таблиці стилів. Застосування стилів дозволяє також вирішити багато проблем підтримки браузерів, тому що основні компанії-розроблювачі браузерів вмонтували таблиці CSS у свої програмні продукти.

Основні властивості елементів CSS подано у таблиці 1.

Таблиця 1

Список основних властивостей елементів CSS

Властивість	**Опис**	**Приклад**
1	2	3
{background: url(картинка)}	Визначає фонову картинку, що знаходиться позаду тексту	TD {background: url(img/logo-3.gif);}
{background-color: значення кольору}	Визначає колір фону	A {background-color: #FFFFFF;}
{border-bottom-color: color} аналогічно border-top-color, border-left-color border-right-color border-color	Визначає колір нижньої (bottom), верхньої (top), лівої (left), правої (right) лінії рамки. border-color – визначає колір одразу всіх ліній	TD {border-bottom-color: #00FFFF; border-top-color: #0000FF; border-left-color: #800080; border-right-color: aqua;}
{border-bottom-style: none або solid або double або groove або ridge або inset або outset}	Визначає стиль нижньої (bottom), верхньої (top), лівої (left), правої (right) лінії рамки. border-style – визначає стиль одразу всіх ліній	TD {border-top-style: solid; }

{border-bottom-width: thin або medium або thick	Визначає ширину нижньої (bottom), верхньої (top), лівої (left), правої	TD {border-width: 3px; }

або ширина в пік-селах (дюймах, пунктах, см, мм)}	(right) лінії рамки. border-width – визначає ширину одразу всіх ліній	
{color: значення кольору}	Визначає колір перед-нього плану елемента	H1 {color: red;}
{font-family: найменування шрифту}	Визначає тип шрифту	A {font-family: Arial;}
{font-size: розмір}	Визначає розмір шрифту в абсолютних (пікселі, см та ін.) або у віднос-них величинах (%)	BODY {font-size: 5cm;}
{font-style: normal або italic}	Визначає нормальний або курсивний стиль шрифту	INPUT {font-style: italic;}
{font-weight: normal або bold або bolder або lighter}	Визначає жирність шрифту	INPUT {font-weight: bold;}
{height: число} {width: число}	Встановлює, відповідно, висоту та ширину еле-менту	TD {height: 20px;}
{margin-left: число} аналогічно: margin-right, margin-top, margin bottom, margin	Задає лівий (left), пра-вий (right), верхній (top) або нижній (bottom) відступи від краю. margin – задає величину одразу усіх відступів	BODY {margin: 0px;}
{top: число} {left: число}	Задає, відповідно, коор-динату Y та X елементу	DIV.my {top:10px;left:100px;}
{padding-left: число} аналогічно: padding –right padding –top padding –bottom padding	Задає розмір лівої (left), правої (right), верхньої (top) або нижньої (bottom) відстані між рамкою та змістом. padding – задає величи-ну одразу усіх відстаней	TD {padding-top: 10px; }

{text-align: left або right або center або justify}	Встановлює вирівнювання тексту відносно лівої та правої межі вікна або елементу	TD {text-align: center;}
{text-decoration: none або [overline або(та) underline]}	Визначає підкреслений або (та) закреслений текст, або їх відсутність	A {text-decoration: none;}

{vertical-align: top або bottom або middle}	Встановлює вирівнювання тексту відносно верхньої та нижньої межі вікна або елемента	TD {vertical-align: middle;}
{z-index: число;}	Встановлює порядок перекривання елементів	<DIV STYLE=”z-index:-1”>

Існує три способи застосування таблиць стилів у документі HTML.

- ***Вбудовування* (*Inline*),** при якому дескриптори Web-сторінки доповнюються короткими оголошеннями формату. Вбудовування надає контроль над фрагментом, до якого воно застосовується. Атрибут style приєднується до дескриптора HTML і відповідний фрагмент сторінки буде виводитися браузером із застосуванням формату, зазначеного стилем.
- ***Впровадження* (*Embed*)** забезпечує контроль над сторін- кою HTML. Використання дескриптора <style> у межах розділу
<head> дозволяє описати атрибути стилю.
- ***Зв’язування* (*Link*)** відоме також як застосування зовніш- нього аркуша стилів. У цьому випадку зв’язаний документ вико- ристовує еталону таблицю стилів, що застосовується для всього сайта (зберігається у файлі з розширенням .css).

Термін «каскадні» (таблиці стилів) описує, в першу чергу, той факт, що браузер додержується певного порядку (каскаду) при інтерпретації стилів. Можна використовувати всі три типи стилів і браузер інтерпретує їх у наступному порядку: зв’язаний – впроваджений – вбудований.

Другий аспект комбінованого застосування стилів – спадкування. Наприклад, при оголошенні в елементі абзацу деякого кольору шрифту всі елементи, що розташовані в його межах, успадкують цей колір.

Вбудовування стилю

Опис стилю можна вмонтувати в різні елементи HTML, для яких стиль має сенс, наприклад, у дескриптори абзаців, заголовків, гіперпосилань або комірок таблиці.

При вбудовуванні в елемент додається атрибут style, значенням якого є властивість і його значення, розділені двокрапкою. Таких пар («властивість: значення») може бути декілька, у цьому випадку вони розділяються крапкою з комою. Значення атрибута style беруться в лапки.

Два елементи div (розділ) і span (інтервал) є універсальними елементами-контейнерами й дозволяють застосовувати стилі до фрагментів тексту, як це показано у прикладах 1–3:

Приклади 1–3.

Приклад 1

```
<p style=«font-size: 2em; color: #ff0000»>Червоний текст із розміром шрифту 2em
</p>
<p style=«font-size: 1em; background-color: #ffff00»>Текст із розміром шрифту 1em і жовтим фоном</p>
```

Приклад 2

```
<div style=«font-size: x-small»>Текст із розміром шрифту x-small
<span style=«font-size: large»>Текст із розміром шрифту
large</span></div>
<p style=«font-size: 12pt; font-variant: small-caps»>Текст із розміром шрифту 12pt і малими прописними буквами</p>
<p style=«font-size: 16pt; font-family: Garamond, Georgia,
serif»>Текст із
розміром шрифту 16pt і Garamond або Georgia шрифтом</p>
```

Приклад 3

```
<p style=«font: bold italic 14pt Arial, sans-serif»>Текст зі
скороченою формою запису властивостей шрифту</p>
```

На рис.1 показано відображення у браузері сторінки з текстами прикладів 1–3.

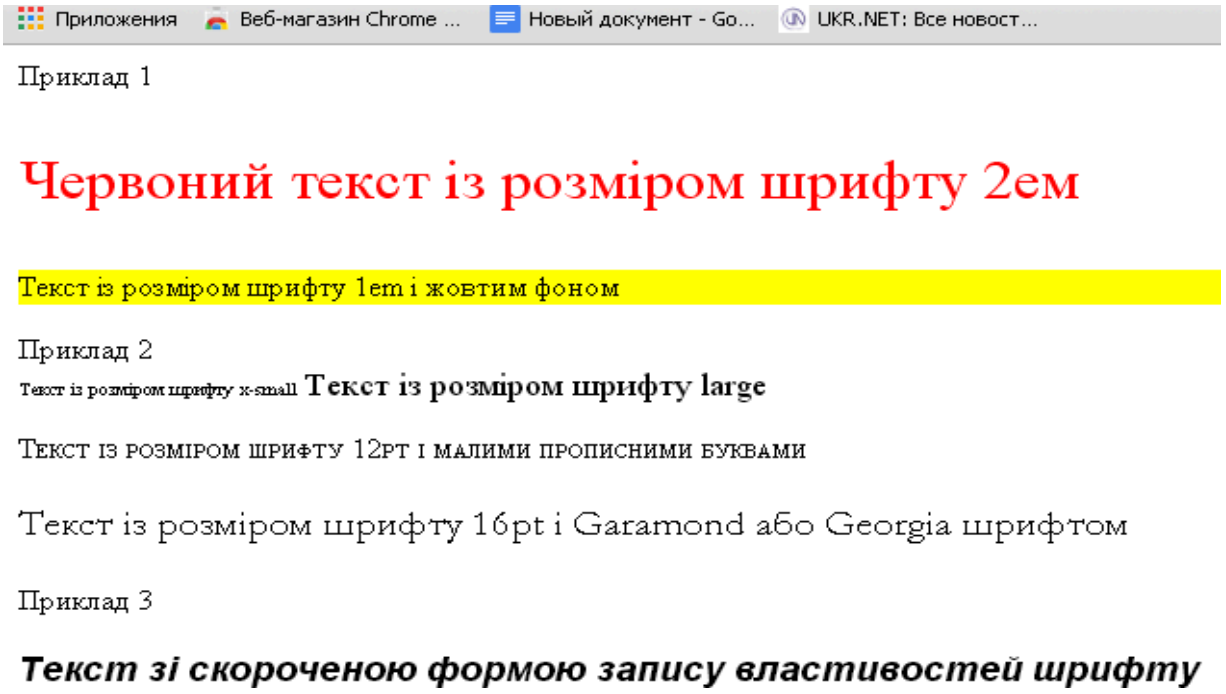


Рис. 1. Відображення сторінки з текстами прикладів 1–3

Впровадження стилю

Для впровадження стилю використовується елемент (контейнер) <style> у межах розділу <head>. Можна сховати таблиці стилів від перегляду застарілими браузерами, якщо вміст контейнера <style> укласти в коментарі (<!--…-->), як показано у прикладі 4.

У наведеному далі прикладі використані правила, за якими повинні бути представлені елементи документа. Кожне правило має дві основні частини: селектор і блок оголошень. Блок оголошень береться у фігурні дужки й може містити одне й більше оголошень. Кожне оголошення являє собою комбінацію властивості й значення, які розділяються двокрапкою. Усередині блоку оголошень може бути задано кілька оголошень, вони відділяються один від одного крапкою з комою.

Приклад 4

```
<html><head>
<title>Приклад 4</title>
<style type=«text/css»>
```

```
<!--
body {
background:#000000;color:#ffffff
}
h1 {
font:14pt Verdana; color:yellow
}
-->
</style>
</head>
<body>
<strong>Приклад 4</strong><br><br>
```

Впроваджений стиль для розділу body забезпечує виведення білих символів на чорному фоні

<h1> Для виводу заголовка використовується впроваджений стиль: шрифт Verdana жовтого кольору й розміром 14 пунктів
</h1>
</body></html>

У контейнері <style> даного прикладу знаходяться описи впроваджених стилів. Кількість стилів може бути довільною. Атрибут type указує на тип таблиць стилів, що використовується – CSS.

На рис. 2 показано відображення у браузері сторінки з текстом прикладу 4.

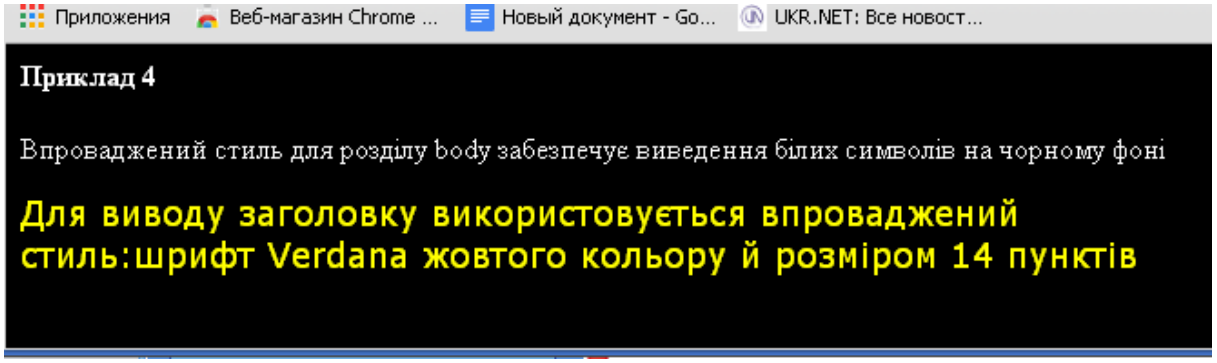


Рис. 2. Відображення в браузері сторінки з текстом прикладу 4

Використання різних селекторів для стильового оформлення.

Селектори елементів

Селектори body і h1 з прикладу 4 є селекторами елементів і визначають область застосування даного стилю. Селектор body вказує на те, що цей стиль буде застосований до розділу body, а селектор h1 на те, що стиль буде застосований до тексту усередині всіх елементів h1 документа. Таким чином, можна оголосити різні стилі для заголовків і для іншого тексту.

У прикладі 4 парами властивість-значення для селектора body є background:#000000 і color:#ffffff, а для селектора h1 − font:14pt Verdana і color:yellow.

У правилі допускається групування декількох селекторів при одному блоці оголошень стилю або ж групування властивостей і значень в оголошенні стилю.

У прикладі 5 показано, що для того, щоб додати однакові властивості заголовку h5 і абзацу p, використовується групування селекторів.

Приклад 5

```
<style type=«text/css»>
<!--
h5, p {
font-family:Garamond
; font-size:14pt;
color:#660066;}
-->
</style>
```

У прикладі 6 показано групування властивостей і значень, що належать до шрифтового оформлення.

Приклад 6

```
<style type=«text/css»>
<!--
body {
font: italic bold 15pt/18pt Verdana,sans-serif;}
```

-->
</style>

При такому групуванні повинен підтримуватися певний порядок проходження значень властивостей шрифту: нахил (зображення) шрифту, насиченість шрифту, розмір символів, коса риска, міжрядковий інтервал, сімейство шрифтів.

Селектори класів

За необхідності застосування правила стилів незалежно від елементів використовуються селектори класів. Класи дозволяють також створити кілька наборів стилів на базі одного елемента HTML. Назва класу випереджається крапкою.

У прикладі 7 клас, що дозволяє при необхідності застосувати жовтий фон до комірки таблиці (елемент td), має селектор виду: td.yellow. Клас black того ж прикладу не прив'язаний до конкретного елемента й може бути застосований до будь-якого елемен- та тексту. У даному прикладі клас black застосований для ство- рення фонів комірок таблиці в шаховому порядку. Для задання ширини й висоти таблиці застосований селектор елемента table.

Для застосування класу використовується атрибут class (приклад 7). Усі комірки, яким привласнений клас стилю yellow, будуть мати жовтий фон. Аналогічно, комірки, яким привласнений клас black, будуть мати чорний фон.

Клас black застосований також і до елемента p, текст якого розміститься на чорному фоні.

Приклад 7

```
<html><head><title>Приклад 7</title>
<style>
td.yellow {background-color:yellow;}
.black
{background-color:black;} table
{width:45%;height:30%;}
</style>
```

</head>

```
<body>
<strong>Приклад 7</strong><br>
<table border=2px>
<tr>
                                        <td class=«black»> </td>
                                        <td class=«yellow»> </td>
                                        <td class=«black»> </td>
</tr>
<tr>
                                        <td class=«yellow»> </td>
                                        <td class=«black»> </td>
                                        <td class=«yellow»> </td>
</tr>
<tr>
                                        <td class=«black»> </td>
                                        <td class=«yellow»> </td>
                                        <td class=«black»> </td>
</tr>

</table>
<p class=«black» style=«color:white;text-align:center»> До абзацу застосований клас black і вбудований стиль вирівнювання вмісту по центру з білим кольором символів</p>
</body></html>
```

На рис. 3 показано відображення тексту прикладу 7 у браузері.

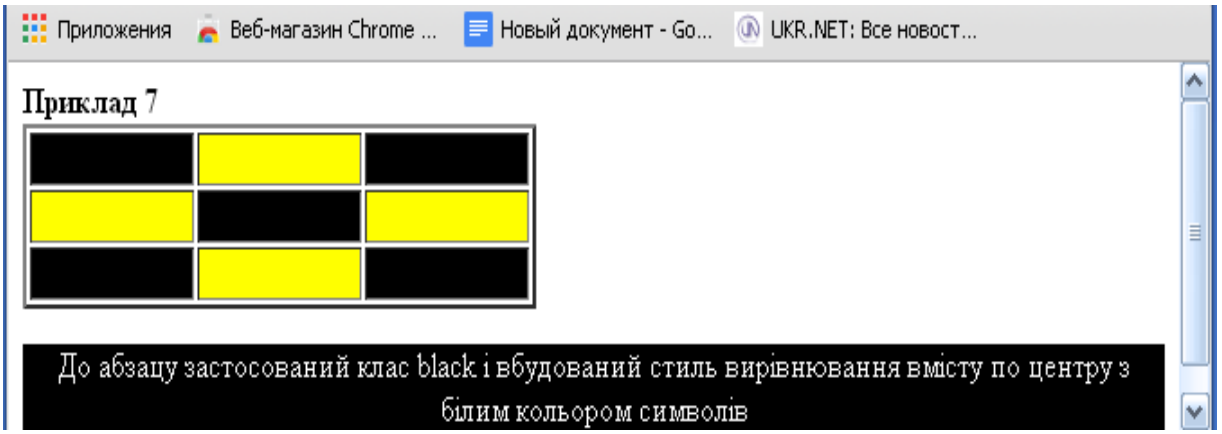


Рис. 3. Відображення тексту у браузері прикладу 7

Селектори ідентифікаторів

Селектори класів можна застосовувати до будь-якої кількості елементів. У правилі стилю можна використовувати селектор ідентифікатора (селектор ID). На відміну від селектора класу селектор ID повинен бути унікальний для даного документа. Селектор ID починається із символу #. Замість атрибута class у відповід- них елементах HTML використовується атрибут id. Його застосу- вання показано в прикладі 8. Селектор #pict використовується для позиціонування графічного зображення, яке поміщено в контей- нер .

Приклад 8

```
<html><head><title>Приклад 8</title>
<style
type=«text/css»> #pict
{ position:absolute;
top:100;
left:50;}
</style>
</head>

<body>
<strong>Приклад 8</strong>
<div id=«pict»>
<img src=«метелик.jpg» alt=«cats» width=100 height=100>
</div></body></html>
```

На рис. 4 показано відображення тексту прикладу 8 у браузері.

Рис. 4. Відображення тексту у браузері прикладу 8

Селектори нащадків

Селектори нащадків записуються через пропуск після батьківського селектора, наприклад, h1 em {color:red;}. Це правило зробить червоним контент елемента em, який є нащадком елемента h1. У селекторі нащадків може бути використано кілька елемен- тів, наприклад, ul li a {color:red;}. Правило застосує червоний колір до посилань, що перебувають в елементах li маркованого списку.

Селектори атрибутів

Селектори атрибутів можуть застосовуватися для вибору елементів на підставі їх атрибутів і значень цих атрибутів. Селектори атрибутів були введені в стандарті CSS2 і уточнювалися в стандартах CSS2.1 і CSS3.

Для вибору всіх заголовків h1, що мають атрибут class і фарбування тексту цих заголовків у сірий колір, можна використовувати правило h1[class] {color:gray} або для виділення напівжирним шрифтом текст будь-якого гіперпосилання, яке має й атрибут title і атрибут href можна записати a[title][href] {font-weight: bold}.

Вибір на підставі конкретного атрибута проводиться таким чином: a[title=«Goto GUT»] {font-weight: bold}. Напівжирним виділенням будуть відключені гіперпосилання, значенням атрибута title яких буде Goto GUT.

Можливий також вибір за частковими значеннями атрибута, наприклад, img[title~=«Малюнок»]{border:1px solid gray}. Це правило забезпечить малювання рамки навколо зображень, в атрибуті title яких міститься слово «Рисунок».

Псевдокласи й псевдоелементи

У **CSS2.1** визначені псевдокласи й псевдоелементи. Традиційно псевдокласи використовуються для оформлення гіперпосилань. Вони описують властивості посилань невідвідуваних **(:link**), відвідуваних **(:visited**), над якими перебуває курсор миші (тобто властивості посилань «при наведенні» курсору миші) (:hover) і активних посилань (:active).

У прикладі 9 наведений код **HTML**, що застосовує псевдокласи для елемента <**a**>. У даному прикладі невідвідувані посилання мають чорний колір, активні посилання фарбуються в синій, відвідувані – у зелений, а при наведенні миші на посилання колір її міняється на червоний і розмір шрифту збільшується до 16 пунктів (рис. 5). Порядок розташування псевдокласів у селекторі важливий, тому що при його порушенні принцип каскадності застосування стилів може привести до ігнорування властивостей окремих псевдокласів.

Приклад 9

```
<html><head><title>Пример 9</title>
<style type=«text/css»>
<!--
a:link {color:black;}
a:visited {color:#00ff00;}
a:hover
{color:#ff0000;font-size:16pt;}
a:active {color:#0000ff;}
-->

</style>
</head>
<body>
<strong>Приклад 9</strong><br>
<a href=«класи_1.html» > Перехід до файла «класи_1.
html»</a><br>
<a href=«класи.html» > Перехід до файла «класи.html»</a>
</body></html>
```

На рис. 5 показано відображення у браузері результату дії псевдокласу: hover – наведення миші на друге гіперпосилання.

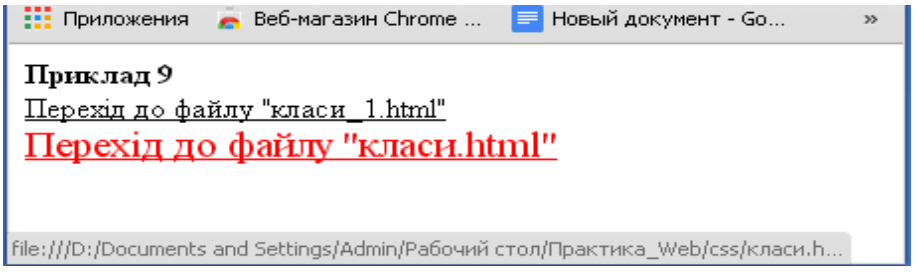


Рис. 5. Відображення в браузері результату дії псевдо-

класу: hover

До псевдокласів відносять також**:focus**. Цей псевдоклас ставиться до будь-якого елемента, до якого в даний момент належить фокус вводу, тобто який готовий приймати введення з клавіатури або може бути активований іншим способом.

Псевдоелементи вводять фіктивні елементи в документ, щоб досягти певних ефектів. У **CSS2.1** визначено чотири псевдоелементи, для яких можуть бути застосовані спеціальні стилі: перша буква (**:first-letter**), перший рядок (**:first-line**), до (**:before**) і після (**:after**) елемента.

Наприклад, правило **h3:first-letter {font-size: 150 %}** збільшить першу букву заголовка в півтора раза стосовно іншого тексту. А правило **p:first-line {color:red}** зробить символи першого рядка абзацу червоними.

Приклади 10 і 11 ілюструють використання псевдоелементів**:first-letter** і**:first-line.**

Приклад 10

```
<html><head>  <title>Приклад 10</title>
<style
type=«text/css»>
p:first-letter{
font-size:3em;
font-weight:bold
; color:#ff0000;}
p {line-height:1em;text-align:justify;}
</style>
</head>

<body>
<strong>Приклад 10</strong>
<p>Псевдоелемент first-letter використовується для створення бук- виц в абзаці. Абзац має буквицу розміру 3ем і накреслення bold.</p>
<p style=«text-indent:200px»>Абзац має буквицу розміру 3 ем, накреслення bold і відступ першого рядка 200px. Абзац має буквицу розміру 3 ем, накреслення bold і відступ першого рядка 200px.</p>
```

</body></html>

На рис. 6 показано відображення прикладу 10 у браузері.

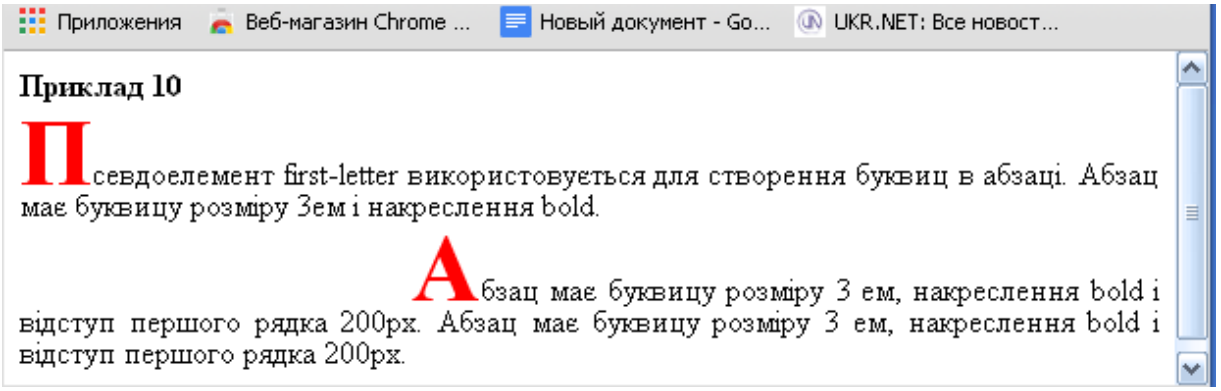


Рис. 6. Відображення у браузері прикладу 10

Приклад 11

```
<html><head>  <title>Приклад 11</title>
<style
type=«text/css»>
p:first-line{
font-weight:bold;}
p {line-height:1em;text-align:justify}
</style>
</head>
<body>
<strong>Приклад 11</strong>
```

Перші рядки абзаців стилізовані за допомогою псевдоелементу p:first-line. Перші рядки абзаців стилізовані за допомогою псевдокласу p:first-line.

 Перші рядки абзаців стилізовані за допомогою псевдоелементу p:first-line. Перші рядки абзаців стилізовані за допомогою псевдокласу p:first-line.

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

На рис. 7 показано відображення прикладу 11 у браузері.

Псевдоелементи**: before** і **:after** дозволяють вставляти *зге-нерований вміст,* а потім застосовувати до нього спеціальні стилі. Цей вміст вставляється за допомогою властивості **content**, наприк- лад, **h2:before {content: «ГУТ»}** додає перед заголовками друго- го рівня текст «ГУТ» або для **body:after {content: «The end»}** до- дає в кінець документа текст «The end».

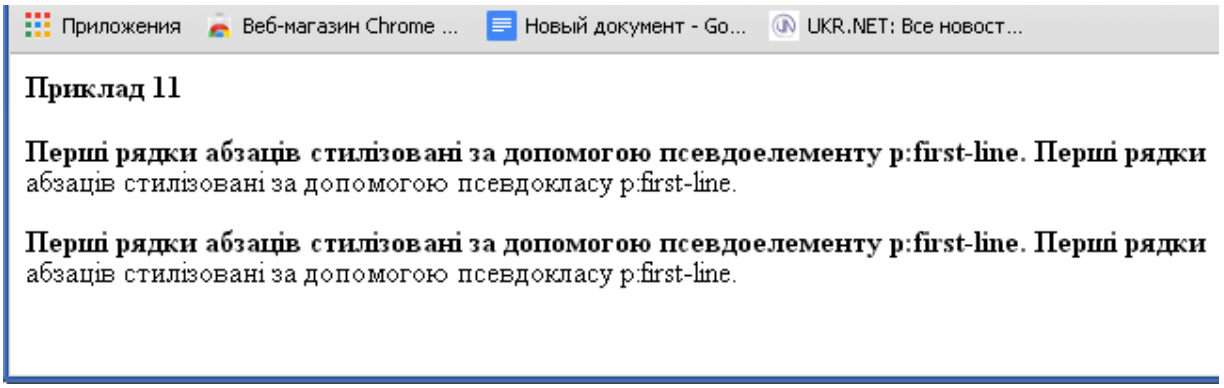


Рис.7. Відображення у браузері прикладу 11

Зв'язування стилю

При зв'язуванні стилю всі стильові описи проводяться за тикими правилам, що й при впровадженні, але зберігаються вони в окремому файлі, що має розширення .**css**. Сам файл повинен перебувати в кореневому каталозі сайта, а якщо ні, то потрібно коректно вказати зв'язок з ним.

Перевагою зв'язування є те, що стильові описи застосовуються до всіх сторінок сайта й за необхідності зміни оформлення досить внести нововведення у файл, що містить ці описи, замість того, щоб виправляти всі сторінки сайта.

У наведеному нижче прикладі ілюструється використання зовнішньої таблиці стилів.

Приклад. Файл із іменем **my_style.css** містить *тільки* правила стилів:

```
body {
background:
#000000; color:
#ffffff;
}
a {
color: #ff0000; text-decoration:none;
}
```

Документ **HTML**, який посилається на цей файл, повинен містити в частині **head** наступне посилання, що використовує елемент **link**:

<html><head>

<link rel=stylesheet href=«my_style.css» type=«text/css»>
</link>

Аналогом елемента **link** є директива **@import url (url)**. Директива **@import** також вказує браузеру на необхідність завантаження зовнішньої таблиці стилів. Розміщається директива в контейнері **<style>** перед іншими правилами **CSS**. Приклад використання директив **@import** :

<style>
@import url («sheets22.css»);
@import url («http://example.org/library/layout.css»)**;**
@import url («printer.css») print;
</style>

Із прикладу видно, що може бути використано кілька директив **@import** для приєднання декількох файлів з таблицями стилів. В останній директиві зазначене обладнання, для якого створена приєднана таблиця.

Текстові властивості CSS

1. **text-align –** вирівнювання тексту із значеннями: **left** (по лівому краю), **right** (по правому краю), **center** (по центру), **justify** (по обох краях).
2. **text-indent –** відступ у першому рядку блоку (абзацу, роз- ділу) зі стандартними значеннями довжини (**pt, px, cm, mm**).
3. **text-decoration –** оформлення тексту підкресленням зі значеннями: **none** (відсутнє – за замовчуванням), **underline** (підкреслення), **overline** (лінія над текстом), **line-through** (підкреслення), **blink** (мерехтіння).
4. **text-transform –** переклад букв у верхній або нижній ре- єстр зі значеннями: **none** (відсутнє – за замовчуванням), **capitalize** (перша буква кожного слова стає прописною), **uppercase (**пере- водить усі букви у верхній реєстр), **lowercase** (переводить усі бук- ви в нижній реєстр).
5. **text-shadow** – установка ефекту затемнення тексту зі зна- ченнями: **none** (відсутнє – за замовчуванням), **color left top radius** (колір затемнення з відстанями ліворуч (праворуч), униз (нагору) від тексту й радіусом нерізкості).

6. **letter-spacing** – відстань між символами тексту зі стандарт- ними значеннями довжини (**pt, px, cm, mm, em,ex**).

7. **word-spacing** – відстань між словами зі стандартними значеннями довжини **(pt, px, cm, mm**).

Крім стандартних одиниць виміру довжини **px, pt** (дорівнює 0,35 мм), **mm, cm**, може використовуватися **em** (**1em** дорівнює ширині букви **m** у шрифті заданого розміру), **ex** (**1ex** дорівнює висоті шрифту), **px** (**1px** дорівнює ширині пікселя).

Приклад 12

```
<html><head><title>Приклад 12</title>
<style type=«text/css»>
<!--
p.class1 {
text-align:
justify;
text-indent:  20pt;
color: #c0c0c0;
background-color: #000000;
}
.class2 {
text-decoration: underline;
}
.class3 {
letter-spacing: 0.5em;
}
.class4 {
text-transform: uppercase;
}
-->
</style>
</head>
<body>
<strong>Приклад 12<b/>
```

Текст абзацу вирівнюється по ширині,
має відступ першого рядка 20 пунктів і використовує
білий колір символів на чорному фоні
У тексті абзацу використовується
підкреслення,
 збільшення відстані між символами
, а також відображення символів
у верхньому регістрі
</body></html>

На рис. 8 показано відображення тексту прикладу 12 у браузері.

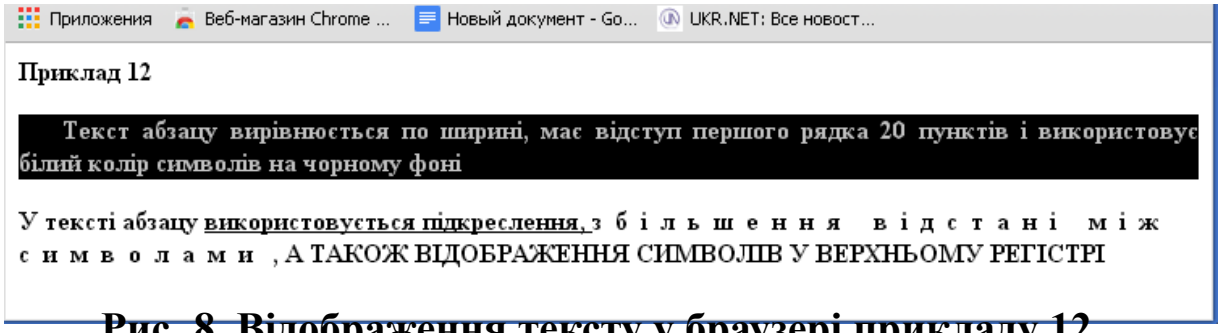


Рис. 8. Відображення тексту у браузері прикладу 12

Колір і фон

1. **color** – колір тексту – будь-яке значення, що відповідає стандарту.
2. **background-color** – колір фону – будь-яке значення, що відповідає стандарту.
3. **background-image:URL(«URL») –** фонове зображення – будь-яке значення **URL**, що відповідає стандарту.
4. **background-repeat –** напрям повторення фонового зображення зі значеннями: **repeat** (повторення по горизонталі й вертикалі – за замовчуванням), **repeat-x** (повторення тільки по горизонталі), **repeat-y** (повторення тільки по вертикалі), **no-repeat** (відсутність повторення).
5. **background-position** – положення фонового зображення щодо верхнього лівого кута утримуючого його елемента зі зна- ченнями: відстань по горизонталі, відстань по вертикалі в стан- дартних одиницях довжини або в процентному співвідношенні.

Значення 50 % по горизонталі й 50 % по вертикалі дає розташування фонового зображення по центру. Розміщення фонового зображення може бути також **top (**по верхньому краю), **center** (по центру), **bottom** (по нижньому краю), **left** (по лівому) і **right** (правому) краях.

6. **background-attachment** – прокручування фонового зображення разом з документом зі значеннями: **scroll** (прокручування – за замовчуванням) або **fixed** (фіксація) зображення у вікні браузера.

7. Для опису одразу всіх параметрів фону можна використо- вувати властивість background: background-color background- image background-repeat background-attachment background- position

Приклад 13_1

```
<html><head><title>Приклад 13_1 </title>
<style>
p {background-image:
    url(fox.jpg);
    background-position:center;
    border: 1px dotted gray;}
p.c1 {background-repeat:
repeat-y;} p.c2
{background-repeat: repeat-x;}
</style>
</head>

<body><strong>Приклад 13_1</strong>
      <p class=«c1»> У даному абзаці фонове зображення позиціоноване по центру і поширюється по вертикалі. </p>
       <p class=«c2»> У даному абзаці фонове зображення позиціоноване по центру і поширюється по горизонталі. </p>
       </body></html>
```

На рис. 9 показано відображення тексту прикладу 13_1 у браузері.

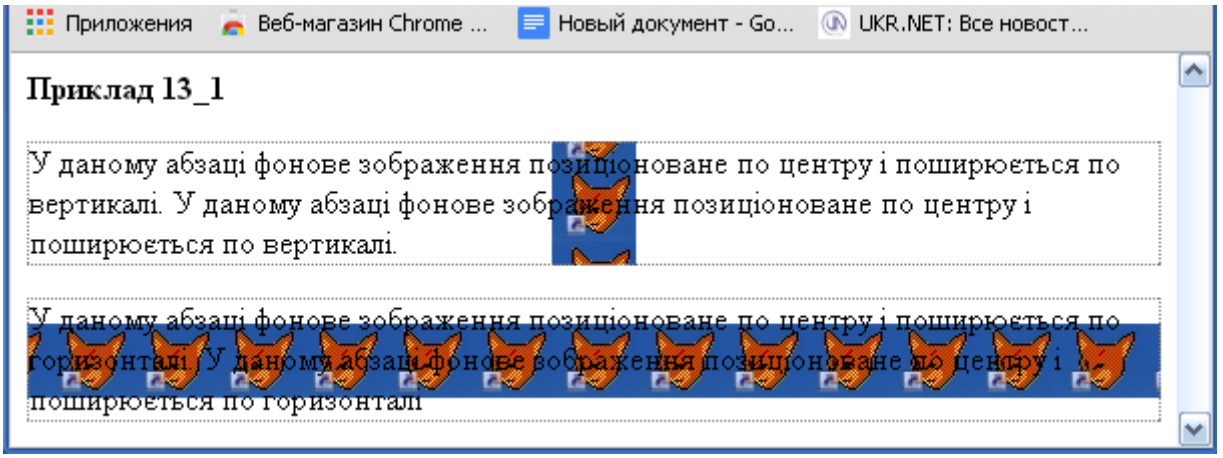


Рис. 9. Відображення тексту у браузері прикладу 13_1

У прикладі 13_2 фонове зображення позиціоноване на 25 % по горизонталі й, за замовчуванням, по центру по вертикалі.

Приклад 13_2

```
<html><head><title>Приклад 13_2</title>
<style>
p {background-image:
    url(fox.jpg);
    background-position:25%;
    background-repeat: no-repeat;
    border: 2px dotted gray;}
</style>
</head>

<body>
<strong>Приклад 13_2</strong><br>
    <p>У даному абзаці фонове зображення позиціоноване на 25 % по горизонталі і по центру по вертикалі. </p>
</body></html>
```

На рис. 10 показано відображення тексту прикладу 13_2 у браузері.

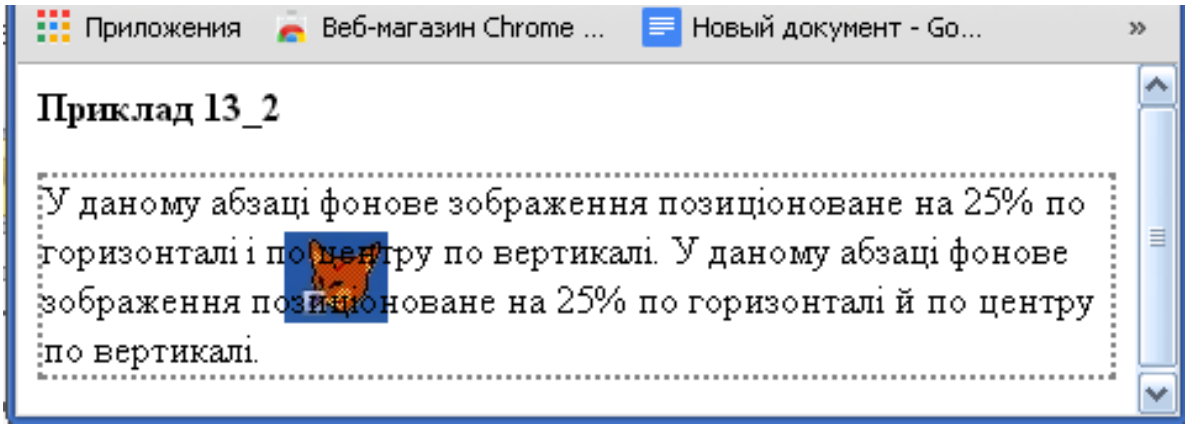


Рис. 10. Відображення тексту у браузері прикладу 13_2

Шрифти

1. **font-family** – сімейство шрифтів. Може бути кілька сімейств, відділених один від одного комами. Пріоритет визначається порядком у списку. Значенням може бути ім'я типового шрифту (**serif/sans-serif/cursive/fantasy/monospace**), а також шрифту, що належить одному з типів.

2. font-style – зображення шрифту зі значеннями: **normal** (звичайне за замовчуванням), курсив (**italic)**, похиле зображення (**oblique).**

3. **font-variant** – у вигляді малих прописних букв зі значеннями: **normal** (звичайне – за замовчуванням) або **small-caps** (у вигляді малих прописних букв).

4. **font-weight** – товщина виведеного шрифту зі значеннями: **normal** (звичайна – за замовчуванням), **bold** (напівжирний), **bolder** (жирний), **lighter** (світлий) або числовими значеннями: 100-світлий, 400-звичайний, 700-напівжирний, 900-жирний.

5. **font-size** – висота символів (кегль). Значенням є будь-яка відповідна до стандартів висота або процентне значення, що позначає зменшення або збільшення у відсотках від кегля батьківського елемента. Значення абсолютного розміру можуть бути записані у вигляді ключових слів: **xx-small** (занадто малий), **small** (малий), **medium** (середній – за замовчуванням), **large** (великий), **x-large** (дуже великий), **xx-large** (занадто великий).

Значення відносного розміру можуть бути записані у вигля-

ді ключових слів: **larger** (більше) і **smaller** (менше).

6. Для опису одразу всіх параметрів шрифту можна використовувати властивість **font: font-style font-variant font-weight font-size font-family**

Використання різних властивостей шрифтів наведено в прикладі 14.

Приклад 14

```
<html><head><title>Приклад 14 </title>
<style
type=«text/css»> p {
font: oblique 18pt Impact;}
p1 {font: large Verdana, sans serif;
color:#0000ff;}
</style>
</head>
<body>
<strong>Приклад14</strong>
<p> Для абзацу застосовано шрифт Impact стилю oblique,
18 пунктів</p>
<p class=«p1»> Для абзацу застосовано шрифт Verdana, large</p>
</body></html>
```

На рис. 11 показано відображення тексту прикладу 14 у браузері.

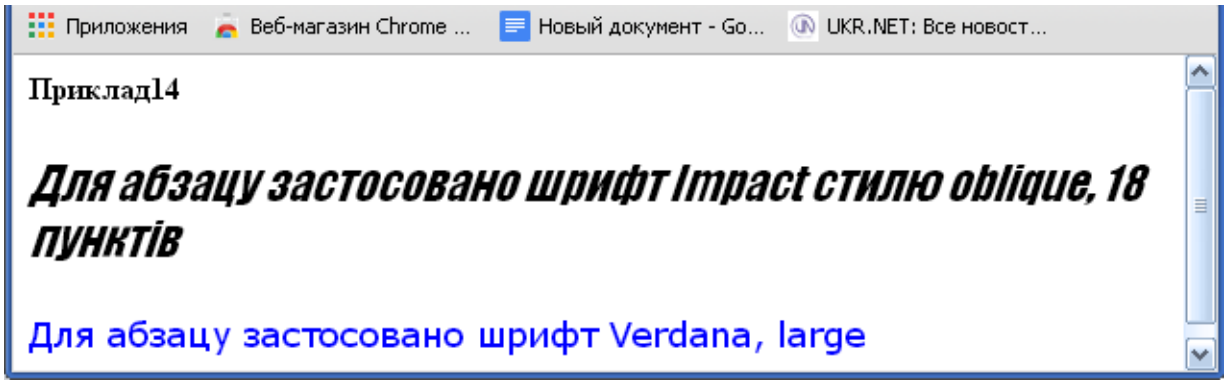


Рис. 11. Відображення тексту у браузері прикладу 14

Блокова модель

1. **margin-top, margin-right, margin-bottom, margin-left** – ширина верхнього, правого, нижнього й лівого поля. Значення за замовчуванням – 0, значеннями є довжини, відповідні до стандарту, або процентне значення, що визначає відношення ширини поля до ширини елемента.

2. **margin** – ширина полів для всіх сторін елемента. У цієї властивості може бути від 1 до 4 значень. Одне значення привласнюється усім полям, із двох значень перше привласнюється *верхньому* й *нижньому* полю, а друге – *лівому* й *правому*. При трьох: перше – верхньому полю, друге – лівому й правому, а третє – нижньому.

3. **padding-top, padding-right, padding-bottom, padding-left** – ширина проміжку між вмістом елемента й певною ділянкою його границі. Значення за замовчуванням – 0, значеннями є довжини, відповідні до стандарту, або процентне значення, що визначає ві- дношення ширини проміжку до ширини елемента.

4. **padding –** ширина проміжку для всіх сторін елемента. У цієї властивості може бути від 1 до 4 значень. Одне значення привласнюється всім проміжкам, із двох значень перше привласнюється *верхньому* й *нижньому* проміжку, а друге – *лівому* й *правому*. При трьох: перше – верхньому полю, друге – лівому й правому, а третє – нижньому.

5. border – рамка. Для опису всіх властивостей рамки мож- на використовувати конструкцію: **border: border-width border- style border-color.**

Властивостями є **border-width** (ширина рамки), **border-style** (стиль рамки) і **border-color** (колір рамки).

5.1. **border-width –** ширина рамки. Значеннями є: **thin** (тонка лінія), **medium** (середня – за замовчуванням), **thick** (товста), а також стандартні значення ширини.

Можна також встановлювати значення ширини рамки для певної сторони. Для цього використовуються властивості **border-top-width, border-right-width, border-bottom-width, border-left-**

width з аналогічними значеннями.

5.2. **border-style –** стиль рамки. Значеннями є: **none** (відсутність – за замовчуванням), **hidden** (схована), **dotted** (пунктир), **solid** (суцільна), **double** (подвійна), **dashed** (штрих-пунктир), **groove** (подвійна борозна), **ridge** (гребінь), **inset** (врізання), **outset** (орнамент).

Можна також установлювати значення стилю рамки для певної сторони. Для цього використовуються властивості **border-top-style, border-right-style, border-bottom-style, border-left-style**.

5.3. **border-color –** колір рамки. Значенням є будь-яке, що відповідає стандарту.

Можна також установлювати значення кольору рамки для певної сторони. Для цього використовуються властивості **border-top-color, border-right-color, border-bottom-color, border-left-color.**

Приклад 15

```
<html><head><title>Приклад 15</title>
<style
type=«text/css»> p {
border: 10px double red;
margin: 50px 100px 100px 50px;
padding:20px }

</style>
</head>
<body>
<strong>Приклад15</strong>
<p>Текст абзацу поміщений у подвійну рамку червоного кольору, товщиною 10 пікселів. Установлені поля шириною 100 пікселів праворуч і знизу, зверху й ліворуч – 50 пікселів , а проміжок між текстом абзацу і рамкою встановлено 20 пікселів з усіх сторін</p>
</body> </html>
```

На рис. 12 показано відображення тексту прикладу 15 у браузері.

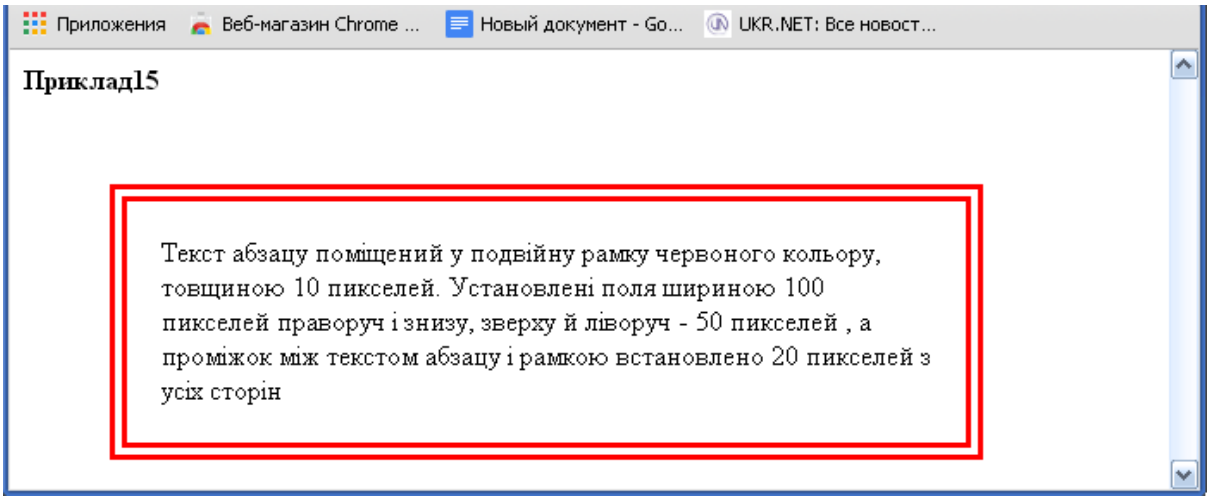


Рис. 12. Відображення тексту у браузері прикладу 15

Плаваюча модель для розміщення зображень й інших елементів

До появи останніх стандартів **CSS** для обтікання текстом зображень користувалися атрибутом **align** елемента **img** зі значеннями **left** і **right**. Зараз стандартизована властивість **float** зі значеннями **left** і **right**, яка може бути застосована не тільки до зображень, але й до інших елементів **HTML**. У комбінації із властивістю **float** часто застосовують властивість **clear** зі значеннями **left** і **right** або **both** для того, щоб звільнити місце ліворуч або (і) праворуч від інших елементів Web-сторінки.

У прикладі 16_1 показано використання плаваючої моделі для зображення.

Приклад 16_1

```
<html><head>  <title>Приклад 16_1</title>
            <style>
            .leftfloat {float:left}
            .rightfloat {float:right}
            </style>
</head>
<body>
<strong>Приклад 16_1</strong><br>
<img src=«метелик.jpg» class=«leftfloat» alt=«flax» width= «80px»
```

height=«80px» border=«1px»>

 Для обтікання текстом зображення ліворуч замість атрибута align тегу img використовується властивість float зі значенням left. Для обтікання текстом зображення ліворуч атрибута align тегу img використовується властивість float зі значенням left. Для обтікання текстом зображення ліворуч замість атрибута align тегу img використовується властивість float зі значенням left.

<img src=«метелик.jpg» class=«rightfloat» alt=«flax» width= «80px» height=«80px» border=«1px»>

 Для обтікання текстом зображення праворуч замість атрибута align тегу img використовується властивість float зі значенням right. Для обтікання текстом зображення праворуч атрибута align тегу img використовується властивість float зі значенням right. Для обтікання текстом зображення праворуч атрибута align тегу img використовується властивість float зі значенням right.

</body></html>

На рис. 13 показано відображення тексту прикладу 16_1 у браузері.

Рис. 13. Відображення тексту у браузері прикладу 16_1

У прикладі 16_2 показано використання плаваючої моделі для абзацу із цитатою.

Приклад 16_2

<html><head><title>Приклад 16_2</title>

```
<style>
p
{font-family:Garamond;
text-align:justify;
margin-left:0.25em;}
.clearp
{clear:left;}
p.Floatl  {float:left;
width:45%;
text-align:center;
margin:0.5em 0.25em;
padding:0.25em;
border-top:1.3em solid #999 ;
border-bottom:1.3em solid #999 ;
background-color:black;
color:white;
font-size:1.3em;
font-family:Garamond;
}
</style>
</head>
<body>
<strong>Приклад 16_2</strong>
<p>У прикладі використовується обтікання текстом абзацу цитати. Властивості цитати записана в класі .Floatl</p>
<p class=«Floatl»>«Фантазія важливіше знань» А.Ейнштейн </p>
    <p>Обтікання цитати, укладеної в абзац, забезпечується властивістю float. Для елементу обтікання повинна бути обов’язково задана його ширина, крім того, в прикладі зазначені властивості margin і padding, рамки сірого кольору зверху і знизу, а також властивості шрифту: сімейство, колір символів, колір фону і розмір. Обтікання цитати, укладеної в абзац, забезпечується властивістю float. Для елементу обтікання повинна бути обов’язково задана його ширина, крім того, в прикладі зазначені властивості margin і padding, рамки сірого кольору зверху і знизу, а також властивості шрифту: сімейство, колір символів, колір фону і роз-
```

мір</p>
</body></html>

На рис. 14 показано відображення тексту прикладу 16_2 у браузері.

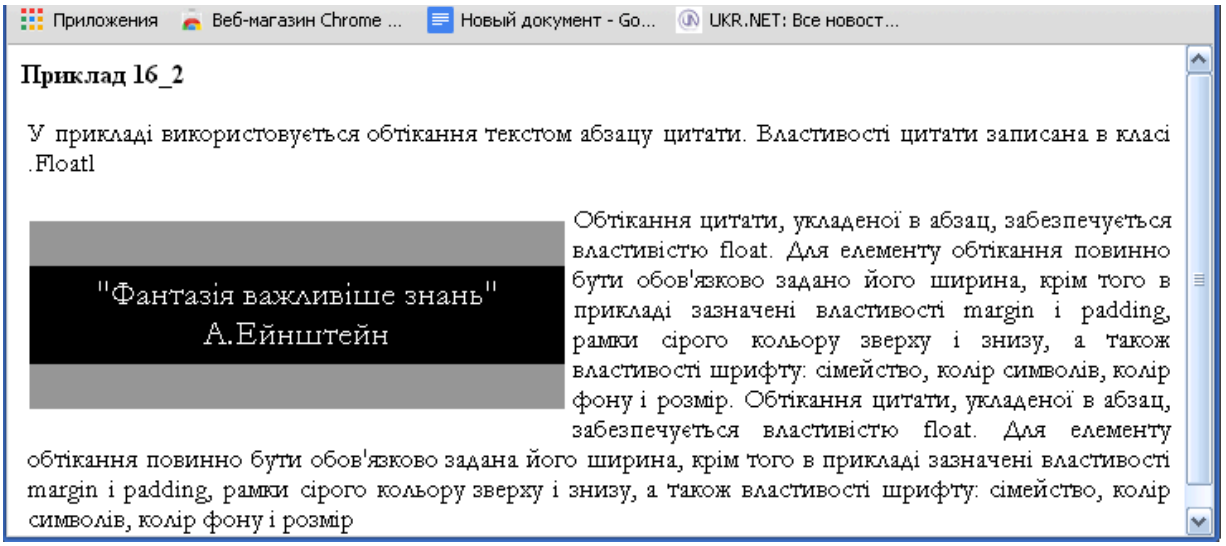


Рис. 14. Відображення тексту у браузері прикладу 16_2

У прикладі 16_3 показано використання обтікання для створення триколонного документа із вставленою в одному зі стовпчиків цитатою.

Приклад 16_3

```
<html><head><title>Приклад 16_3</title>
<style type=«text/css»>
#header {color: white;
background-color:
#666;
border-bottom:1px solid
#333; text-align: center;
padding: 0.1em 0;
margin: 0 0 1em 0}

#leftcol {width: 25%;
float:left;
background: white;
padding-bottom:
1em; text-align: left;
}
```

```
#rightcol {width: 25%;
```

```
float:left;
background: white;
padding-bottom:
1em; text-align: left;
}
#centercol
{
width:43%;
float: left;
background:
white; padding: 0
1em;
border-left: 1px solid #333;
border-right: 1px solid
#333
}
#cite {
float:right;
width:90%
;
border: 1px solid
red; padding:0.1em;
margin: 0 0.1em;
font-size:larger;
text-align:center;
}
#footer {
clear: both;
padding-bottom: 1em;
border-top: 1px solid
#333; text-align: center;}
</style>
</head>
<body>
<div id=«header»>
```

<h3>Технологія каскадних таблиць стилів (Cascading Style Sheets, CSS)</h3>

Таблиці стилів – це набір елементів оформлення, які застосовуються до різних частин документа і описують способи їх представлення на екрані. Таблиці стилів реалізовані у всіх функціонально-розвинених текстових процесорах

Прийняття в 1996 році Консорціумом W3C CSS першого рівня як стандарту дозволило відокремити зміст Web-сторінки (текст, графічні зображення тощо) від її оформлення (макет сторінки і характеристики тексту, наприклад, шрифти, колірне оформ- лення тощо). Після цього мова HTML знову стала функціонально- орієнтованою, а не орієнтованою на форму. Стандарт CSS2, прий- нятий в 1998 році, заснований на CSS першого рівня дозволяє ро- зробникам здійснювати контроль над Web-сторінками на більш високому рівні. Він включає деякі нові функції, зокрема, можли- вість точно розташовувати елементи й об’єкти Web-сторінки, за- стосовувати шрифти, що завантажуються, або використовувати звукові таблиці стилів. Застосування стилів дозволяє також розв’язати багато проблем підтримки браузерів, тому що основні компанії-розроблювачі браузерів вмонтували таблиці CSS у свої програмні продукти.

 За допомогою таблиць стилів можна форматувати текст, використовуючи методи, наближені до методів форматування звичайних друкованих сторінок. Створені сторінки будуть коректно функціонувати, враховуючи деякі обмеження, пов’язані із плат- формами, браузерами, різними розмірами екранів і дозволами моніторів. Процес розробки Web-стандартів незабаром обіцяє розв’язати ці проблеми

«Мистецтво – це відображення світу в образах, а наука – відображення світу в поняттях» Н. К.

Консорціум W3C CSS

</body></html>

На рис. 15 показано відображення тексту прикладу 16_3 у браузері.

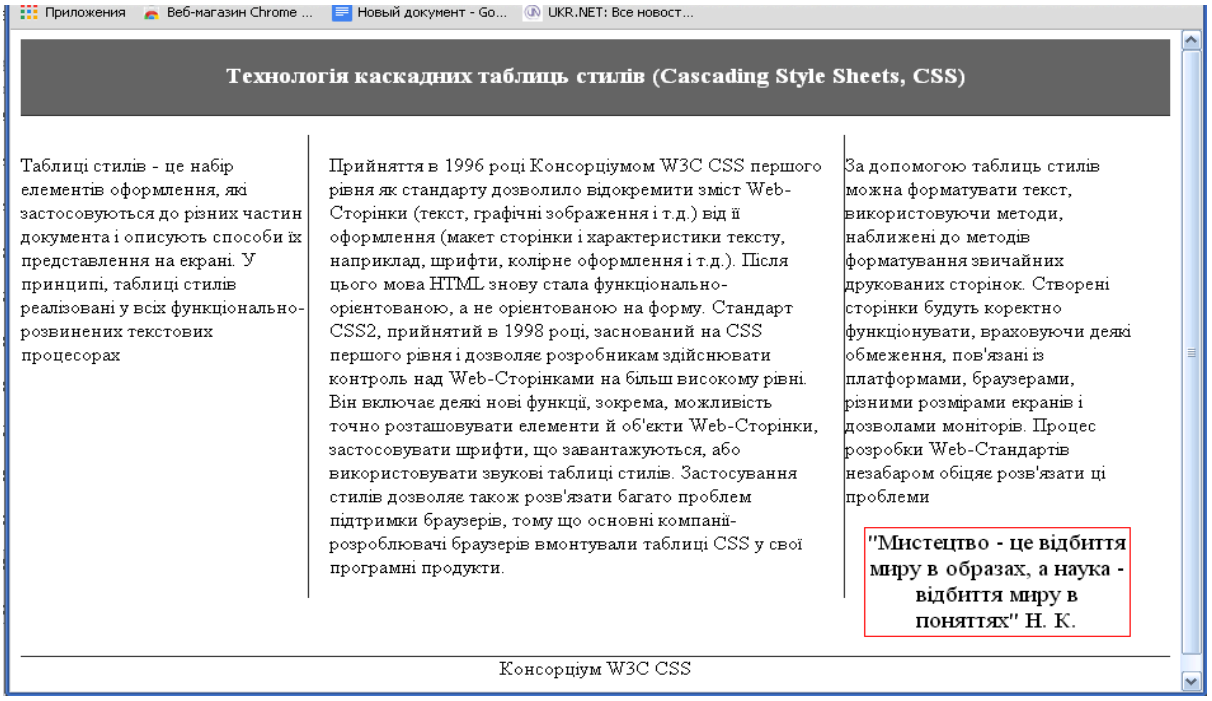


Рис. 15. Відображення тексту у браузері прикладу 16_3

Позиціонування й візуалізація

Застосування позиціонування припускає використання ряду понять:

Обмежена область (блок-контейнер) – невидима прямокутна область, що визначена браузером. Таблиці стилів дозволяють управляти цією областю, установлюючи її положення на сторінці з використанням абсолютних або відносних значень позиціонування.

Абсолютне позиціонування – технологія, що дозволяє задавати координати блоку щодо його блоку-контейнера, яким може бути інший (батьківський) елемент документа або початковий блок-контейнер – прямокутник, відповідно до вікна перегляду браузера. При цьому блок елемента повністю видаляється з потоку документа й позиціонується щодо його блок-контейнера.

Відносне позиціонування – технологія, що дозволяє задавати координати блоку щодо його незміщеного положення в потоці документа. При відносному позиціонуванні елемент зрушується зі свого звичайного місця, але простір, який він повинен був займати, не зникає.

Фіксоване позиціонування – технологія, що дозволяє задавати координати блоку щодо початкового блоку-контейнера (вікна перегляду браузера).

1. **position** – метод позиціонування блоку за замовчуванням має значення **static**, іншими значеннями є **absolute** (абсолютне) і **relative** (відносне) позиціонування, а також значення **fixed**, при якому позиціонування блоку є зсувом, як у випадку абсолютного позиціонування, але блок фіксується у вікні браузера й не пере- міщається при прокручуванні вікна.

2. **top** – величина зсуву вниз від верху його блок-контейне- ра, **bottom** – нагору від нижньої сторони блок-контейнера, **left** – відповідно, уліво, а **right** – вправо від лівої й правої сторін блок- контейнера. Значеннями є будь-які відповідні до стандарту дов- жини, а також процентне значення: відношення у відсотках дов- жини зсуву до ширини (висоти) блоку, а також припустимі нега- тивні значення зазначених властивостей зсуву.

3. **width** – ширина блоку. Значеннями є будь-які відповідні до стандарту довжини, а також процентне значення: відношення у відсотках довжини зсуву до ширини вікна.

4. **height** – висота блоку. Значеннями є будь-які відповідні до стандарту довжини, а також процентне значення: відношення у відсотках довжини зсуву до висоти вікна.

5. **z-index – z-індекс** визначає порядок розташування блоків. Значеннями є цілі числа (позитивні й негативні), причому блоки з більшими значеннями z-індексу будуть з’являтися над блоками з меншими значеннями.

6. **visibility** – видимість. Визначає, чи є елемент видимим – **visible** або прихованим – **hidden**.

7. **overflow** – керування переповненням. Має три значення: **visible** (елемент видний), **hidden** (частина, що перекривається, ві- дтинається)**, scroll** (використовується механізм прокручування для візуалізації елемента).

8. **clip: rect (top right bottom left) –** відсікання. Визначає об- ласті, що вирізаються.

9. **display –** подання елемента зі значеннями none, inline,

block, list-item, table, inline-table.

Приклад 17_1 ілюструє основні варіанти позиціонування

```
<html><head><title>Приклад 17_1</title>
<style type=«text/css»>
<!--
p, h5 {margin:0;padding:0}
-->
</style>
</head>
<body>
<strong>Приклад 17_1</strong>
<div style=«background-
color:gray;position:absolute;top:70;left:50;color:blue;»
>
<h5>relative</h5>
<p>Екб екб екб екб</p>
<p
style=«position:relative;top:70;left:50;color:red»>
Екб екб екб екб</p>
</div>
<div style=«background-
color:gray;position:absolute;top:70;left:250;color:blue»
>
<h5>normal stream</h5>
<p>Екб екб екб екб</p>
<p style=«color:red»>Екб екб екб екб</p>
</div>

<div style=«background-color:gray;position: absolute;top: 70;left:450;
color:blue»;>
<h5>absolute</h5>
<p>Екб екб екб екб</p>
<p style=«position:absolute;top:70;left:50;color:red»>Екб екб екб
екб</p>
</div>
</body></html>На рис. 16 показано відображення тексту прикла-
```

ду 17_1 у браузері.

Рис. 16. Відображення тексту у браузері прикладу 17_1

У випадку відносного позиціонування текст червоного кольору на **top:70** і **left:50** від свого незміщеного положення в потоці елементів, а у випадку абсолютного – від батьківського елемента, яким є прошарок **div**.

У прикладі 17_2 показано використання **абсолютного** позиціонування для відображення двох прошарів. Вертикальний прошарок кольору **deeppink** містить логотип деякого ЗАТ, який позиціонований щодо його нормального положення в потоці елемен- тів сторінки, а також текст «ЗАТ Повітряна куля», що має оформ- лення, описане в класі «**revers**».

Горизонтальний прошарок кольору **mistyrose** розташований поверх вертикального й містить текст, оформлений згідно зі стилем **bluetext**.

Приклад 17_2

```
<html><head><title>Приклад 17_2</title>
<style type=«text/css»>
.revers {
font-weight:bold
; color:white;
text-align: center;}
.bluetext {
font-weight:bold;
color:darkblue;
text-align:
justify;
padding:0.5em;
```

```
margin:0}
</style>
</head>

<body>
<strong>Приклад 17_2</strong>
<!-вертикальний шар-->
<div style=«position: absolute; top:0;left:
250;width:200; height:300;background:gray»>
<img src=«метелик1.jpg» style=«position:relative;
top:20; left:25;» >
<p class=«revers»>МЕТЕЛИК</p>
</div>
<div style=«position: absolute;
top:210; left:25;width:400;
height:90;background :white»>
    <p class=«bluetext»>Метелик з абзацем лежить поверх правого стовпчика (шару кольору deeppink) з логотипом. Зображення логотипу позиціоноване відносно (relative) його нормального положення в потоці елементів сторінки</p>
</div>
</body>
</html>
```

На рис. 17 показано відображення тексту прикладу 17_2 у браузері.

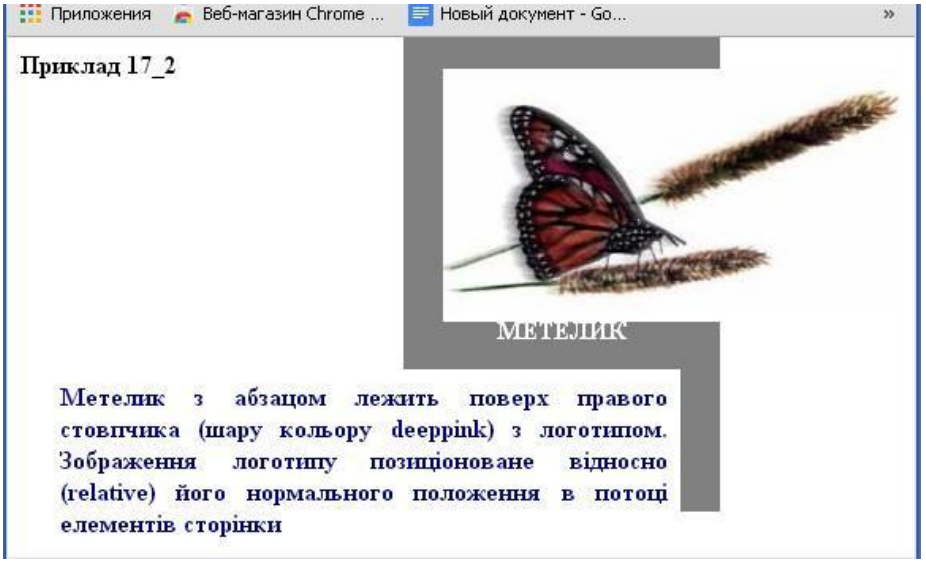


Рис. 17. Відображення тексту у браузері прикладу 17_2

Приклад 17_3 демонструє використання властивості **z-index**, яке дозволяє задати порядок (рівень) розташування елемента. Елемент ** ** використовується для визначення властивостей прошарку. Він є контейнерним елементом і дозволяє застосувати позиціонування й **z-index** для одного або декількох елементів, що вміщені в ньому.

У прикладі 17_3 на Web-сторінці виводиться текст і прошарок ** ** сірого кольору.

Приклад 17_3

```
<html><head><title>Приклад 17_3</title>
<style type=«text/css»>
<!--
.div_layer {
background-color:#c0c0c0
; position:absolute;left:80;
width:150; height:100;
visibility:visible;}
div_text {
color:blue
;
position:absolute;left:50
; width:350; height:100;
font-size:24pt;}
-->
</style>
</head>

<body>
<strong>Приклад17_3</strong>
<div class=«div_layer» style=«top:50;z-index:1;»>
</div>
<div class=«div_text» style=«top:50;»> z-індекс шару більше,
ніж тексту. Шар розташований «ближче» до користувача.</div>

<div class=«div_layer» style=«top:200;z-index:-1;»>
</div>
```

 z-індекс тексту більше, ніж шару. Текст розташований «ближче» до користувача.
</body></html>

На рис. 18 показано відображення тексту прикладу 17_3 у браузері.

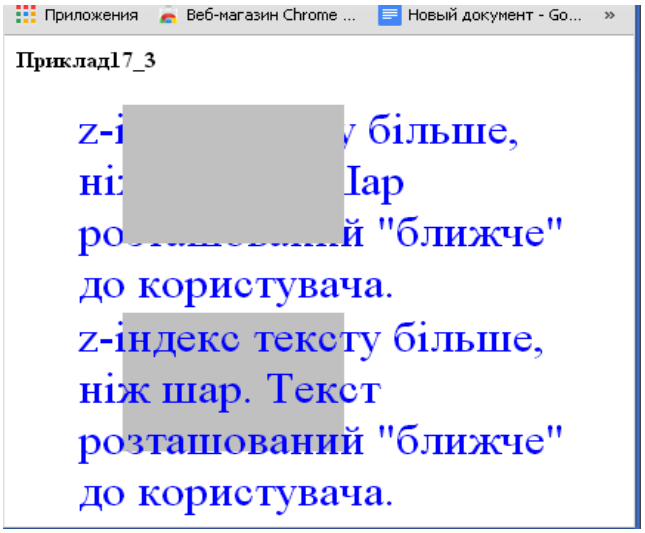


Рис. 18. Відображення тексту у браузері прикладу 17_3

Властивості прошарку сірого кольору й прошарку тексту описані за допомогою відповідних класів у розділі опису стилів (<**style**>). У верхній частині прикладу **z-index** сірого прошарку позитивний (рівний 1), тоді як текст має за замовчуванням **z-index** рівний 0. У результаті прошарок тексту перебуває на задньому плані, а сірий прошарок – на передньому плані.

У нижній частині прикладу **z-index** сірого прошарку негатив- ний (рівний –1), а текст лежить у нульовому прошарку. При цьо- му текст виявляється на передньому плані, а сірий прошарок – за текстом.

У прикладі 18 демонструється ефект, завдяки якому можна управляти поданням інформації у межах обмеженої області. Обмежена область менша, ніж зображення, а переповнення сховано. Таким чином, зображення вписується в обмежену область. Область із переповненням розміщена нижче звичайного зображення, яке оточено тонкою рамкою, а область із переповненням – рамкою з більшою товщиною.

Приклад 18

```
<html>
<html><head><title>Приклад 18</title>
<style type=«text/css»>
<!--
.overflow {
position: absolute; top:190;left:
50; width:80;
height:80;
border:2px solid
black;
overflow:hidden;
}
-->
</style>
</head>
<body>
<strong>Приклад18</strong><br>
<img src=«roza.jpg» width=150 height=150 alt= «flower» border=1>
<img class=«overflow» src=«roza.jpg» width=150 height=150 alt=
«flower»>
</body>
</html>
```

На рис. 19 показано відображення тексту прикладу 18 у браузері.

Рис. 19. Відображення тексту у браузері прикладу 18

У прикладі 19 використаний клас **clip1**, який описує текст, поміщений у квадратну область, описану дескриптором ** **. Клас **clip** забезпечує застосування до цієї області відсікання (зверху й ліворуч на 25 пікселів, а знизу й праворуч на 125 пікселів). На рис. 20 текст у квадратній області розташований праворуч, а відсічений – ліворуч.

Приклад 19

```
<html><head><title>Приклад 19</title>
<style type=«text/css»>
<!--
.clip {
position: absolute;
top:50; left: 230;
width:150;
height:150;
color:yellow
;
background-color:black;
clip:rect(25px 125px 125px 25px);}
.clip1{
position: absolute;
top:50; left: 30;
width:150;
height:150;
color:yellow
;
background-color:black;
}
-->
</style>
</head>
<body>
<strong>Приклад19</strong>
<div class=«clip1»>Текст абзацу буде обрізаний за допомогою ві-
```

дсікання, застосованого до даної квадратної області розміром 150 на 150 пікселів.

Текст абзацу буде обрізаний за допомогою відсікання, застосованого до даної квадратної області розміром 150 на 150 пікселів.
</body></html>

На рис. 20 показано відображення тексту прикладу 19 у браузері.

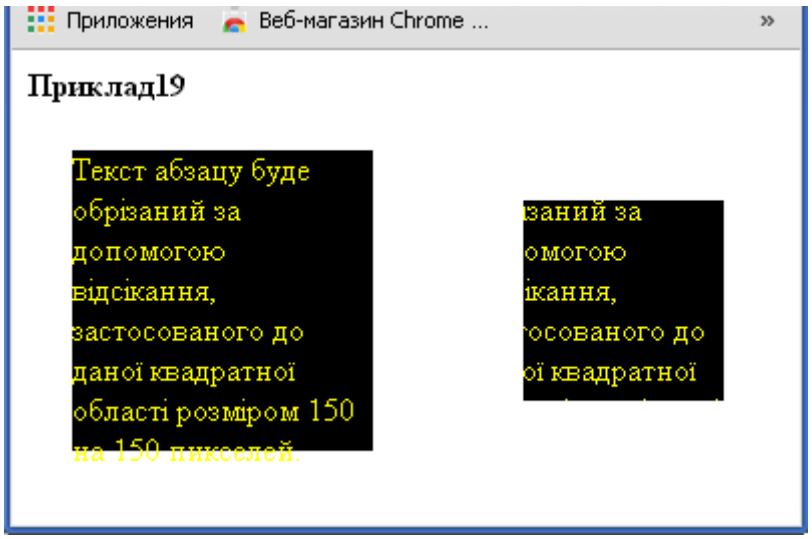


Рис. 20. Відображення тексту у браузері прикладу 19

У прикладі 19_1 показано застосування властивості **display** для створення горизонтального меню зі списку посилань. *Блоковий* елемент, яким є список посилань, перетворений у *рядковий* з оформленням, відповідним до елементів меню.

Приклад 19_1

```
<html><head><title>Пример 19_1</title>
<style>
#nav h5 {text-align:center;font-size:16pt}
#nav ul {padding: 3px 0;
border-bottom:1px solid #778;
font :bold 12px Verdana, sans-serif;
}

#nav ul li {
display: inline;
}
```

```
#nav ul li a {
padding: 3px
0.5em; margin-left:
3px;
border: 1px solid
#778; border-bottom:
none; background:
#dde;
text-decoration: none;
}
#nav ul li a:link
{ color: #0000ff;
}

#nav ul li a:visited
{ color: #667;
}
 #nav ul li a:link:hover, #nav ul li a:visited:hover {
color: #000;
background: #aae;
border-color:
#227;}

#nav ul li a#current
{ background: white;
border-bottom: 1px solid white;
}
</style>

</head>
<body>
<div id=«nav»>
<h5> navigation bar <h5>
<ul>
<li><a href=«bg1.html»>home</a></li>
<li><a href=«bg1.html»>about</a></li>
```

archives
contact

</body></html>

На рис. 21 показано відображення тексту прикладу 19_1 у браузері.

Рис. 21. Відображення тексту у браузері прикладу 19_1

Фільтри

Є два типи спеціальних ефектів для зміни виду тексту й графіки, які називаються **Visual Filter** (візуальні або статичні фільтри) і **Transitions Filter** (динамічні фільтри).

Ефекти фільтрів – від дзеркального відображення тексту й графіки до зникнення їх на шаховій дошці.

Статичні фільтри

Статичні фільтри можуть бути застосовані до тем елементів **HTML**, які розглядаються як керуючі. Це елементи, що створюють на Web-сторінці прямокутну область. До таких елементів відносять: **body, div, img, span, table, tr, td, input.**

Статичні фільтри міняють зовнішній вигляд об’єкта, і ця кар- тина залишається нерухливою. За властивістю стилю **filter** ста- виться двокрапка, потім – ім’я фільтра, за яким у круглих дужках перелічуються атрибути з їхніми значеннями, розділені комами. Відмінність у кодуванні різних фільтрів полягає у назві фільтра, а також у кількості й типі використовуваних атрибутів. У таблиці 2 наведений список статичних фільтрів, їх атрибути й функції.

Таблиця 2

Фільтр	**Необхідні атрибути**	**Дія**
alpha	opacity (0-100%), finish opacity, style (0,1 –зміна прозорості уздовж лінії, 2 – радіально від центру), startx, starty, finishx, finishy (координати крапок для style=1)	Задає рівень прозорості
Blur	add(=1), direction (0 – 360 – напрямокразмытия), strength (величина зсуву)	Створює ефект руху
chroma	color	Створює одноколірний транспарант
dropshadow	color (колір тіні), offx (зсув по осі x), offy (те ж по y), positive (0–1, тіні немає або є)	Створює тінь (на відстані)
fliph		Робить горизонтальне відображення об’єкта
flipv		Робить вертикальне відображення об’єкта
glow	color, strength	Створює ефект сяйва
Grayscale		Перетворить об’єкт у чорно-білий
Invert		Інвертує кольори, відтінки і яскравість
Light		Висвітлює об’єкт від джерела
Mask	color	Робить із об’єкта маску-транспарант
Shadow	color, direction	Створює тінь
Wave	add, freq (число максимумів), lightstrength, phase, strength	Створює хвилясте викривлення об’єкта
Xray		Показує контур об’єкта

У прикладі 20 наведений приклад дії на текст «сяючого» фільтра й фільтра горизонтального відображення.

Приклад 20

```
<html><head ><title>Приклад 20</title><style type=«text/css»>
<!--
.effect_glow {filter:glow(color=gray,strength=5);}
```

```
.effect_fliph {filter:flipv;}
-->
</style>
</head>
<body>
<strong>Приклад 20</strong><br><br>
<div class=«effect_glow»
style=«position:absolute;width:200;height:100;top:50;left:100;font
- family:verdana;font-style:bold;font-size:20;text-align:center;»>
Текст із ефектом «сяючого» фільтра!</div>
<div class=«effect_fliph»
style=«position:absolute;width:250;height:150;top:130;left:70;font
- family:verdana;font-style:bold;text-align:center;»>
Текст і графіка з ефектом вертикального відображення.
<img src=«roza.jpg» width=60 height=70 alt=«roza»>
</div>
</body>
</html>
```

На рис. 22 показано відображення тексту прикладу 20 у браузері.

Рис. 22. Відображення тексту у браузері прикладу 20

Динамічні фільтри

Динамічні фільтри дозволяють спостерігати плавний перехід від одного стану об'єкта до іншого зі швидкістю, що задається розроблювачем.

Для установки динамічного фільтра використовується дескриптор **meta.** За допомогою цього дескриптора можна управляти завантаженням сторінки в браузер і її відходом з екрана.

При вході на сторінку застосовується ефект «кола, що розширюється» (номер фільтра-3), а при виході зі сторінки – «шаховий порядок униз» (номер фільтра – 11). Html-код сторінки прикладу 21.1 містить гіперпосилання (на сторінку прикладу 21.2) і при активізації цього гіперпосилання відбувається вихід зі сторінки й спостерігається ефект − «шаховий порядок униз», пов'язаний з виходом зі сторінки. При завантаженні сторінки або поверненні на неї за допомогою кнопки браузера «назад» спостерігається вхідний ефект - «коло, що розширюється».

Для перегляду всіх ефектів можна встановити номер фільтра, рівний 23, що відповідає випадковому вибору типу фільтра.

Приклад 21.1

```
<html>
<head><title>Приклад 21.1</title>
<meta http-equiv=«Page-Enter»
content=«Revealtrans(Duration=6,Transition=3)»
>
<meta http-equiv=«Page-Exit»
content=«Revealtrans(Duration=6,Transition=11)»>
</head>
<body bgcolor=#d3d3d3 text=yellow>
<strong>Приклад 21.1</strong><br><br>
<a href=«pr21_2.html»> Перехід до прикладу 21.2</a>
</body>
</html>
```

Приклад 21.2

```
<html>
<head><title>Приклад 21.2</title>
```

</head>
<body bgcolor=darkcyan text=yellow>
Приклад 21.2

 Перехід до прикладу 21.1
</body>
</html>У таблиці 3 наведені номери фільтрів і ефекти переходу, відповідні до цих фільтрів.

Таблиця 3

Ефект переходу	Номер фільтра	Ефект переходу	Номер фільтра
Прямокутник, що стягається	0	Випадковий наплив	12
Прямокутник, що розширюється	1	Вертикальний розподіл усередину	13
Коло, що стягається	2	Вертикальний розподіл назовні	14
Коло, що розширюється	3	Горизонтальний розподіл усередину	15
Стирання нагору	4	Горизонтальний розподіл назовні	16
Стирання вниз	5	Стирання вліво вниз	17
Стирання праворуч	6	Стирання вліво нагору	18
Стирання ліворуч	7	Стирання вправо вниз	19
Вертикальні жалюзі	8	Стирання вправо нагору	20
Горизонтальні жалюзі	9	Випадкові горизонтальні лінії	21
Шаховий порядок поперечний	10	Випадкові вертикальні лінії	22
Шаховий порядок униз	11	Випадковий вибір з попередніх	23

На рис. 23 показаний ефект, що виникає при завантаженні

сторінки прикладу 21.1 у браузер.

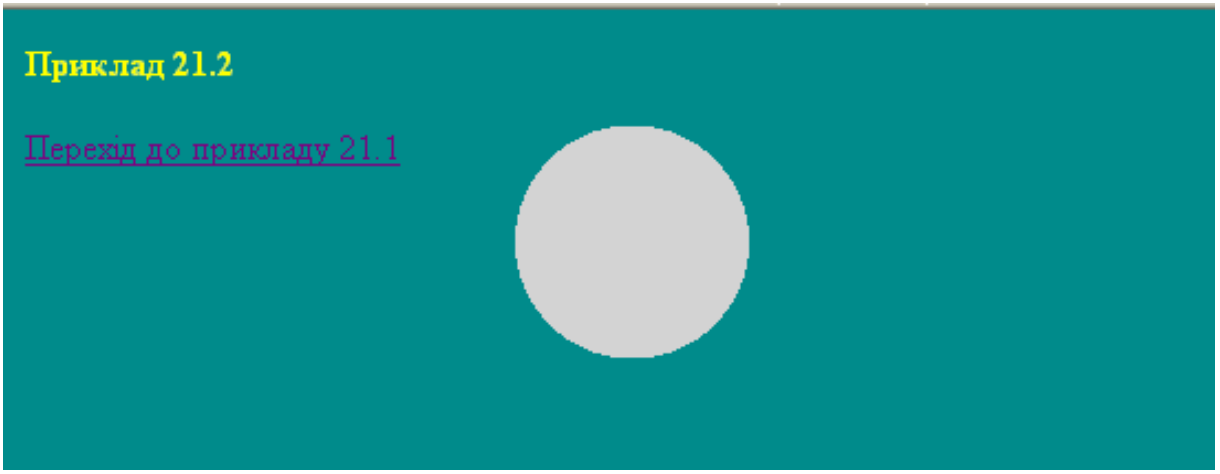


Рис. 23. Ефект, що виникає при завантаженні сторінки прикладу 21.1 у браузер

На рис. 24 показаний ефект, що виникає при виході зі сторінки прикладу 21.1 за допомогою гіперпосилання на сторінку прикладу 21.2.

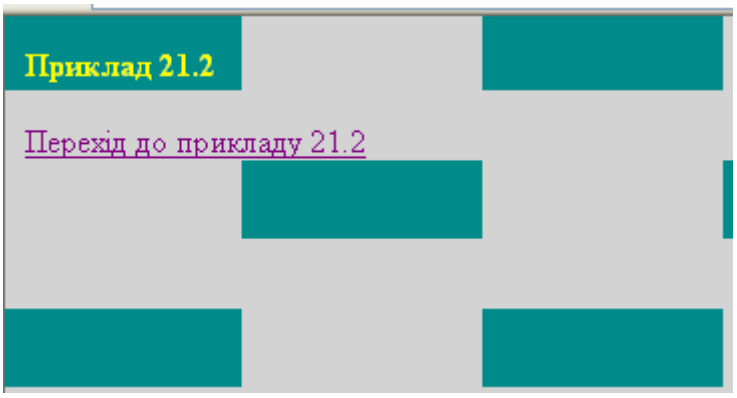


Рис. 24. Ефект, що виникає при виході зі сторінки прикладу 21.1 за допомогою гіперпосилання на сторінку прикладу 21.2

3. PHP

PHP – мова написання скриптів, які вбудовуються безпосередньо в гіпертекстові файли й виконуються на Web-сервері. Програма на PHP береться у теги, а інтерпретатор обробляє команди між цією парою тегів і формує остаточний файл, що передається на локальну машину. Як уже згадувалося, мова PHP досить проста і основні її конструкції (цикли, умови, функції) нагадують мову C.

У PHP 5 є сесії, подібні Active Server Page, що й дозволяє маніпулювати даними протягом усього сеансу доступу користувача, розширені вбудовані функції системи. Кожна конструкція PHP закінчується крапкою з комою, блоки програм виділяються фігурними дужками, для операцій присвоювання використовується знак рівності («=«), а для порівняння – подвійна рівність («==»). Мова підтримує унарні операції на зразок «++», «—=», «+=» та ін. Крім великої кількості вбудованих функцій, для роботи з рядками, числами, датами, файлами й масивами можна повідомляти додаткові функції за допомогою ключового слова function. До особливостей PHP відносять формат запису змін- них і масивів – кожний ідентифікатор змінної починається зі значка $.

Масиви в PHP служать для зберігання даних різних типів, у них допускається присвоювання елементам масиву чисел, рядків, дат та інших типів даних. Змінні можуть автоматично конвертуватися у масиви й назад, наприклад, виконання наступної конструкції не викликає помилок (при перетворенні попередні значення змінної виявляються загубленими):

```
// змінної $myvar привласнюється строковий вираз
$myvar=«filename.txt»;
// змінна $myvar перетвориться в масив, кожний елемент якого –
окремий рядок файла $myvar=File($myvar);
// змінної $myvar привласнюється числове значення $myvar=0;
```

Масиви автоматично створюються при завантаженні вмісту файла або при читанні рядка таблиці, але їх можна повідомляти й за допомогою функції array(), наприклад:

```
$a=array();
$a[1]=«січень»;
$a[2]=«лютий»;
:
:
$a[12]=«грудень»;
```

У квадратних дужках вказується індекс масиву. Якщо він опускається, то масив вважається динамічним. Так, можна написати:

```
$b=array();
$b[]=1;
$b[]=2;
$b[]=3;
$b[]=4;
```

При цьому в елементі $b[0] буде зберігатися значення 1, а в $b[2] – значення 3.

Для виводу значень у PHP використовуються оператори echo, print і функція printf(). Як правило, конструкції echo і print роблять те саме, а функція printf(), крім видачі строкових значень, виконує їхнє форматування, наприклад:

```
$name=«Петров»;
$name1=«Петро»
; print «<table>«;
printf(«<tr><td>%s</td><td>%s</td></tr>\n», $name, $name1);
print «</table>«
```

PHP тісно інтегрований з Web. Це, зокрема означає, що внесені у форми значення, передані гіпертекстовому файлу параметри, збережені Web-браузером, автоматично конвертуються в змінні. Так, описавши у формі поле введення **<input type=«text» name=«pdate» size=«12» value=«дата»>**, у скрипті можна маніпулювати змінною $pdate. Інтеграція з Web дозволяє створювати динамічні Web-сторінки, що залежать від параметра, наприклад гіперпосилання, по якому перейшов користувач, або введеного у

форму значення. Наприклад, задавши у файлі список тем опитувань, можна вивести їх у традиційній для сайтів формі.

```
<form method=«POST» action=«dovotes.php3»>
<?
$votefile=File(«votes.txt»)
; for ($i=0; $i<count
($votefile); $i++)
{
print «<p><input type=radio name=iselect value= $votefile[$i]>»;
}
?>
<center><input type=«submit» value=«Готове»
name=«B1»></center>
</form>
```

Елементи мови

Будь-який скрипт PHP складається з послідовності операторів. Оператор може бути присвоюванням, викликом функції, циклом, умовним виразом або порожнім виразом. Оператори, звичайно, закінчуються крапкою з комою. Також оператори можуть бути об’єднані в групи операторів у фігурних дужках. Група операторів також є оператором.

Константа

PHP визначає кілька констант і надає механізм для визначення ваших констант. Константи подібно до змінних, але вони мають злегка змінений синтаксис.

Визначені константи – це_FILE and_LINE_, які від- повідають імені файла й номеру рядка, який виконується у даний момент.

Приклад 1. Використання__FILE__і__LINE__

```
<?php
function report_error($file, $line, $message) {
```

```
echo «An error occured in $file on line $line: $message.»;
}

report_error(__FILE__,__LINE__, «Something went wrong!»);
?>
```

Ви можете визначити додаткові константи за допомогою функцій define() і undefine() .

Приклад 2. Опис констант

```
<?php
define(«CONSTANT», «Hello world.»);
echo CONSTANT; // outputs «Hello world.»
undefine («CONSTANT»);
?>
```

Вирази

Вирази – це наріжний камінь PHP. У PHP майже все є виразами. Найпростіший і найбільш точний спосіб визначити вираз – це «щось, що має значення».

Найпростіший приклад – це константи й змінні. Коли друкуєте «$a = 5», ви привласнюєте значення ‘5’ змінній $a.

Трохи більш складні приклади виразів – це функції. Подумаємо над наступною функцією:

```
function foo ()
{ return 5;
}
```

Припускаючи, що ви знайомі з концепціями функції, вважаєте, що $c = foo() це практично те саме, що написати $c = 5, і ви праві. Функції – це вирази з тим значенням, які вони повертають. Тому що foo() повертає 5, значення виразу ‘foo()’ – 5. Звичайно, функції підраховують значення, що повертається, а не повертають постійне значення.

Звичайно, значення у PHP не зобов’язані бути цілими і найчастіше вони не є такими. PHP підтримує 3 скалярних типи

значень: ціле, число із плаваючою крапкою і рядки (скалярні вирази ви не можете «розбити» на більш маленькі частини, як, наприклад, масиви). PHP підтримує 2 складових (нескалярних) типи: масиви й об'єкти. Кожне з таких значень може бути привласнене змінній або повернуте функцією.

Має сенс запис типу '$b = ($a = 5)' схожий на запис '$a = 5; $b = 5;' (крапка з комою означає кінець виразу). Тому що присвоювання розглядається з права на ліво, ви також можете написати '$b = $a = 5'.

У PHP є 2 типи інкремента – попередній і наступний. І попередній, і такий інкремент збільшує значення змінної і впливає на змінну ідентично. Різниця в значенні виразів інкремента. Попереднє збільшення, яке записується як '++$variable', прирівнюється до збільшеної змінної (PHP збільшує змінну до того, як прочитати її значення).

Дуже розповсюджений тип виразів – це вирази порівняння. Ці вирази мають значення 0 або 1 (означає неправду або істину відповідно). PHP підтримує > (більше, ніж), >= (більше або дорівнює), = (дорівнює), < (менше, ніж) і <= (менше або дорівнює). Ці вирази в основному використовуються всередині умов, наприк- лад оператора IF.

Останній приклад виразів, який розглянемо, це сполучені вирази оператор-присвоювання. Для того щоб збільшити значення $a на одиницю, ви можете написати '$a++' або '++$a'. Але, якщо потрібно збільшити значення більше, ніж на одиницю, наприклад на 3? Ви можете написати '$a++' кілька разів, але це не дуже зручно й ефективно. Набагато більш поширене написання '$a = $a + 3', '$a + 3' обчислюється, як значення $a плюс 3, а потім привласнюється змінній $a, у результаті чого значення $a збіль- шується на 3. Будь-який бінарний оператор може бути записаний таким методом, наприклад: '$a-=5' (відняти 5 зі значення $a), '$b*=7' (помножити значення $a на 7) тощо.

Є ще такий вираз, як умовний оператор із трьома операндами:

$first ? $second : $third

Якщо значення першого виразу істина (не дорівнює 0), то виконується другий вираз і це є результатом даного умовного виразу. Інакше виконується третій оператор.

Цей приклад повинен допомогти вам краще зрозуміти попереднє і наступне збільшення і взагалі вираз:

```
?
function double($i) /* функція подвоєння змінної */
 {
 return $i*2;
}
 $b = $a = 5; /* привласнюємо значення змінним $a і $b */
$c = $a++; /* наступне збільшення, привласнюємо $c початкове значення $a (5)*/
$e = $d = ++$b; /* попереднє збільшення, привласнюємо $d і $e збільшене значення$b (6) */
 /* отут і $d і$e рівні6 */
$f = double($d++); /* привласнити подвоєне значення $d до його збільшення, тобто 2*6 = 12, змінній $f*/
$g = double(++$e); /* привласнити подвоєне значення $e після його збільшення, тобто 2*7 = 14, змінній g */
$h = $g += 10; /* спочатку збільшити значення $g на 10, що дає в результаті 24, а потім привласнити це значення змінній $h, що також дає 24 */
```

Таблиця 4

Арифметичні оператори

приклад	назва	результат
$a + $b	Додавання	Сума $a і $b
$a - $b	Віднімання	Віднімає $b з $a
$a * $b	Множення	Добуток $a і $b
$a / $b	Ділення	Ділення $a на $b
$a % $b	Залишок ділення	Залишок від ділення $a на $b

Оператор ділення («/») повертає цілу величину, якщо обидва оператори – цілі (або рядок перетворено в ціле). Якщо кожний операнд є величиною із плаваючою комою, виконується розподіл із плаваючою комою.

Оператори рядків

(«.»).

Насправді є тільки один оператор – оператор конкатенації

$a = «Hello «;
$b = $a . «World!»; // тепер $b = «Hello World!»

Оператори присвоювання

Основним оператором присвоювання є «=». Це означає, що лівий операнд одержує значення виразу з права (збірне присвоювання).

Значенням виразу присвоювання є величина, що привласнюється. Тому, якщо «$a = 3», то це 3. Це дозволить робити деякі мудрі речі:

$a = ($b = 4) + 5; // тепер $a дорівнює 9, а $b стало дорівнювати 4.

До основних операторів присвоювання ще є додаткові

«комбінаційні оператори», для всіх арифметичних і строкових операторів, що дозволяє використовувати значення у виразах і потім установлювати своє значення у результаті цього виразу. Наприклад:
$a = 3;

$a += 5; // тепер $a дорівнює 8, нібито сказали: $a = $a + 5;
$b = «Hello «;
$b .= «There!»; // тепер $b дорівнює «Hello There!», неначе написали $b = $b . «There!»;

Бітові оператори

Бітові оператори дозволяють вам змінювати біти в цілих числах (таблиця 5).

Таблиця 5

Бітові оператори

приклад	назва	результат
$a & $b	И	Будуть установлені біти, які були встановлені і в $a і в $b. *Приклад $a=5; /* 0101 */ $b=12; /* 1100 */ $c=$a & $b; /* $c буде дорівнювати 4 (0100) */
$a | $b	Або	Будуть установлені біти, установлені в $a або $b. *Приклад: $a=5; /* 0101 */ $b=12; /* 1100 */ $c=$a|$b; /* $c буде (1101) */
~ $a	Не	Будуть установлені не_присутні в $a біти (реверс) *Приклад: $a=5; /* 0101 */ ~ $a /* $a буде рівно x (1010) */
$a ^ $b	Xor	Установлюються біти, які встановлені в $a або $b, але не в обох
$a << $b	Зсув вліво	Зрушує біти змінної $a на $b кроків уліво (кожний крок/зсув означає «помножити на 2»)
$a >> $b	Зсув вправо	Зрушує біти змінної $a на $b кроків вправо (кожний крок/зсув означає «розділити на 2»)

Таблиця 6

Логічні оператори

приклад	назва	результат
$a and $b	AND	TRUE, якщо і $a і $b TRUE.
$a or $b	OR	TRUE, якщо $a або $b TRUE.
$a xor $b	XOR	TRUE, якщо $a або $b TRUE, але не обоє.
! $a	NOT	TRUE, якщо $a не TRUE.
$a && $b	AND	TRUE, якщо і $a і $b TRUE.
$a || $b	OR	TRUE, якщо $a або $b TRUE.

Різниця у двох різних варіантах операторів «and» і «or» – у відмінності пріоритетів операцій.

Оператори порівняння

Оператори порівняння дозволяють вам порівнювати дві величини.

Таблиця 7

Оператори Порівняння

приклад	назва	результат
$a == $b	Дорівнює	TRUE, якщо $a еквівалентно $b.
$a != $b	Не дорівнює	TRUE, якщо $a не дорівнює $b.
$a<> $b	Не дорівнює	TRUE, якщо $a не дорівнює $b.
$a < $b	Менше	TRUE, якщо $a строго менше ніж $b.
$a > $b	Більше	TRUE, якщо $a строго більше $b.
$a <= $b	Менше або дорівнює	TRUE, якщо $a менше або дорівнює $b.
$a >= $b	Більше або дорівнює	TRUE, якщо $a більше або дорівнює $b.

IF

Структура IF – це одна з найважливіших можливостей багатьох мов, включаючи PHP. Вона дозволяє організувати виконання фрагментів коду за умовою. Можливості PHP щодо використання виразу IF схожі на C:

if (expr) statement

Тут обчислюється логічний результат «expr». Якщо expr рівно TRUE, то PHP виконає «statement», а якщо FALSE – проігнорує.

Наступний приклад виведе фразу ‘a is bigger than b’ якщо $a більше $b:

```
if ($a > $b)
print «a is bigger than b»;
```

Найчастіше потрібно виконати більше ніж один вираз за умовою. Звичайно, не треба оточувати кожний вираз конструктцією IF. Замість цього можна згрупувати кілька виразів у блок виразів. Наприклад, такий код не тільки виведе фразу, але і привласнить значення $a змінній $b:

```
if ($a > $b) {print «a is bigger than b»; $b = $a;}
```

Вираз IF може мати необмежений ступінь вкладеності в інші вирази IF, що дозволяє вам ефективно використовувати виконання за умовою різних частин програми.

ELSE

Найчастіше потрібно виконати один вираз, якщо дотримується яка-небудь умова, і інший вираз – якщо не дотримується. Для цього застосовується ELSE. ELSE розширює можливості IF по частині обробки варіантів виразів, коли воно дорівнює FALSE. Даний приклад виведе фразу ‘a is bigger than b’, якщо $a більше $b, і ‘a is NOT bigger than b’ – якщо ні:

```
if ($a > $b) {
print «a is bigger than b»;
} else {
 print «a is NOT bigger than b»;
}
```

Вираз ELSE виконується тільки, якщо вираз IF дорівнює FALSE, а якщо є конструкції ELSEIF – то якщо і вони також дорівнюють FALSE.

ELSEIF

ELSEIF є комбінацією IF і ELSE, ELSEIF. Як і ELSE дозволяє виконати вираз, якщо значення IF дорівнює FALSE, але на відміну від ELSE воно виконається тільки, якщо вираження ELSEIF дорівнює TRUE. Наприклад такий код виведе ‘a is bigger than b’, якщо $a>$b, ‘a is equal to b’, якщо, $a==$b, і ‘a is smaller than b’ – якщо $a<$b:

```
if ($a > $b) {
print «a is bigger than b»;
} elseif ($a == $b) {
print «a is equal to
b»;
} else {
print «a is smaller than b»;
}
```

Усередині одного виразу IF може бути декілька ELSEIF. Перший вираз ELSEIF, який буде дорівнюватиме TRUE, буде виконано. У PHP ви можете написати ‘else if’ (два слова), що буде означати те ж саме, що й ‘elseif’ (одне слово).

Вираз ELSEIF буде виконано, тільки якщо вираз IF і всі попередні ELSEIF дорівнюють FALSE, а даний ELSEIF дорівнює TRUE.

Інший синтаксис для оператора IF: IF(): ... ENDIF;

PHP пропонує інший шлях для групування операторів з оператором IF. Найбільш часто це використовується, коли ви впроваджуєте блоки HTML всередину оператора IF, але взагалі може використовуватися де завгодно. Замість використання фігурних дужок за «IF(вираження)» повинна стояти двокрапка, один або кілька виразів і завершальний ENDIF. Розгляньте такий приклад:

<?php if ($a==5): ?> A = 5 <?php endif; ?>

У цьому прикладі блок «A = 5» впроваджений всередині виразу IF, що використовується альтернативним способом. Блок HTML буде видний, тільки якщо $a дорівнює 5.

Цей альтернативний синтаксис застосуємо і до ELSE і ELSEIF (expr). Приклад подібної структури:

```
if ($a == 5):
print «a equals
5»; print «...»;
elseif ($a == 6):
print «a equals
6»; print «!!!»;
else:
print «a is neither 5 nor
6»; endif;
```

WHILE

Цикл WHILE – найпростіший тип циклу в PHP. Він діє як і його аналог у C. Основна форма оператора WHILE:

WHILE(expr) statement

Зміст оператора WHILE простий. Він пропонує PHP виконувати вкладений оператор доти, доки expr дорівнює TRUE. Значення виразу перевіряється щоразу на початку циклу, тому, якщо значення виразу зміниться усередині циклу, то він не перерветься до кінця поточної ітерації (виконання всього блоку вкладених операторів це одна ітерація). Іноді, якщо значення expr дорівнює FALSE із самого початку, цикл не виконується жодного разу.

У IF ви можете згрупувати трохи операторів усередині фігурних дужок або використовувати альтернативний синтаксис:

WHILE(expr): вираз... ENDWHILE;

Наступні приклади ідентичні, обоє виводять номера з 1 по 10:

```
/* example 1 */
$i = 1;
while ($i <= 10)
 { print $i++; }

/* example 2 */
$i = 1;
while ($i <= 10):
 print $i;
 $i++;
endwhile
;
```

DO..WHILE

Цикл DO..WHILE досить схожий на WHILE за винятком того, що значення логічного виразу перевіряється не до, а після закінчення ітерації. Основна відмінність у тому, що DO..WHILE гарантовано виконається хоча б один раз, що у випадку WHILE не обов’язково.

Для циклів DO..WHILE існує тільки один вид синтаксису:

```
$i = 0;
do {
print $i;
```

```
} while ($i>0);
```

Цей цикл виконається один раз, оскільки після закінчення ітерації буде перевірене значення логічного виразу, а воно дорівнює FALSE ($i не більше 0), і виконання циклу завершиться.

Досвідчені програмісти на C знайомі з іншим використанням DO..WHILE, що дозволяє припинити виконання блоку операторів у середині шляхом впровадження його в цикл DO..WHILE(0) і використання оператора BREAK. цей код демонструє таку можливість:

```
do {
if ($i < 5) {
print «i is not big
enough»; break;
}
$i *= $factor;
if ($i < $minimum_limit)
{ break;
}
print «i is ok»;
...process i...
} while(0);
```

FOR

Цикли FOR – найбільш потужні цикли в PHP. Вони працюють подібно їхнім аналогам в C.

Синтаксис циклу FOR:

FOR (expr1; expr2; expr3) statement

Перший вираз (expr1), безумовно, обраховується (виконується) на початку циклу.

На початку кожної ітерації обчислюється expr2. Якщо воно дорівнює TRUE, то цикл триває і виконуються вкладені оператори. Якщо воно дорівнює FALSE, то цикл закінчується.

Наприкінці кожної ітерації обчислюється (виконується) expr3.

Кожен із цих виразів може бути порожнім. Якщо expr2 порожній, то цикл триває нескінченно (PHP за замовчуванням вважає його рівним TRUE, як і C).

Розглянемо такі приклади. Усі вони виводять номери з 1 по 10:

```
/* приклад 1 */
for ($i = 1; $i <= 10; $i++)
 { print $i;
}

/* приклад 2 */
for ($i = 1;;$i++)
{ if ($i > 10) {
break;
 }
 print $i;
}

/* приклад 3 */
$i = 1;
for (;;) {
if ($i > 10)
{ break;
}
print $i;
$i++;
}

/* приклад 4 */
for ($i = 1; $i <= 10; print $i, $i++) ;
```

Звичайно, перший варіант видається кращим (або четвертий), але можливість використання порожніх виразів у циклі FOR найчастіше виявляється корисною.

PHP також підтримує альтернативний синтаксис FOR:

FOR (expr1; expr2; expr3): вирази; ...; endfor;

Інші мови використовують оператор foreach для того, щоб обробляти масиви або списки. PHP використовує для цього оператор while і функції **list()** і **each()** .

BREAK

BREAK перериває виконання поточного циклу.

```
$i = 0;
while ($i < 10) {
if ($arr[$i] == «stop»)
{ break;
}
$i++;
}
```

CONTINUE

CONTINUE переходить на початок найближчого циклу.

```
while (list($key,$value) = each($arr)) {
if ($key % 2) { // skip even
members continue;
}
do_something_odd ($value);
}
```

SWITCH

Оператор SWITCH схожий на групу операторів IF з однаковим виразом. У багатьох випадках вам потрібно порівняти змінну (або вираз) з багатьма різними значеннями і виконати різні фрагменти коду залежно від того, чому буде дорівнювати значення виразу. Це саме те, для чого призначено оператор SWITCH.

Наступні 2 приклади – це 2 різних шляхи для досягнення однієї мети, але один використовує серію операторів IF, а інший – оператор SWITCH.

```
/* приклад 1
*/ if ($i == 0) {
print «i equals 0»;
}
if ($i == 1) {
print «i equals
1»;
```

```
}
if ($i == 2) {
 print «i equals 2»;
}

/* приклад 2
*/ switch ($i) {
case 0:
print «i equals
0»; break;
case 1:
print «i equals
1»; break;
case 2:
print «i equals
2»; break;
}
```

Математичні функції довільної точності

Математичні функції довільної точності: **bcadd, bccomp, bcdiv, bcmod, bcmul, bcpow, bcscale, bcsqrt, bcsub**.

Дані функції задіяні тільки за умови, що PHP був скомпільований у режимі --enable-bcmath, тобто при включених у конфігурацію bcmath функціях.

bcadd

bcadd – додавання двох чисел довільної точності.

Опис

string bcadd (string лівий операнд, string правий операнд, int [масштаб]);

Додає *лівий операнд* до *правого операнда* і повертає суму типу string (строкова змінна). Факультативний параметр *масштаб* використовується, щоб установити кількість розрядів після десяткової оцінки в результаті.

bccomp

bccomp – порівняння двох чисел довільної точності.

Опис

int bccomp (string лівий операнд, string правий операнд, int [масштаб]).

Порівнює *лівий операнд* із *правим операндом* і повертає результат типу integer (ціле). Факультативний параметр *масштаб* використовується для установки кількості цифр після десяткової оцінки, що використовуються при порівнянні. При рівності двох операндів повертається значення 0. Якщо *лівий операнд* більший *правого операнда,* повертається +1, і якщо *лівий операнд* менший *правого операнда* – повертається –1.

bcdiv

bcdiv – операція розподілу для двох чисел довільної точності.

Опис

string bcdiv (string лівий операнд, string правий операнд, int [масштаб]).

Ділить *лівий операнд* на *правий операнд* і повертає результат. Факультативний параметр *масштаб* установлює кількість цифр після десяткової оцінки в результаті.

bcmod

bcmod – отримання модуля числа довільної точності.

Опис

string bcmod (string лівий операнд, string модуль).

Одержання модуля *лівого операнда*, використовуючи операнд *модуль*.

bcmul

bcmul – операція множення для двох чисел довільної точності.

Опис

string bcmul (string лівий операнд, string правий операнд, int [масштаб]).

Множить *лівий операнд* на *правий операнд* і повертає результат. Факультативний параметр *масштаб* установлює кількість цифр після десяткової оцінки в результаті.

bcpow

bcpow – зведення одного числа довільної точності в ступінь іншого.

Опис

string bcpow (string x, string y, int [масштаб]).

Зведення *x* у ступінь *y*. Параметр *масштаб* може використовуватися для установки кількості цифр після десяткової оцінки в результаті.

bcscale

bcscale – встановлює масштаб за замовчуванням для всіх математичних функцій.

Опис

string bcscale (int масштаб).

Ця функція встановлює заданий за замовчуванням параметр масштабу для всіх наступних математичних функцій, які явно не визначають параметр масштабу.

bcsqrt

bcsqrt – одержання квадратного кореня числа довільної точності.

Опис

string bcsqrt (string операнд, int масштаб).

Повертає квадратний корінь *операнда*. Факультативний параметр *масштаб* установлює кількість цифр після десяткової оцінки в результаті.

bcsub

bcsub – віднімає одне число довільної точності від іншого.

Опис

string bcsub (string лівий операнд, string правий операнд, int [масштаб]).

Віднімає *правий операнд* із *лівого операнда* й повертає результат типу string. Факультативний параметр *масштаб* установлює кількість цифр після десяткової оцінки в результаті.

Функції **Дати/Часу**

Функції Дати/Часу: checkdate, date, strftime, getdate, gmdate, mktime, gmmktime, time, microtime.

checkdate

checkdate – перевіряє правильність дати/часу.

Опис

int checkdate(int month, int day, int year).

Повертає true, якщо дана дата правильна, інакше false. Перевіряє правильність дати, заданої аргументами. Дата вважається правильною, якщо:

- рік між 1 900 і 32 767 включно;
- місяць між 1 і 12 включно;
- день знаходиться в діапазоні дозволених днів даного міся- ця. Високосний рік враховується.

date

date – формат локального часу/дати.

Опис

string date (string format, int timestamp).

Повертає рядок, відформатований згідно з даним рядком і використовуючи дану *часову мітку* або поточний локальний час, якщо не задана тимчасова мітка. У форматному рядку повинні використовуватися такі символи:

- a – «am» або «pm»;
- A – «AM» або «PM»;
- d – день місяця, цифровий, 2 цифри (на першому місці нуль);
- D – день тижня, текстовий, 3 букви; тобто «Fri»;
- F – місяць, текстовий, довгий; тобто «January»;
- h – година, цифровий, 12-часовий формат;
- H – година, цифровий, 24-часовий формат;
- i – хвилини, цифровий;
- j – день місяця, цифровий, без початкових нулів;
- l (рядкова ‘L’) – день тижня, текстовий, довгий; тобто «Friday»;
- m – місяць, цифровий;
- M – місяць, текстовий, 3 букви; т.е. «Jan»;
- s – секунди, цифровий;
- S – англійський порядковий суфікс, текстовий, 2 симво- ли; тобто «th», «nd»;
- U – секунди з початку століття;
- Y – рік, цифровий, 4 цифри;
- w – день тижня, цифровий, 0 означає неділю;
- y – рік, цифровий, 2 цифри;
- z – день року, цифровий; тобто «299».

Нерозпізнані символи у форматного рядка будуть друкува- тися як є.

Приклад функції date()

```
print(date(«l dS of F Y h:i:s A»));
print(«July 1, 2000 is on a « . date(«l», mktime(0,0,0,7,1,2000))).
```

Функції date () і mktime () можливо використовувати разом для того, щоб знайти дати в майбутньому або минулому.

Приклад функцій date() і mktime()

```
$tomorrow = mktime(0,0,0,date(«m») ,date(«d»)+1,date(«Y»));
$lastmonth = mktime(0,0,0,date(«m»)-1,date(«d»), date(«Y»));
$nextyear = mktime(0,0,0,date(«m»), date(«d», date(«Y»)+1).
```

Для того, щоб відформатувати дати на інших мовах, Ви потрібноі використовувати функції **setlocale()** і **strftime()**.

strftime

strftime – форматує локальний час згідно з налагодженням locale.

Опис

string strftime (string format, int timestamp).

Повертає рядок, відформатований згідно з даним форматним рядом і використовуючи дану часову мітку або поточний локальний час, якщо позначка не задана. Назви місяців і тижнів та інші, залежні від мови рядки, залежать від поточного locale, що встановлюється за допомогою **setlocale()**.

У форматному рядку слід використовувати такі специфікатори перетворень:

- %a – скорочена назва дня тижня згідно з поточним locale;
- %A – повна назва дня тижня згідно з поточним locale;
- %b – скорочена назва місяця згідно з поточним locale;
- %B – повна назва місяця згідно з поточним locale;
- %c – переважне уявлення дати і часу для поточного locale;
- %d – день місяця як десяткове число (у діапазоні від 0 до 31);
- %H – час як десяткове число в 24-годинному форматі (в діапазоні від 00 до 23);
- %I – час як десяткове число в 12-годинному форматі (в діапазоні від 01 до 12);
- %j – день року як десяткове число (у діапазоні від 001 до 366);

- %m – місяць, день року як десяткове число (у діапазоні від 1 до 12);
- %M – хвилини як десяткове число;
- %p – `am' або `pm' згідно з поточним часом або відповід- ні рядки для поточного locale;
- %S – секунди як десяткове число;
- %U – номер тижня поточного року як десяткове число, починаючи з першої неділі як перший день першого тижня;
- %W – номер тижня поточного року як десяткове число, починаючи з першого понеділка як перший день першого тижня;
- %w – день тижня як ціле число, неділя – 0-й день;
- %x – бажане уявлення дати для поточного locale, що не включає час;
- %X – бажане представлення часу для поточного locale, що не включає дату;
- %y – рік як десяткове число без сторіччя (у діапазоні від 00 до 99);
- %Y – рік як десяткове число, включаючи сторіччя;
- %Z – тимчасова зона чи назва або скорочення;
- %% – символ `%'.

Приклад функції strftime()

```
setlocale («LC_TIME», «C»);
print(strftime(«%A in Finnish is
«)); setlocale («LC_TIME», «fi»);
print(strftime(«%A, in French «));
setlocale («LC_TIME», «fr»);
print(strftime(«%A and in
German»)); setlocale («LC_TIME»,
«de»);
print(strftime(«%A.\n»));
```

Приклад буде працювати, якщо у вас установлені відповідні locale.

getdate

getdate – одержує інформацію про дату/час.

Опис

array getdate (int timestamp).

Повертає асоціативний масив, що містить інформацію про дату з такими елементами:

- «seconds» – секунди;
- «minutes» – хвилини;
- «hours» – годинник;
- «mday» – день місяця;
- «wday» – день тижня, цифровий;
- «mon» – місяць, цифровий;
- «year» – рік, цифровий;
- «yday» – день року, цифровий; тобто «299»;
- «weekday» – день тижня, текстовий, повний; тобто «Friday»;
- «month» – місяць, текстовий, повний; тобто «January».

gmdate

gmdate – форматує GMT/CUT час/дату.

Опис

string gmdate (string format, int timestamp).

Аналогічна функції **date()** за винятком того, що час вертається в Гринвічському форматі Greenwich Mean Time (GMT). Наприклад, при запуску у Фінляндії (GMT +0200) перший рядок нижче надрукує «Jan 01 1998 00:00:00», у той час, як другий рядок надрукує «Dec 31 1997 22:00:00».

Приклад функції gmdate()

```
echo date(«M d Y H:i:s»,mktime(0,0,0,1,1,1998));
echo gmdate(«M d Y H:i:s»,mktime(0,0,0,1,1,1998));
```

mktime

mktime – одержує тимчасову мітку UNIX для дати.

Опис

int mktime(int hour, int minute, int second, int month, int day, int year).

Попередження. Зверніть увагу на незвичайний порядок аргументів, який відрізняється від порядку аргументів у виклику функції mktime() з UNIX і який погано поводиться при неправильно заданих параметрах. Помилка досить часто зустрічається у скриптах.

Повертає тимчасову мітку Unix згідно з даними аргументами. Ця тимчасова мітка є цілим числом, рівним кількості секунд між епохою Unix (1 Січня 1970) і зазначеним часом.

Аргументи можуть бути опущені з права на ліво; кожний опущений у такий спосіб аргумент буде встановлений у поточну величину згідно з локальною датою й часом.

Mktime is useful for doing date arithmetic and validation, as it Mktime корисна при арифметичних діях з датою і її перевіркою, вона буде автоматично обчислювати коректну величину для тих, що вийшли за межі параметрів.

Наприклад, кожна з наступних строк повертає строку «Jan-01-1998».

Приклад функції mktime()

```
echo date(«M-d-Y», mktime(0,0,0,12,32,1997));
echo date(«M-d-Y», mktime(0,0,0,13,1,1997));
echo date(«M-d-Y», mktime(0,0,0,1,1,1998));
```

gmmktime

gmmktime – одержує тимчасову мітку UNIX для дати в GMT.

Опис

int gmmktime(int hour, int minute, int second, int month, int day, int year).

Ідентична **mktime()** за винятком переданих параметрів, що представляють дату в GMT.

time

time – повертає поточну тимчасову мітку UNIX.

Опис

int time (void).

Повертає поточний час, обмірюване в числі секунд із епохи Unix (1 Січня 1970 00:00:00 GMT).

microtime

microtime – повертає поточну тимчасову мітку UNIX у мікросекундах.

Опис

string microtime (void);

Повертає рядок «msec sec», де sec поточний час, обмірюваний у числі секунд із епохи Unix (0:00:00 1 Січня, 1970 GMT), а msec – це частина в мікросекундах. Ці функції доступні тільки в операційних системах, що підтримують системний виклик gettimeofday().

Функції для роботи з каталогами

Функції для роботи з каталогами: chdir, dir, closedir, opendir, readdir, rewinddir.

chdir

chdir – зміна каталога.

Опис

int chdir(string directory).

Змінює поточний PHP каталог на *directory*. Повертає FALSE, якщо не можна змінити, TRUE, якщо зміна відбулась.

dir

dir – клас каталога (псевдо-об’єктно орієнтований механізм).

Опис

new dir (string directory).

Псевдо-об’єктно орієнтований механізм для читання каталогу. Відкриває каталог із *directory*. Два реквізити доступні, якщо каталог був відкритий. Реквізит handle може бути використаний разом з іншими функціями роботи з каталогом типу **readdir()**, **rewinddir()** і **closedir()**. Реквізит path встановлює шлях каталогу, який був відкритий. Три методи доступні: читання, повернення до початку і закриття.

Приклад функції Dir()

```
$d = dir(«/etc»);
echo «Handle:
«.$d->handle.»<br>\n»; echo «Path:
«.$d->path.»<br>\n»;
while($entry=$d->read()) {
 echo $entry.»<br>\n»;
}
$d->close();
```

closedir

closedir – закрити дескриптор(handle) каталогу.

Опис

void closedir (int dir_handle).

Закриває потік каталогу, позначений як *dir_handle*. Потік попередньо повинен бути відкритий функцією **opendir()**.

opendir

opendir – відкрити дескриптор (handle) каталогу.

Опис

int opendir (string path).

Повертає дескриптор (handle) каталогу, який надалі використовується в **closedir()**, **readdir()**, і **rewinddir()** обігах.

readdir

readdir – читання даних з каталогу по дескриптору (handle).

Опис

string readdir (int dir_handle).

Повертає ім’я наступного файла з каталогу. Імена не вертаються в будь-якому специфічному порядку.

Приклад виводу усіх файлів у поточному каталозі.

```
<?php
$handle=opendir(‘.’);
echo «Directory handle: $handle\n»;
echo «Files:\n»;
hile ($file = readdir($handle))
{ echo «$file\n»;
}
closedir($handle);
?>
```

rewinddir

rewinddir – повернення до початку даних каталогу по дескриптору (handle).

Опис

void rewinddir (int dir_handle).

Скидає потік каталогу, позначений як *dir_handle* у початок даних.

Поштові функції

Функція mail() дозволяє відсилати пошту.

Опис

bool mail(string to, string subject, string message, string additional_headers).

Mail() автоматично посилає повідомлення, що міститься в message, адресатові, зазначеному в to. Декілька одержувачів можуть бути зазначені в полі to у вигляді рядка з адресами, розділеними пробілами.

Приклад 1 посилки пошти:

mail(«rasmus@lerdorf.on.ca», «Моя тема»,
«Рядок 1\nстрока 2\nстрока 3»);

Якщо заданий четвертий строковий аргумент, він автоматично вставляється у кінець заголовка. Звичайно, це використовується при додаванні додаткових полів у заголовок. Кілька додаткових полів розділяються символом нового рядка.

Приклад 2 посилки пошти з додатковими полями заголовка: mail(«ssb@guardian.no», «the subject», $message,

«From: webmaster@$SERVER_NAME\nreply-To: webmaster@$SERVER_NAME\nx-Mailer: PHP/» . phpversion());

4. XML

XML (*Extensible Markup Language*) – це мова розмітки, що описує цілий клас об'єктів даних, які називаються XML-документами. Ця мова використовується як засіб для опису граматики інших мов і контролю за правильністю складання документів. Тобто сам по собі XML не містить ніяких тегів, призначених для розмітки, він просто визначає порядок їх створення. Таким чином, якщо, наприклад, ми вважаємо, що для позначення елемента *porche* в документі необхідно використовувати тег <*car*>, то XML дозволяє вільно використовувати визначений користувачем тег і включати його у XML – документ.

Виділяють сім основних характеристик мови XML:

1. XML пропонує метод структуризації файла у вигляді тек- стового файла.
2. XML подібний до HTML.
3. XML зрозумілий як комп'ютеру, так і людині.
4. XML утворює ціле сімейство технологій.
5. XML достатньо гнучкий.
6. XML достатньо новий, але у нього глибоке коріння.
7. XML вільний від ліцензійних відрахувань, платформен- но-незалежний, має широку підтримку.

XML пропонує метод структуризації файла у вигляді текстового файла.

XML забезпечує таку можливість, оскільки будь-яке його застосування може працювати з текстовими документами і будь-яка людина може прочитати і зрозуміти текст.

XML дозволяє зберігати у текстовому форматі структуровані дані. XML – це набір правил для створення текстових форматів, простих для обробки комп'ютерами різних типів. Отримані текстові файли структуровані таким чином, що вони:

- точно виражені;
- розширені;
- платформенно-незалежні.

Для розроблення XML-файлів можна використовувати будь-

який текстовий редактор. XML-документи, як правило, мають ро-

зширення *.xml, але спеціалізовані діалекти, створені в рамках XML, можуть мати розширення:

– *.xls– файли розширеної таблиці стилів (Extensible Stylesheet Language);

– *.xsd – визначення розширеної схеми (Extensible Schema Definition);

– *.xdr – скорочена схема даних XML (XML Data Reduced Schema);

– *.mml – математична мова розмітки (MATHML Mathematical Markup Language);

– *.cdf – формат визначення каналів (Channel Definition Format).

Створення XML-даних

У мовах XML і HTML є декілька потрібних характеристик. Якщо розглянемо приклад розмітки наступного тексту, відповідь на питання, це XML або HTML, буде правильною у будь-якому випадку, оскільки це приклад оформлення документів і в XML, і в HTML.

 Так, зазвичай, оформлюють

 виділений текст у HTML

Проте мова XML була розроблена для того, щоб подолати обмеження, що накладаються мовою HTML. Так, розробник XML-документа може сам визначити ряд своїх власних дескрипторів. Наприклад, якщо дескриптор параграфа в HTML – – єдиний, який може задавати й описувати параграф, то розробник документа XML може самостійно ввести дескриптор параграфа одним з нижче перелічених дескрипторів:

<indent>

<paragraph>

<para>.

Найпростіший елемент включає дескриптор, що відкривається, вміст, дескриптор, що закривається. Наприклад, <title> Вивчаємо XML </title>.

ПРАВИЛО. Увесь рядок <title> Вивчаємо XML</title> називається ***елементом***, дані між дескрипторами називаються ***вмістом елемента***.

ПРАВИЛО. Всі елементи мають бути обов'язково завершені. Усі не порожні елементи обов'язково повинні містити дескриптор, що відкривається, і дескриптор, що закривається. Порожні елементи мають бути закриті за таким правилом.

ПРАВИЛО. Порожній елемент завжди записується за стандартним правилом синтаксису порожнього елемента:

<ім'я _елемента/>.

Розміщення атрибутів у екземплярі XML

Слід зазначити, що як і в HTML у мові XML є атрибути, які змінюють або класифікують елементи і вказуються у дескрипторі, що відкривається.

Синтаксис визначення атрибута для елемента такий:

<ім'я_елемента ім'я_атрибута= «значення»> Зміст елемента відповідного елемента </ім'я_атрибута>.

Атрибути розміщуються завжди у дескрипторі, що відкривається. Дескриптор, що відкривається, у елементі може містити декілька атрибутів, дотримуючись таких правил:

<ім'я_елемента
ім'я_атрибута= «значення»
ім'я_атрибута= «значення»
ім'я_атрибута= «значення»> Зміст елемента відповідного елемента </ім'я_атрибута>.

Наприклад,

```
<account type= «checking» currency= «Gryvnja»>
    <name>Івченко</name>
    <balance>18623,12</balance>
</account>.
```

Усі значення атрибутів мають бути обов'язково в лапках. У разі відсутності хоча б однієї з лапок парсер видає таке зауваження (рисунок 25):

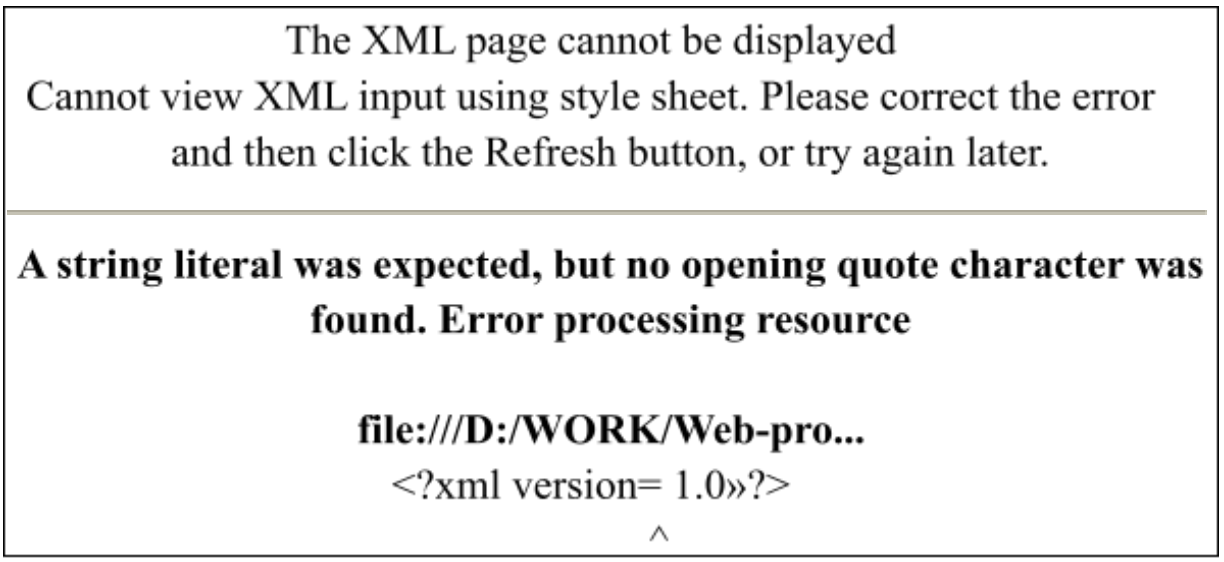


Рис. 25. Результат відображення неправильно оформленого атрибута

Визначення того, чи є дана властивість елементом або атрибутом, досить непросте питання.

Порожні елементи

Елементи XML можуть містити текст, інші елементи, будь-яку комбінацію тексту та інших елементів або ж бути просто порожніми елементами.

Порожній елемент завжди записується за стандартним правилом синтаксису елемента:
<ім'я _елемента/>.

Наприклад, **<date month= «September» day= «19» year= «2009» />.**

Даний елемент є порожнім, навіть не дивлячись на те, що містить атрибути. Враховуючи той факт, що повна інструкція
<date month= «September» day= «19» year= «2009»>
</date>, функціонує також і коротка, для написання порожнього елемента прийнято використовувати коротку форму.

Розглянемо фрагмент HTML-тексту:
<html>
<h1> letter </h1>
From: O. Protsenko
to: All students

Subject: Questions to exam
Date: 19.09 2009
<message></message>
</html>

аналогічний фрагмент XML-тексту виглядає так:

<letter>
<from> O. Protsenko</from>
<to> All students </to>
<subject> Questions to exam</subject>
<date month=”September” day=“19” year=”2009” />
<message> Questions </message>
</letter>.

Якщо поставити собі питання, який із фрагментів містить більше даних для обробки програмним додатком, то відповідь зрозуміла – XML.

Інша відмінність полягає у тому, що HTML змішує зміст і форматування в одному потоці розмітки. Так, наприклад, елементи <h1> і <b> показують, яким чином мають бути виділені елементи і де розміщені об’єкти, виділені такими елементами.

Мова XML припускає, що зміст і зовнішній вигляд повинні зберігатися окремо від даних розмітки. XML повністю покладається на каскадні таблиці стилів (CSS або XSL) при відображенні або перетворенні документів з однієї структури в іншу.

Сім’я XML-технологій

Оскільки XML-документ містить елементи, які описують самі себе, він зрозумілий людині на інтуїтивному рівні. Семантика даних забезпечує «інтелектуальність», яка подана в XML-елементах і значеннях атрибутів.

Незважаючи ні на що, XML – це програмний код, який читається і використовується обробниками XML.

XML утворює цілу сім’ю технологій XML, до якої належить ряд важливих технологій:

XML Version 1.0	Технічні рекомендації про використання XML
DTD	Визначення типу документа
XDR	Формат XML Reduced (схема Microsoft)
XSD	Визначення схеми XML (схема консорціуму W3C)
Простори імен	Метод визначення імен елементів та атрибутів
XPath	Мова шляхів XML
XLink	Мова посилань XML
XPointer	Мова покажчиків XML
DOM	Програмний інтерфейс API для об'єктної моделі документів
SAX	Simple API for XML (Простий програмний інтерфейс API для XML)
XSL	Розширена мова таблиць стилів
XSL-FO	Об'єкти форматування XSL
XSLT	Мова перетворень XSL
X Include	Синтаксис XML Include
XBase	Синтаксис XML Base URI

Деякі з перелічених компонентів до сьогодні знаходяться у процесі розроблення, хоча використовуються досить широко і можуть зазнавати значних змін. Тому особливу увагу необхідно приділяти тому, як та чи інша технологія описана в W3C.

Створення і перегляд XML-документа

По-перше, для створення XML-документів необхідний будь-який текстовий редактор (редактор, який здатний зберігати дані формату ASCII).

Проте існують спеціальні програмні засоби, які дозволяють вводити код і перевіряти синтаксис XML-документа, наприклад Architag X-Rey Edition (www.architag.com/xray) – версія доступна для загального користування. Існують і комерційні версії ПО, знайти які не важко в Інтернеті.

По-друге, потрібна спеціальна програма-обробник XML-файлів – парсер. Парсер – це програмне забезпечення, яке переві-

ряє дотримання синтаксичних правил XML і повідомляє про всі виявлені помилки. Якщо розмітка правильна, парсер перетворить його в такий вигляд, щоб його можна було читати. Цей процес називається перетворенням *розширеної мови таблиць XML XSLT.*

Парсер можна використовувати у браузер Internet Explorer. Наведений приклад помістіть у файл *example_1.xml:*

<letter>
<from> O. Protsenko</from>
<to> All students </to>
<subject> Questions to exam</subject>
<date month= «september» day= «19» year= «2009» />
<message> Questions </message>
</letter>.

У разі правильно оформленого документа XML браузер перетворить документ, застосовуючи до нього певні стилі.

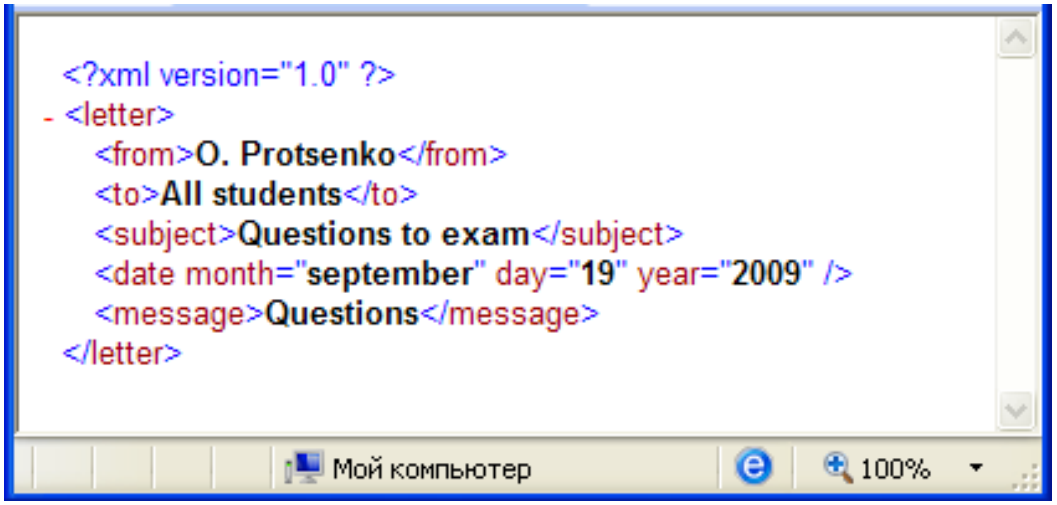


Рис. 26. Результат відображення XML-коду

Синтаксис мови XML

Документ XML вважається добре оформленим, якщо він відповідає всім правилам XML. Якщо ж хоч одне правило порушується, документ не вважається добре оформленим і не буде оброблений парсером.

Добре оформлений документ повинен містити один і лише один кореневий елемент, що містить решту всіх елементів. Елементи кореневого елемента можуть містити дочірні елементи, які мають бути правильно вкладені. Усі елементи, які лежать у кореневому елементі, вважаються дочірніми щодо кореневого.

Крім того, дескриптори в XML чутливі до регістру. Дескрип- тори, що відкриваються, і дескриптори, що закриваються, одного і

того ж елемента мають бути вказані з використанням одного і того ж регістру.

Якщо створюється XML-документ на основі існуючого документа, то необхідно добре вивчити оригінал, щоб зрозуміти структуру документа. Цей процес називається *аналізом документа*. Наприклад, є інструкція щодо миття машини. Машину опо- лоснути водою, потім нанести піну і залишити на 5 хвилин. Піну змити. Нанести на поверхню кузова віск, витерти корпус машини. Розглянемо структуру документа (рис. 27). На схемі видно, що кореневим елементом буде елемент «інструкція», всі інші бу- дуть дочірніми щодо нього.

Створення XML-документа

1. Визначте кореневий елемент, наприклад <directions> (інструкція). Документ повинен мати дескриптор, що відкривається, і дескриптор, що закривається <dire ctions> </directions>.
2. Кореневий елемент <directions> буде містити решту елементів <title>, <ingredients>, <instrument>, <actions>.
3. Елементи <ingredients> і <instrument> будуть містити до- чірні елементи <items> <quantity>.
4. Розставляємо по місцях усі відкриваючі і закриваючі дескриптори, заповнюючи їх необхідними даними.
5. Зберігаємо отриманий документ у файлі example_2.xml.
6. Правильність оформлення документа можна продивитися у вікні Internet Explorer.

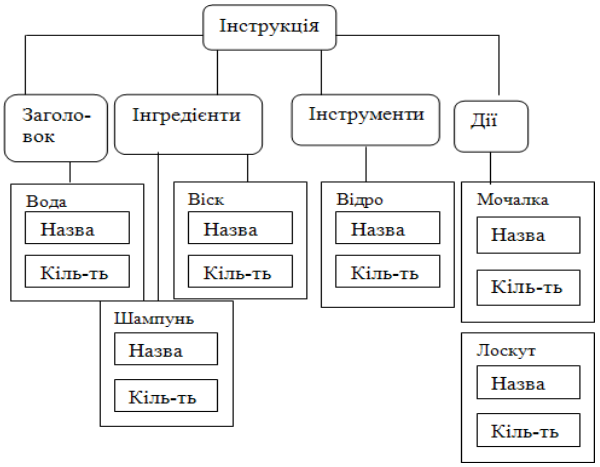


Рис. 27. Схема документа

<directions>
<title>Інструкція як помити машину</title>
<ingredients>
<items>Вода</items>
<quantity>50 л</quantity>
<items>Шампунь</items>
<quantity>0,5 л</quantity>
<items>Віск</items>
<quantity>0,3 л</quantity>
</ingredients>
<instrument>
<items>Відро</items>
<quantity>1 шт. </quantity>
<items>Мочалка </items>
<quantity>1 шт. </quantity>
<items>Лоскут для полірування</items>
<quantity>2 шт. </quantity>
</instrument>
<actions> Машину ополоснути водою, потім нанести піну і залишити на 5 хвилин. Піну змити. Нанести на поверхню кузова віск, витерти корпус машини
</actions></directions>.

Результат обробки парсером коду, що розглянули, поданий на рисунку 28.

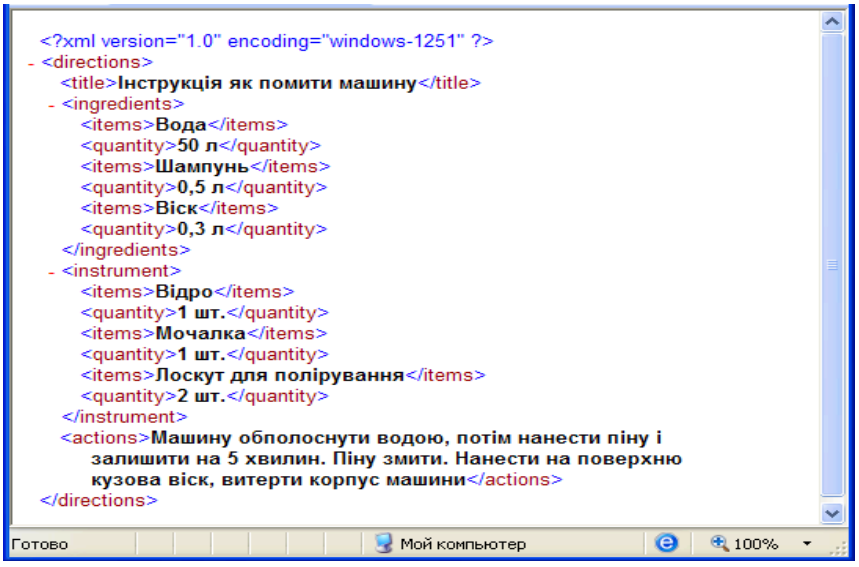


Рис. 28. Результат обробки парсером програмного коду

Правильне вкладення елементів

Усі документи XML мають бути перевірені на правильність вкладення елементів.

Мова HTML позбавлена таких недоліків. Вкладення елементів у наведених нижче інструкціях дають однаковий результат.

** Цей текст буде виділений курсивом **

** Цей текст буде виділений курсивом .**

XML досить чутливий до неправильного вкладення елементів, тому коректним вкладенням буде те, в якому елементи не перетинаються.

** Цей текст буде виділений курсивом .**

ПРАВИЛО. Завжди ставте рядки між дескриптором, що відкривається, та дескриптором елемента, що закривається, тоді ніколи не отримаєте накладення рядків і перетину документів.

Наприклад,

Цей текст буде виділений курсивом
.

Визначення імен у XML

ПРАВИЛО. При визначенні імен елементів у документі XML необхідно дотримуватися таких правил:

Ім’я елемента повинне починатися з букви, знаку підкреслення (_) або двокрапки (:).

Після першого символу в імені елемента можуть бути букви, цифри, знаки перенесення (–), знаки підкреслення (_), крапка або двокрапка (:).

Імена елементів не можуть починатися з букв XML або варіацій на цю тему, оскільки всі подібні імена захищені правами на інтелектуальну власність консорціуму W3C.

Декларації XML

Відомо, що добре оформлені документи без проблем від ображаються будь-яким парсером. Хоча парсер розуміє, що від

ображуваний документ є XML-документом. Порібно вказувати, що це документ XML.

Деякі парсери вимагають наявність у документі відповідного рядка декларації XML, який має такий вигляд:

<? xml version= «1.0» ?>.

Оголошення XML-документа може містити також ***оголошення кодування*** (encoding declaration), яке вказує на форму символів, і оголошення ***самостійності документа*** (standalone declaration).

Повний рядок декларації виглядає так

<?xml version= «1.0» encoding= «.» standalone= «.»?>.

Значення атрибута encoding містить кодування символів документа, а значення атрибута standalone вказує, чи є даний документ самостійним, і може набувати значення yes або no.

ЗАУВАЖЕННЯ. Якщо не вказувати тип кодування XML-документа, в якому є символи кирилиці, браузер, наприклад Internet Explorer, сприйматиме добре оформлений документ як документ, що містить помилки, і відображувати його не буде.

Додавання коментарів

Коментарі в XML додаються так, як і в HTML.

<!-- це коментар -->.

Правильні екземпляри XML

У XML разом з концепцією «добре оформленого документа» розглядається концепція «дійсного документа XML».

Правильний документ гарантує цілісність структури даних. Завдяки цьому значно спрощується доставка й обмін даними, які коректуються параметрами XSLT.

Для визначення правильності документа необхідно:

– визначити використання тільки заданого набору дескрипторів.

– перевірити, щоб порядок проходження елементів і їх атри- бутів повністю відповідав змісту документа або певним

правилам.

Іншими словами, у XML-документі має бути правильно реалізована схема документа, що визначає його структуру.

Як правильно визначити структуру. Повернемося, наприклад, до створення XML документа для інструкції щодо миття машини. Документ XML добре оформлений, але з погляду правильності він надлишковий (на кожну складову відводиться два елементи). Логічно було б використовувати один елемент з атрибутом <items quantity= “.”>. А XML-код виглядатиме так, як показано на рис. 29.

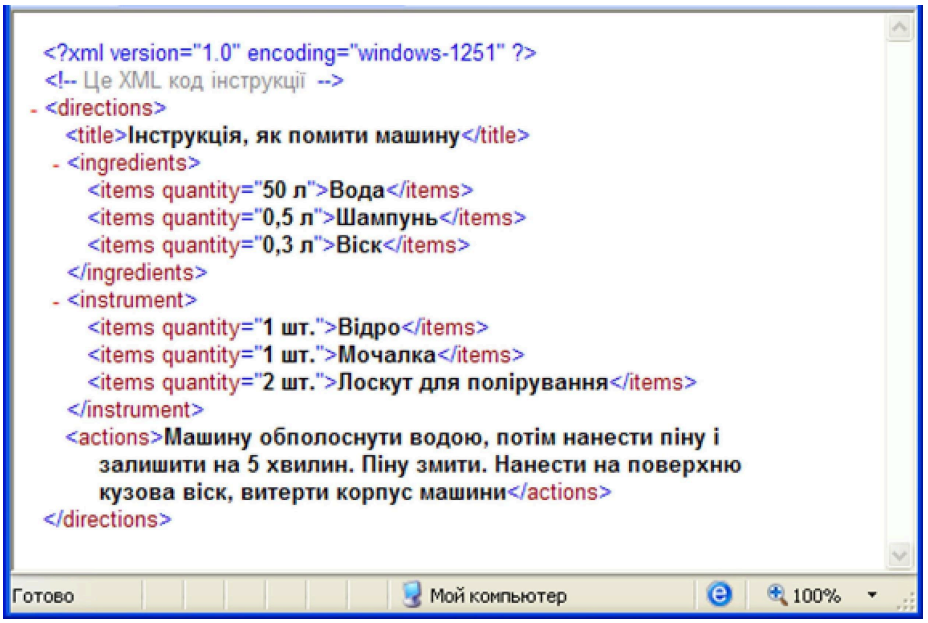


Рис. 29. Приклад документа

1. Контроль типів даних

Контроль типів даних досягається завдяки використанню відповідної схеми. Дані XML-документа, які використовуються при обміні, повинні використовувати один формат для запобігання плутанини. Наприклад, дата може бути оформлена в одному з таких форматів:

mmddyy yyddmm
ddmmyy yymmdd.

Існує декілька підходів до контролю типів даних. Пізніше розглянемо технології XDR і DTD.

До наступних типів даних може буде потрібна перевірка правильності. Необхідність перевірки залежить від того, обмін якою інформацією здійснюється.

Тип даних	Опис
Boolean	Логічний тип, значення ІСТИНА / БРЕХНЯ
Char	Один символ
String	Рядок символів
Float	Дійсні числа
Int	Цілі числа
Date	Дата у форматі YYYY-DD-MM
Time	Час у форматі HH-MM-SS
Id	Текст, унікальним чином ідентифікуючий елемент
Idref	Посилання на ідентифікатор
Enumeration	Послідовність значень, з якої можна вибрати будь-яке значення

2. Здійснити контроль цілісності даних для забезпечення оптимального обміну інформацією через Web за допомогою транз акцій.

Якщо йдеться про перевірку правильності оформлення документа, то йдеться не про представлення даних, а про структуру даних. Розглянемо, у чому ж полягає відмінність структури документа від структури даних?

Структура документа дозволяє читачеві швидко зрозуміти, в якому саме вигляді подав інформацію автор документа.

Структури даних вказують шлях комп'ютерного застосування даних, які містяться в різних контейнерах цілого документа. У структурі даних не міститься визначення важливості одного компонента документа щодо іншого. Усі компоненти рівні.

Парсер – це програма (у специфікації консорціуму W3C називається обробником XML), яка інтерпретує символ за символом. Існує два типи парсерів:

– перевіряють форматування документів, тобто їх

відповід- ність синтаксичним правилам;

– спочатку перевіряють форматування документа, а потім їх відповідність усім обмеженням, вказаним у пов’язаних з ним документах.

Визначення типу документа

Зв’язок схеми з документом дозволяє розширити можливість поширення документа незалежно від додатка. Схема додає ряд обмежень, які визначають список необхідних елементів і атрибутів, порядок їх проходження, а також за необхідності їх допустимі значення.

Схема DTD надає шаблон розмітки документа, в якому вказується наявність, порядок проходження і розміщення елементів і їх атрибутів у документі.

Також XML-документ можна подати у вигляді дерева, схему DTD також можна подати у вигляді дерева. Проте відмінність DTD полягає у тому, що дерево DTD не повторює елементи або структуру.

Наприклад, деревоподібну структуру XML-документа, що містить інструкцію до миття машини, можна подати у вигляді схеми на рисунку 30, а деревоподібна схема DTD виглядає так, як показано на рисунку 31.

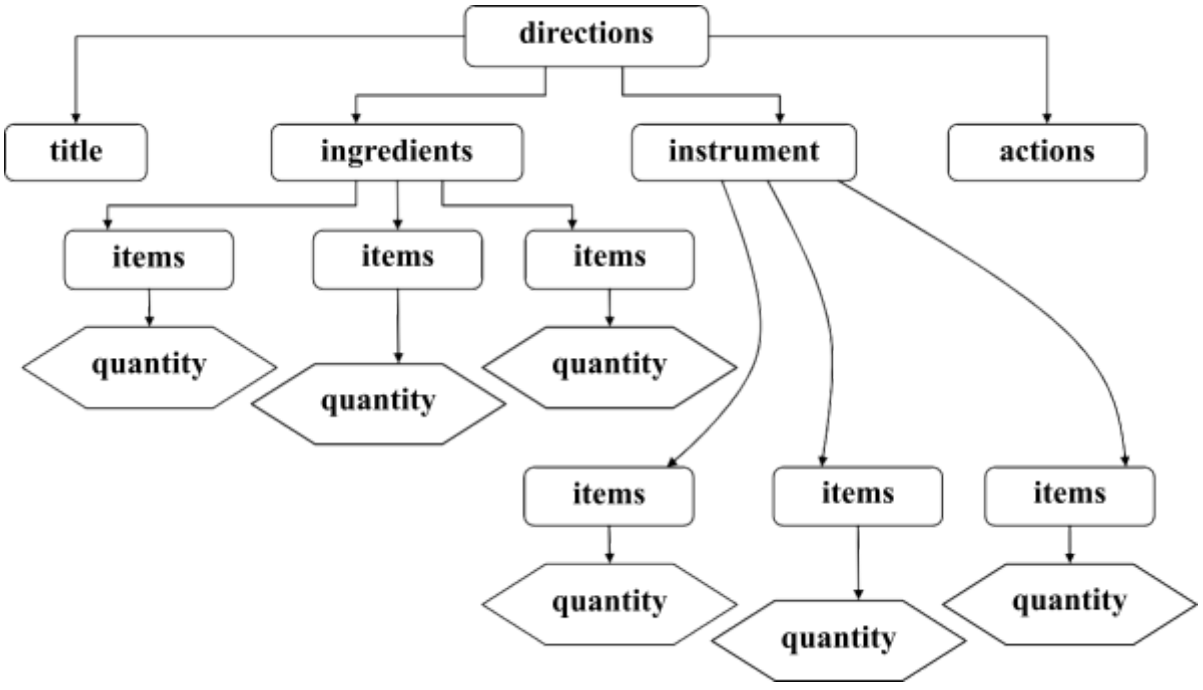


Рис. 30. Деревоподібна структура XML-документа

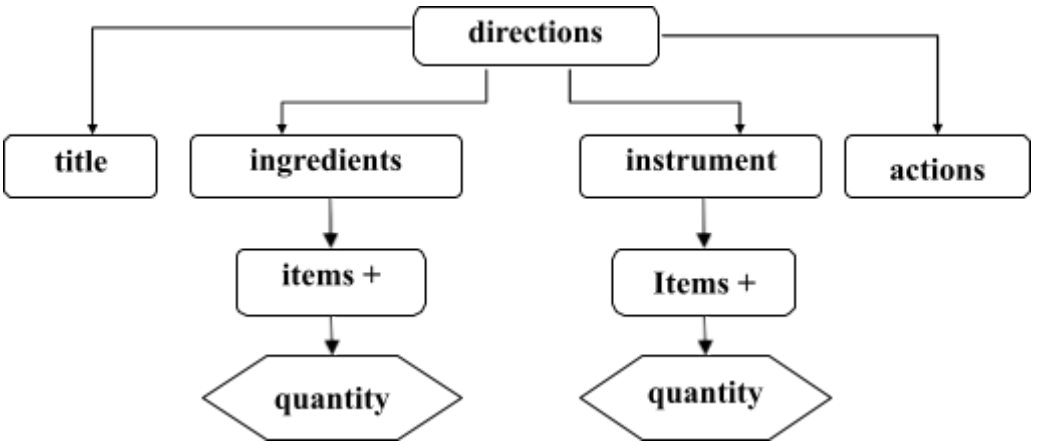


Рис. 31. Дерево DTD для XML-файлf

Схеми DTD мають справу з елементами документа XML, елементами-контейнерами, пустими елементами. Елементи-контейнери можуть містити дані, наприклад, текст, дочірні елементи або те й інше.

Оголошення змісту елемента або атрибута в схемі DTD називається ***моделлю змісту для*** цього ***елемента*** або ***атрибута***.

У будь-якому XML-документі елементи – фундаментальні структури, які об'єднані для представлення екземпляра XML. Тому кожен елемент має бути оголошений у схемі DTD разом з оголошенням його типу.

Оголошення типів елемента мають таку структуру:

<! ELEMENT ім'я_елемента (модель)>.

Усі приклади, які будуть наведені нижче, пов'язані між собою. Кожен приклад зберігатимемо в окремому файлі як нову версію попереднього.

Простий елемент із текстовим вмістом

Розглянемо приклад, де необхідно оформити як екземпляр XML записку «Завтра о 12.45 лекція з Web-дизайну».

Екземпляр добре оформленого коду example_3.xml виглядатиме так:

<?xml version="1.0" encoding="windows-1251" ?>

<note> Завтра о 12.45 лекція з Web-дизайну </note>

У цьому екземплярі елемент <note> містить тільки текст.

Цей документ можна перевірити, створивши DTD схему з оголошенням типу цього елемента, яка вказує на те, що цей елемент може мати тільки зарезервоване ключове слово #PCDATA або текст.

#PCDATA (завжди вказується знаками тільки верхнього регістру) – це звичайні текстові дані, але які зчитуються парсером і обробляються належним чином. Перевірка документа на відповідність DTD-схемі у разі, коли вказаний тип #PCDATA може привести як до очикуванних, так і небажаних результатів.

Оскільки схема DTD забезпечує перевірку правильності структури документів, вона містить правила для змісту. Наприклад, елемент може містити текст або інші елементи, а може бути порожнім елементом. Усі ці моделі змісту подаються у схемах DTD по-різному. Схеми DTD можуть бути вказані в різних документах і бути пов’язаними в цих документах.

Виділяють внутрішні і зовнішні схеми DTD.

Внутрішні схеми DTD.

Розглянемо приклад, в якому схема DTD буде вбудована в XML-документ.

Для детального розгляду пронумеруємо рядки.

1: <?xml version= «1.0» encoding= «windows-1251»?>

2: <!DOCTYPE note [

3: <!ELEMENT note (#PCDATA)>

4:]>

5: <note> Завтра о 12.45 відбудеться лекція з Web-дизайну</note>

Рядок 1 – звичайне оголошення XML, яке використовується у всіх XML-документах.

У рядку 2 міститься оголошення певного типу документа, яке дається в пролозі документа і яке пов’язує з документом схему DTD. Визначення типу документа завжди починається з <!DOCTYPE і закінчується >. Ключове слово DOCTYPE завжди записується символами верхнього регістру.

Усі рядки, що містяться в XML-документі до кореневого елемента називаються ***прологом***. Пролог містить інструкції оброб- ки документа: оголошення XML <?xml version= “1.0” ?> і

оголошення DOCTYPE у випадку використання DTD схеми. Оголошення типу документа в другому рядку повідомляє обробника про те, що існує оголошення елемента note, а всі відомості, поміщені в квадратні дужки, є схемою DTD. Іншими словами, рядок 2 починає внутрішню підмножину DTD під назвою note.

У рядку 3 міститься оголошення типу елемента для елемента note. Тут вказано, що елемент note може містити тільки текстові дані. У дужках указується або модель змісту, або специфікація змісту. За допомогою моделі змісту парсеру повідомляється чого слід чекати від кожного елемента XML у даному документі.

Рядок 4 містить закриття схеми DTD.

Рядок 5 містить кореневий елемент XML-документа.

Елемент, що містить дочірній елемент

Нехай елемент note містить дочірній елемент. Добре оформлений документ виглядатиме так:

```
<?xml version="1.0" encoding="windows-1251" ?>
<note>
<text>Завтра о 12.45 лекція з Web-дизайну</text>
</note>.
```

Цей самий документ, що містить схему DTD, можна оформити так:

```
1: <?xml version="1.0" encoding="windows-1251" ?>
2: <!DOCTYPE note [
3: <!ELEMENT note (text)>
4: <!ELEMENT text    (#PCDATA)>
5: ]>
6: <note>
7: <text>
8: Завтра о 12.45 відбудеться лекція з Web-дизайну
9: </text>
10: </note>.
```

Тут, у рядку 3 оголошується, що елемент note містить дочірній елемент text, рядок 4 містить інформацію про те, що вмістом елемента text є текст, що інтерпретується парсером. Рядки 6–8 є добре і правильно оформленим документом.

Зауваження. Відступи в тексті використовуються виключно для надання документу зручної форми для читання.

Оголошення порожнього елемента

Порожній елемент у схемах DTD оголошується таким чином:

<! ELEMENT і'мя_елемента EMPTY>.

Припустимо, в нашому повідомленні є порожні елементи

```
<?xml version= «1.0» encoding= «windows-1251»?>
<note>
<time />
<date />
<text>Завтра о 12.45 відбудеться лекція з Web-дизайну</text>
</note>.
```

Тоді DTD схема для такого XML тексту виглядатиме так:

```
1:<?xml version= «1.0» encoding= «windows-1251»?>
2: <!DOCTYPE note [
3: <!ELEMENT note(time,datetext)>
4: <!ELEMENT time EMPTY>
5: <!ELEMENT date EMPTY>
6: <!ELEMENT text (#PCDATA)>
7: ]>
8: <note>
9: <time />
10: <date />
11: <text>
12: Завтра о 12.45 відбудеться лекція з Web-дизайну
13: </text>
14: </note>.
```

Парсер інтерпретує рядок 3 так: елемент note містить три елементи time, date, text, які слідують один за одним у такому порядку, в якому вони перелічені в дужках при описі елемента note.

Згідно зі схемою DTD елементи time і date не можуть містити будь-яких даних.

Використання ключового слова ANY

Іноді наперед не відомо, яка саме модель використовується. У таких випадках у схемах DTD використовують слово ANY.

Ключове слово ANY впливає на визначення структури XML.

```
1:<?xml version= «1.0» encoding= «windows-1251» ?>
2: <!DOCTYPE note [
3: <!ELEMENT note ANY>
4: <!ELEMENT time EMPTY>
5: <!ELEMENT date EMPTY>
6: <!ELEMENT text   (#PCDATA)>
7: ]>
8: <note>
9: <time />
10:<date
/>
11:<text>
12: Завтра о 12.45 відбудеться лекція з Web-дизайну
13: </text>
14: </note>
```

Змішаний вміст елементів

Бувають випадки, коли необхідно дозволити елементу містити деяку комбінацію текстових даних і інших елементів. Подібні можливості забезпечує модель змішаного вмісту.

Наприклад,

```
<?xml version= «1.0» encoding= «windows-1251» ?>
<note> Увага!!!!
<time />
<date />
<text>
Завтра о 12.45 лекція з Web-дизайну
```

</text>
</note>

Тут кореневий елемент note, крім порожніх дочірніх елементів, містить текст «Увага!!!»

Для визначення змішаного типу вмісту використовують таке позначення:

Тобто в даному прикладі XML-код виглядатиме так:

```
1:<?xml version= «1.0» encoding= «windows-1251»?>
2: <!DOCTYPE note [
3: <! ELEMENT note (#PCDATA | time | date | text)*>
4: <!ELEMENT time EMPTY>
5: <!ELEMENT date EMPTY>
6: <!ELEMENT text (#PCDATA)>
7: ]>
8: <note>Увага!!!
9:     <time />
10: <date />
11: <text>
12:Завтра о 12.45 відбудеться лекція з Web-дизайну
13: </text>
14: </note>.
```

Використання атрибутів

Припустимо, в процесі написання XML-коду було вирішено використовувати замість порожнього елемента date порожній елемент з атрибутами:

<date day= «» month= «» year= «» />,

а елемент time з необов’язковими атрибутами:

<time hh= «» mm= «» ss= «» />.

Наприклад,

```
<?xml version= «1.0» encoding= «windows-1251» ?>
<note> Увага!!!!
<time hh= «15» mm= «00» />
<date day= «2» month= «10» year= «2009» />
<text>
```

Завтра о 12.45 відбудеться лекція з Web-дизайну

</text>
</note>.

У схемі DTD існує спеціальний механізм визначення атрибутів з використанням ключового слова ATTLIST.

Оголошення атрибутів відбувається таким чином:

<! ATTLIST ім’я_елемента;

ім’я_атрибута1 (тип) значення за замовченням;

ім’я_атрибута1 (тип) значення за замовчанням >.

Існує **три фундаментальні типи атрибутів**, що оголошуються в рамках DTD схем:

Рядки, що зазначаються за допомогою ключового слова CDATA.

Марковані атрибути, що зазначаються за допомогою визначених раніше маркерів.

Атрибути з переліченням, що пропонують цілий ряд значень.

Існує три стандартні значення атрибутів:

#REQUIRED – вказує на те, що атрибут має бути вказаний;

#FIXED – вказує на фіксоване значення атрибута. Якщо значення атрибута відрізняється від оголошеного, документ не вважається правильним;

#IMPLIED – атрибут необов’язковий. Це означає, що при обробці елемента парсер може використовувати будь-яке значення, якщо в цьому є необхідність.

Оголошення атрибутів першого типу

Складемо DTD схему для прикладу, де позначимо, що атрибути задаються рядками.

```
<?xml version=«1.0» encoding=«windows-1251» ?>
<!DOCTYPE note [
<!ELEMENT note (#PCDATA | time | date | text)* >
<!ELEMENT time EMPTY>
<!ELEMENT date EMPTY>
<!ATTLIST date
```

```
day CDATA #REQUIRED
                month CDATA #REQUIRED
                year CDATA #FIXED «2009»>
<!ATTLIST time
hh CDATA #IMPLIED
                mm CDATA #IMPLIED
                ss CDATA #IMPLIED>
<!ELEMENT text (#PCDATA)>
        ]>
<note>Увага!!!
        <time hh=«13» mm=«00»/>
<date day=«19» month=«10» year=«2009»/>
        <text>
Завтра о 12.45 відбудеться лекція з Web-дизайну
        </text>
</note>.
```

Рядок 3 вказує на те, що кореневий елемент може містити текст #PCDATA, що інтерпретується парсером, або дочірній елемент time, або date, або text. Причому кожен з цих елементів може бути присутнім у XML-документі або бути відсутнім, але один з них повинен бути обов'язково, а можуть бути і всі одразу.

Зауваження. Зазначимо, що в моделях елементів змішаного типу вміст може перелічуватися тільки через знак «|» і повинен бути визначений або як нуль, або як множина (*). Роздільник «,» використовувати не можна.

Визначення атрибутів маркованого типу

При використанні атрибутів маркованого типу можна накладати обмеження на значення атрибутів. Наприклад, можна дозволяти атрибутам брати тільки одне з двох заданих значень.

Розглядають 4 різних типи маркованих атрибутів:

ID – унікальним чином ідентифікує об'єкт;

IDREF – вказує на елементи, що містять атрибути ID;

ENTITIES – посилання на зовнішній елемент;

NMTOKEN – містить букви, цифри, крапки, знаки підкрес- лення,

переноси і двокрапки, але не пропуски.

Використання атрибутів типів id і idref

Наприклад, у житті часто виникає ситуація, коли людина протягом дня робить позначки <note> в щоденнику. Кожному повідомленню присвоюється унікальний ідентифікатор number. Також протягом дня виникають які-небудь коментарі, які можна записувати, а можна і не записувати. Якщо виникає необхідність записати коментар <comment>, його необхідно пов'язати з відповідною позначкою.

```
<?xml version=«1.0» encoding=«windows-1251» ?>
<note>
    <date day=«29» month=«10» year=«2009»/>
    <time hh=«08» mm=«15» ss=«17» />
    <text number= «n1» from=«Проценко О.Б.»>
    Завтра відбудеться лекція з Web-дизану о 13.25</text>
<time hh=«13» mm=«15» ss=«00» />
    <text number=«n2» from=«Керівник
        відділу»> Терміново здати звіт
    </text>
<comment txt= «n1»>
    Що важливіше: лекція або звіт?
    </comment>
 <comment txt= «n1»>
        Не забути конспект
    </comment>
 </note>.
```

DTD-схема для даного документа матиме такий вигляд:

```
1:<?xml version=«1.0» encoding= «windows-1251» ?>
2: <!DOCTYPE note [
3: <!ELEMENT note (date*,time+, text+, comment+)*>
4: <!ELEMENT date EMPTY>
5: <!ATTLIST date
6: day CDATA #REQUIRED
7: month CDATA #REQUIRED
8: year CDATA #FIXED «2009»>
9:
```

10: <!ELEMENT time EMPTY>
11: <!ATTLIST time
12: hh CDATA #REQUIRED
13: mm CDATA #REQUIRED
14: ss CDATA#IMPLIED>
15:
16: <!ELEMENT text (#PCDATA)>
17: <!ATTLIST text
18: number ID #REQUIRED
19: from CDATA #REQUIRED>
20:
21: <!ELEMENT comment (#PCDATA)>
22: <!ATTLIST comment
23: txt IDREF #REQUIRED>
24:
25:]>
26: <note>
27: <date day=«29» month=«10»
year=«2009»/> 28:
29: <time hh= «08» mm= «15» ss= «17» />
30: <text number=«n1» from=«Проценко О.Б.»>
31: Завтра відбудеться лекція з Web-дизану о 13.25
32: </text>
33: <comment txt=
«n1»> 34: Не забути
конспект 35:
</comment>
36:
37: <time hh= «13» mm= «15» />
38: <text number= «n2» from= «Керівник відділу»>
39: Терміново здати звіт. Завтра останній термін
40: </text>
41: <comment txt= «n2»>
42: Що важливіше лекція або
звіт? 43: </comment>

44: </note>.

Отже, запис у рядку 3:

<!ELEMENT note (date*,time+, text+, comment+)>

Вказує на те, що елемент note повинен містити елементи date*, time+, text+, comment+, які ідуть у порядку їх опису. Причому date може використовуватися скільки завгодно разів (*) або не використовуватися взагалі, елементи time, text, comment повин- ні використовуватися мінімум один раз. Знак (*) у кінці специфі- кації елемента вказує на те, що група елементів може повторюва- тися скільки завгодно разів.

У рядках 5–8, 11–14, 17–19, 22 описуються атрибути елементів date, time, text, comment відповідно.

У рядку 18 вказується, що необхідний атрибут number має тип ID. Це *означає*, що *атрибут* number *має бути унікальним, тобто не використовуватися тільки один раз*.

У рядку 37 відсутній атрибут ss елемента time. Це пов’язано з тим, що ss був оголошений як атрибут типу IMPLIED.

Абсолютно по-іншому матиме вигляд XML-документ, якщо специфікація в рядку 3 буде такою:

```
<!ELEMENT note ((date*,time+, text+), comment+)>
<?xml version=«1.0» encoding=«windows-1251» ?>
    <!DOCTYPE note [
    <!ELEMENT note ((date*,time+, text+), comment+)>
    <!ELEMENT date EMPTY>
    <!ATTLIST date
            day CDATA #REQUIRED
            month CDATA #REQUIRED
            year CDATA #FIXED «2009»>
<!ELEMENT time EMPTY>
<!ATTLIST time
            hh CDATA #REQUIRED
            mm CDATA #REQUIRED
            ss CDATA #IMPLIED>
<!ELEMENT text (#PCDATA)>
<!ATTLIST text
            number ID #REQUIRED
            from CDATA #REQUIRED>
```

<!ELEMENT comment (#PCDATA)>
<!ATTLIST comment
txt IDREF #REQUIRED>
]>
<note>
<date day=«29» month=«10» year=«2009»/>
<time hh=«08» mm=«15» ss=«17» />
<text number=«n1» from=«Проценко О.Б.»>
Завтра відбудеться лекція з Web-дизану о 13.25
</text>
<time hh= «13» mm= «15» />
<text number=«n2» from=«Керівник відділу»>
Терміново здати звіт. Завтра останній термін
</text>
<commenttxt=«n2»>
Що важливіше: лекція або звіт?
</comment>
<comment txt= «n1»> Не забути конспект
</comment>
</note>.

А якщо вказати в рядку 3 специфікацію елемента **note**

<!ELEMENT note (date*,time+, text+)>, то для правильності документа рядки слід було б розмістити так:

<?xml version=«1.0» encoding= «windows-1251» ?>
<!DOCTYPE note [
<!ELEMENT note (date*,time+, text+, comment+)>
<!ELEMENT date EMPTY>
<!ATTLIST date
day CDATA #REQUIRED
month CDATA #REQUIRED
year CDATA #FIXED «2009»>
<!ELEMENT time EMPTY>
<!ATTLIST time
hh CDATA #REQUIRED
mm CDATA #REQUIRED
ss CDATA #IMPLIED>

```
<!ELEMENT text (#PCDATA)>
<!ATTLIST text
                number ID #REQUIRED
                from CDATA #REQUIRED>
<!ELEMENT comment (#PCDATA)>
<!ATTLIST comment
                txt IDREF #REQUIRED>
                ]>
<note>                          <date day=«29» month=«10»
                                year=«2009»/>
                                <time hh=«08» mm=«15» ss=«17»
                                />
                                <time hh=«13» mm=«15» />
                                <text number=«n1»
                                from=«Проценко О.Б.»> Завтра
відбудеться лекція з Web-дизану о 13.25</text>
        <text number=«n2» from=«Керівник
    відділу»> Терміново здати звіт. Завтра останній
    термін
        </text>
        <comment txt=«n2»>
            Що важливіше: лекція або звіт?
        </comment>
    <comment txt=«n1»>
            Не забути конспект
        </comment>
</note>.
```

Проте розглянутий в прикладі xml-код суперечить структурі документа, оскільки порушений порядок проходження елементів. Це обумовлено тим, що кожне повідомлення має свій час виникнення і саме це необхідно вказати розробникові. Помилку можна було б виправити шляхом введення в елементі time атрибута txt.

Тепер всі елементи йдуть один за одним у тому порядку, який вказаний у специфікації елемента note.

```
1:<?xml version=«1.0» encoding=«windows-1251» ?>
2: <!DOCTYPE note [
```

3: <!ELEMENT note (date*,time+,text+, comment+)>
4: <!ELEMENT date EMPTY>
5: <!ATTLIST date

6: day CDATA #REQUIRED
7: month CDATA #REQUIRED
8: year CDATA #FIXED «2009»>
9:
10: <!ELEMENT time EMPTY>
11: <!ATTLIST time
12: txt IDREF #REQUIRED
13: hh CDATA #REQUIRED
14: mm CDATA #REQUIRED
15: ss CDATA #IMPLIED>
16:
17: <!ELEMENT text (#PCDATA)>
18: <!ATTLIST text
19: number ID #REQUIRED
20: from CDATA #REQUIRED>
21:
22: <!ELEMENT comment (#PCDATA)>
23: <!ATTLIST comment
24: txt IDREF #REQUIRED>
25:
26:]>
27: <note>
28: <date day=«29» month=«10»
year=«2009»/> 29:
30: <text number=«n1» from=«Проценко О.Б.»>
31: Завтра відбудеться лекція з Web-дизану о 13.25
32: </text>
33: <text number=«n2» from= «Керівник відділу»>
34: Терміново здати звіт
35: </text>
36:
37: <time txt=«n1» hh=«08» mm=«15» ss=«17»
/> 38: <time txt=«n2» hh=«13» mm=«15» />
39:
40: <comment txt=«n1»>

41: Не забути конспект
42: </comment>
43:

44: <comment txt «n1»>
45: Що важливіше: лекція або
звіт? 46: </comment>
47: </note>.

ЛАБОРАТОРНА РОБОТА 1

Тема. Подання інформації в мережі Інтернет

Мета: розібратися в основних етапах створення власної Web-сторінки.

Хід роботи

1. Відкрити вікно браузера Microsoft Internet Explorer.
2. Використавши команди браузера «Справка –> О програм- ме», визначити версію браузера.
3. Налагодити інтерфейс браузера. Для цього можливо використовувати контекстне меню рядка меню або панелі інструментів браузера та команду «Настройка» контекстного меню. Виконання даної команди призводить до появи вікна настройки панелі інструментів браузера.

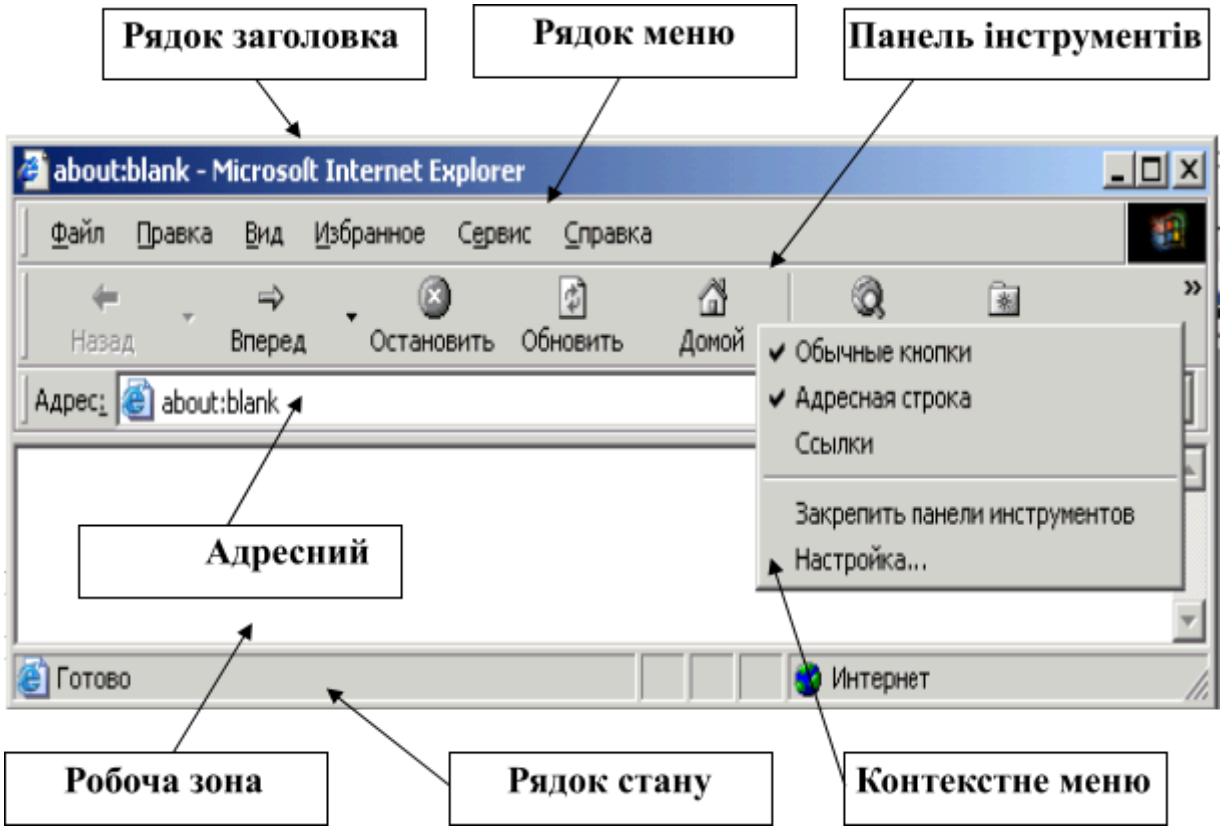


Рис. 1.1

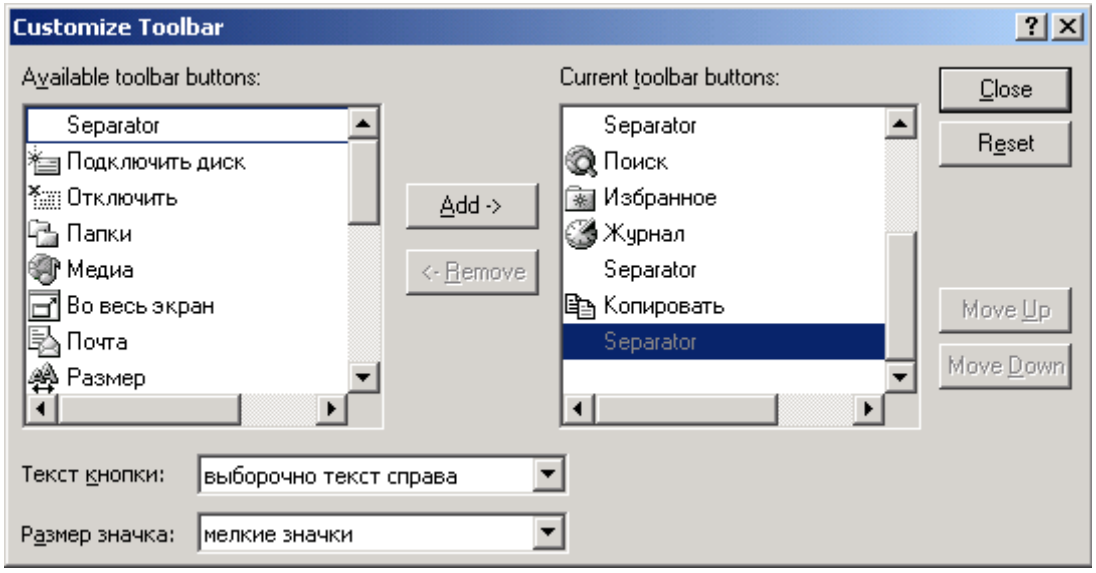


Рис. 1.2. Вікно настройки панелі інструментів браузера

4. Використавши команди «Сервис –> Свойства обозревателя» відкрити вікно «Свойства обозревателя». Перейти на вкладку «Подключения». За допомогою технічного персоналу встановити з'єднання браузера з мережею Інтернет.

5. Відвідати Web-сайти Кабінету Міністрів України (http://www.kmu.ua), Верховної Ради України (http://www.rada.gov.ua), пошукових систем Meta (http://www.meta.ua) та Google (http://www.google.com.ua). Для цього слід ввести в адресному рядку відповідну адресу. Зазначимо, що вводити http:// необов'язково.

6. За допомогою команди «Файл»–>«Создать» –> «Окно» створити нове вікно браузера та відкрити в ньому сайт Конституційного суду України (http://www.ccu.gov.ua/).

7. Ознайомитись з можливостями команди «Вид» –> «Кодировка». Після цього встановити автоматичний вибір браузером кодування Web-сторінки.

8. Ознайомитись з можливостями команди «Вид» –> «Размер шрифта». Встановити найбільш зручний для вас розмір шрифту.

9. Ще раз відвідати зазначені сайти за допомогою кнопок «Назад» та «Вперед».

10. Ознайомитись з призначенням кнопок «Обновить», «До- мой» та «Стоп».

11. Використавши кнопку «Избранное» записати адреси відвіданих сайтів у теку вибраних адрес.

12. Використавши кнопку «Журнал», здійснити навігацію по сайтах, що вже були відвідані за допомогою даного браузера.

13. Використавши команди «Сервис –> Свойства обозревателя» відкрити вікно «Свойства обозревателя». Перейти на вкладку «Общие». Це означає, що браузер буде відкриватись з порожньої сторінки, а адреси переглянутих сторінок будуть зберігатись 20 днів.

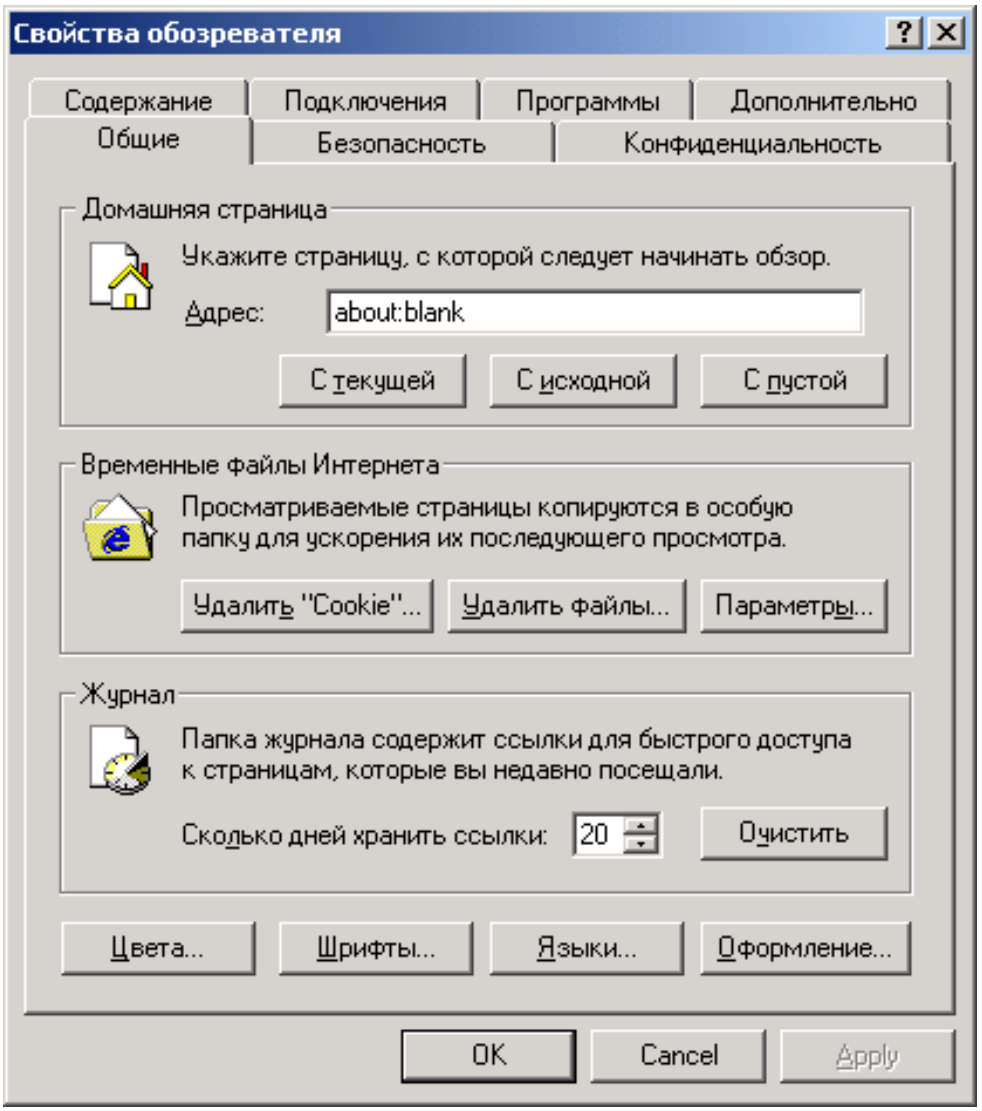


Рис. 1.3. Настройка загальних параметрів функціонування браузера

14. Використавши вкладки «Безопасность» та «Конфединциальность» вікна «Свойства обозревателя», ознайомитись з пара- метрами, що визначають безпечний перегляд різних

Web-сайтів.

15. У вікні «Свойства обозревателя» перейти на вкладку «Содержание» та натиснути кнопку «Включить» в полі «Ограничения доступа». За допомогою нового вікна настройки «Ограничения доступа» обмежити доступ до інформації, отриманої з мережі Інтернет.

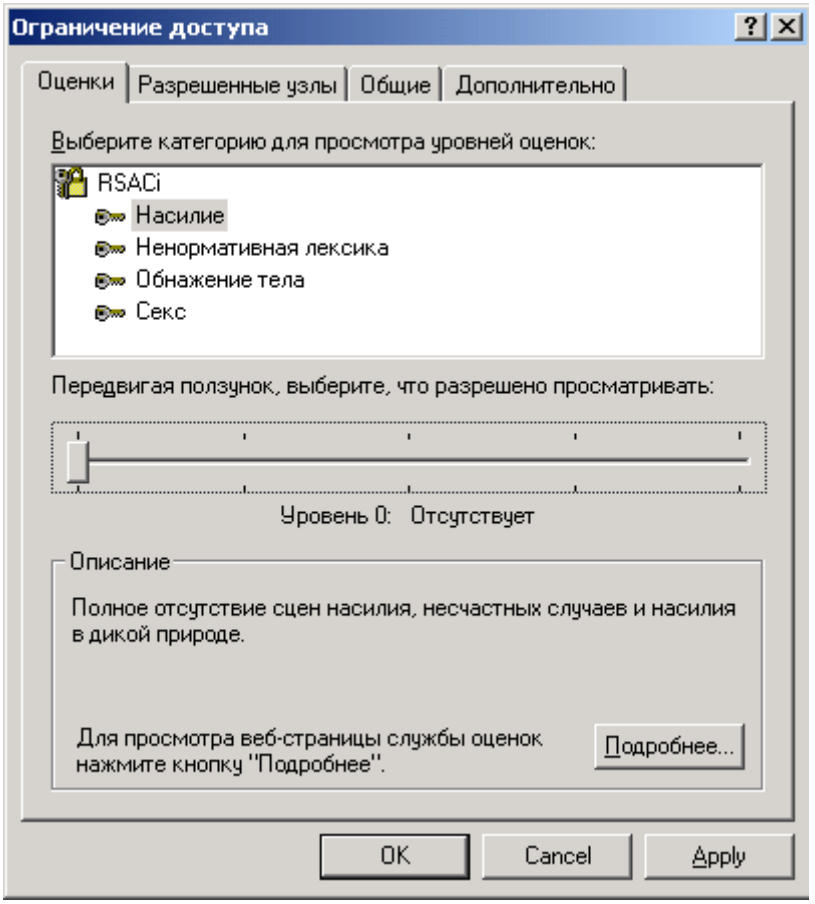


Рис. 1.4. Визначення обмежень доступу до інформації, що відображається у браузері

16. У вікні «Свойства обозревателя» перейти на вкладку «Дополнительно». Знайти розділ «Мультимедиа». Відмінити відображення рисунків у вікні браузера. Відвідати декілька вже переглянутих сайтів. Впевнитись у тому, що час потрібний для відображення сайта у вікні браузера зменшився.

17. За допомогою контекстного меню у місці, призначеному для рисунку, відобразити його на екрані.

18. Відновити відображення рисунків.

19. Освоїти можливості збереження Web-сайтів. Для цього можливо використати команди меню «Файл» –> «Сохранить

как…» Ознайомитись з різними типами файлів, що можуть бути використані при збереженні інформації із Інтернет.

20. Освоїти можливості збереження фрагментів Web сайтів.

– Рисунок можливо зберегти за допомогою команди «Сохранить рисунок как…» відповідного контекстного меню.

– Текст зберігаємо за допомогою копіювання виділеного фрагмента в будь-який текстовий редактор.

21. Записати на свій комп'ютер файл, розміщений на FTP-сервері. Для цього:

– записати в адресному рядку URL визначеного FTP-сервера. Наприклад ftp://ftp.microsoft.com;

– для збереження файла необхідно викликати контекстне меню відповідного елемента у вікні браузера та вибрати команду «Копировать в папку».

Запитання для самоперевірки

1. Яке призначення кнопки «Домой»?
2. Як зберегти тільки текстову частину Web-сторінки?
3. Як повністю зберегти Web-сторінку?
4. Як зберегти рисунок на Web-сторінці?
5. Як зберегти фрагмент тексту на Web-сторінці?
6. Як відмінити можливість відображення рисунків у вікні браузера?
7. Назвіть основні параметри безпеки браузера?
8. Як заборонити доступ до певної інформації, розміщеної в мережі Інтернет?
9. Як змінити кодування Web-сторінки?
10. Як реалізувати доступ до FTP-сервера за допомогою браузера?

ЛАБОРАТОРНА РОБОТА 2

Тема. Мова HyperText Markup Language (HTML) і його використання для структурованого подання інформації

Мета: навчитися використовувати мову розмітки гіпертексту html для структурованого подання різних видів текстової і графічної інформації.

Хід роботи

Завдання

1

Ознайомитися з теоретичним матеріалом про призначення гіпертексту, області його застосування і засоби подання гіпертекстової інформації.

Завдання 2

Вивчити синтаксис мови html, елементи, використовувані в цій мові для розмітки тексту, і їх параметри (текст, виділений синім кольором). Для отримання додаткової інформації рекомендується використовувати інформаційні ресурси Інтернет.

Завдання 3

Виконати навчальні вправи 1–16 (за необхідності) і практичні завдання, результати зберегти.

Завдання 4

Оформити звіт про виконання лабораторної роботи у вигляді документа MS Word.

Завдання 5

Відповісти на контрольні питання, відповіді включити в звіт з лабораторної роботи.

1. Викличте текстовий редактор БЛОКНОТ, що знаходиться у групі «Стандартні». Наберіть такий нижче текст і

збережіть під ім’ям web1.html в особистій папці.

```
<HTML>
<HEAD>
<TITLE>Моя персональна сторінка (заголовок)</TITLE>
</HEAD>
<BODY>
ПРИВІТ! Це моя особиста домашня сторінка! (Укажіть своє прізвище).
</BODY>
</HTML>
```

Викличте тепер Internet Explorer. Використовуючи меню FILE (Файл), команду Open File (Відкрити), відкрийте свій файл web1.html і зазначте, як він буде виглядати в браузері.

2. Відкрийте файл web1.html у БЛОКНОТІ і змініть його так, як показано нижче. Після редагування файл web1.html збережіть під ім'ям web2.html.

```
<HTML>
<HEAD>
<TITLE>Моя персональна сторінка (заголовок)</TITLE>
</HEAD>
<BODY>
<H1 ALIGN = Center> ПРИВІТ! Це моя особиста домашня сторінка! Мене кличуть (Укажіть своє прізвище, ім'я і по батькові).
</H1>
<hr>
<H2> Моє місто </H2>
<p>Я ...
<H3 ALIGN =Left> Моє навчання </H3>
<p align = left> Я учуся в.....
<p align = center> Мої захоплення - ……
<p align = right> Мої колеги       </p>
<br>
<HR>
<H6 ALIGN = Center> Design «BelSUT» </H6>
</BODY>
</HTML>
```

3. Створіть за допомогою графічного редактора рисунок і збережіть його в тому каталозі, що і документи web.html і web1.html. Відкрийте файл **web2.html** у БЛОКНОТІ і відредагуйте його, як показано нижче. Після редагування файла **web2.html** збережіть під ім'ям **web3.html**

<HTML>
<HEAD>
<TITLE> Моя персональна сторінка </TITLE>
</HEAD>
<BODY>
<H1 ALIGN = Center> ПРИВІТ! Це моя особиста домашня сторінка! Мене кличуть (Укажіть своє прізвище, ім'я і по батькові).
</H1>
<hr>
<H2> Моє місто </H2>
Я живу в <FONT SIZE = +2>Києві</FONT>
<H3> Моя робота </H3>
 Я працюю в.....
<H4> Мої захоплення </H4>
 Мої захоплення -<B>музика</B>, спорт.
<H5 > Мої колеги </H5>
<U> Мої колеги</U> в усьому мені <I>допомагають</I>.
<Img SRC=«*ім'я малюнка*.jpg»>
<H6 ALIGN = Center> Design «NIHE» </H6>
</BODY>
</HTML>

4. Створіть такий HTML-документ:

<HTML>
<HEAD>
<TITLE>Поради по виготовленню сторінок</TITLE>
<HEAD>
<BODY>
<PRE**>**

Гарні поради!!!!!

– Використовуйте відповідну мову.

– Не перевантажуйте сторінку великими зображеннями.
– Перевіряйте якнайчастіше посилання на вашій сторінці.
</PRE>
</BODY>
</HTML>

Збережіть даний документ під ім'ям **web4.html**. Порівняйте написаний вами текст (між тегами <PRE> і </PRE>) із зображенням на екрані, отриманим за допомогою програми перегляду.

1. Створіть документ, у якому текст буде організований за допомогою списків трьох видів (табл. 2.1), і збережіть під ім'ям web5.html. Оператори HTML для опису списку:

Таблиця 2.1

Нумерований список – Ordered List	**Маркірований список – Unordered List**	**Вкладений список – Nested List**
** ** ** ** ** **	** ** ** ** ** **	** ** ** ** ** ** ** ** ** ** ** ** ** **

```
<HTML>
<HEAD>
<TITLE>Сторінка зі списками</TITLE>
</HEAD>
<BODY>
<H2 ALIGN = Left>Сторінка зі списками</H2>
<ol>
<li>По можливості намагайтеся уникати піктограми «У процесі виробництва»;
<li>Використовуйте відповідну мову;
</ol>
<ul>
<li>Не перевантажуйте сторінку великими зображеннями;
```

Перевіряйте якнайчастіше посилання на вашій сторінці.

Перед тим як покласти готову сторінку на сервер випливає:

Перевірити граматичні помилки;
Переглянути свою сторінку в різних програмах перегляду.

Для реклами сторінки необхідно:

</BODY>
</HTML>

6. Створіть у текстовому редакторі файл web6.html наступ- ного змісту:

<HTML>
<HEAD>
<TITLE>Терміни </TITLE>
</HEAD>
<BODY>
<dl>
<dt>Web Server
<dd> Web-сервер. Сервер, що зберігає HTML-документи і інші інформаційні ресурси Інтернет з використанням протоколу HTTP. Його називають також HTTP-сервером
<dt>HOME PAGE
<dd> «Домашня сторінка». Головна, початкова сторінка, локальний архів. Перша сторінка якого-небудь чи Web-сервера логічно зв’язаної групи HTML-документів
</dl>
</BODY>
</HTML>

7. Створіть у текстовому редакторі HTML-документ **web7.html**, використовуючи параметри для тега <BODY

Примітка. Колір указується шістнадцятирічним числом і має наступну структуру: RED – GREEN – BLUE чи RGB (червоний – зелений – синій). На кожний колір приділяється 256 значень, що означають присутність того чи іншого компонента в

результуючому кольорі. Наприклад, максимум червоного – FF, інші два кольори – 00, результат – «#FF0000»

```
<HTML>
<HEAD>
<TITLE>Моя персональна сторінка </TITLE>
</HEAD>
<BODY bgcolor=«#FFFFFF»>
<p>На даній сторінці текст і зображення будуть розташовуватися на білому фоні.
</p>
<p>Шрифт може бути
<FONT COLOR=«#0000A0»>різного</FONT> кольору.
</p>
</BODY>
</HTML>
```

8. Створіть за допомогою графічного редактора файл з ім’ям pic2.jpg, у якому будуть знаходитися «обої» – фон для документа. Створіть файл з ім’ям web8.html наступного змісту:

```
<HTML>
<HEAD>
<TITLE>Моя персональна сторінка </TITLE>
</HEAD>
<BODY background =pic2.jpg>
<p><FONT SIZE=+3><b> На даній сторінці текст і зображення будуть розташовуватися на фоні</b>
</FONT></p>
</BODY>
</HTML>
```

9. Відкрийте гіпертекстовий документ web8.html у текстовому редакторі, вставте в нього графічне зображення з файла ag00001_.gif визначеного розміру і збережіть під ім’ям web9.html.

Примітка. Графічне зображення може мати кілька додаткових параметрів:

Width –
ширина;

Height – висота;

Align – вирівнювання
Absmiddle чи center –
центрування; Left – по лівому
краю поля;
Right – по правому краю поля.

Примітка. Якщо графічне зображення потрібно розташувати у середині параграфа, то його можна вирівняти:
по лівому полю – ALIGN=left;
по правому полю – ALIGN=right;
по верхній границі – ALIGN=top;
по нижній границі – ALIGN=bottom;
по центрі – ALIGN=middle чи center.

```
<HTML>
<HEAD>
<TITLE>Моя персональна сторінка </TITLE>
</HEAD>
<BODY background =pic2.jpg>
<p><FONT SIZE=+3><b> На даній сторінці текст і зображення
будуть розташовуватися на фоні «шпалери»</b>
</FONT></p>
<center><img src=«ag00001_.gif «></center>
<p>Зображення в натуральну величину – WIDTH = 150 pix.,
HEIGHT=100 pix.
</BODY>
</HTML>
```

10. Відкрийте в текстовому редакторі файл web9.html і збережіть його під ім’ям web10.html. У новому документі змініть розміри рисунка в два рази.

```
<HTML>
<HEAD>
<TITLE>Моя персональна сторінка </TITLE>
</HEAD>
<BODY background =pic2.jpg>
<p><FONT SIZE=+3><b> На даній сторінці текст і зображення
будуть розташовуватися на фоні «шпалери»</b>
```

</FONT></p>

<img src=«pic3.jpg» width=250 height=150>
<p>Зображення збільшене в три рази – WIDTH = 250 pix., HEIGHT=150 pix.
</BODY>
</HTML>

Подивіться, як виглядає зображення на екрані.

Примітка. На розмір графічного файла в байтах не накладається ніяких обмежень, але необхідно пам’ятати, що передача великого графічного (та й текстового) файлу може зайняти дуже багато часу, особливо при низькій пропускній здатності мережі. Пропонуючи файл розміром більш ніж 100 кбайт, рекомендуємо вам попередити про це одержувача, указавши довжину файла в гіперпосиланні.

11. Створіть новий гіпертекстовий документ наступного змі- сту, збережіть його у файлі під ім’ям web11.html і перевірте його програмою перегляду:

<HTML>
<HEAD>
<TITLE>Welcome to our web-site</TITLE>
</HEAD>
<BODY bgcolor =«#F0F0F0»>
<H3 align=center>Запрошуємо Вас відвідати наш Web-сервер! </H3>
<img src=«pic3.jpg
« align=left hspace=70>
<p>Слово «Інтернет» часто зустрічається.
<p>Сьогодні про Інтернет говорять як про засіб, що змінюж не тільки умови життя, але і світогляд людини.
<p>Зароджується нове покоління, що зможе спілкуватися в новому інформаційному просторі, чи в новій віртуальній реальності.
</BODY>
</HTML>

12. Підготуйте в текстовому редакторі документ наступного виду з табличними елементами, використовуючи атрибути для тега TABLE (таблиця) і інших тегів, і збережете його під ім’ям **web12.html.**

```
<HTML>
<HEAD>
<TITLE>Таблиця</TITLE>
</HEAD>
<BODY>
<H2 ALIGN =Center> Простий варіант таблиці</H2>
<TABLE BORDER>
<TR><TD>комірка A :-)</TD>
<TD>комірка таблиці B :-)</TD>
<TD>комірка C :-)</TD></TR>
<TR><TD>комірка D :-)</TD>
<TD>комірка таблиці E :-)</TD>
<TD>комірка F :-)</TD></TR>
</TABLE>
</BODY>
</HTML>
```

13. Відкрийте в текстовому редакторі файл **web12.html** і за- дайте заголовки для рядків і стовпців. Збережіть змінений файл під ім'ям **web13.html**.

```
<HTML>
<HEAD>
<TITLE>Таблиця</TITLE>
</HEAD>
<BODY>
<H2 ALIGN =Center> Простий варіант таблиці заголовками</H2>
<TABLE BORDER>
<TR><TH>Заголовок 1</TH><TH> Заголовок 2</TH><TH> Заго-
ловок 3</TH></TR>
<TR><TD>комірка A :-)</TD>
<TD>комірка таблиці B :-)</TD>
<TD>комірка C :-)</TD></TR>
<TR><TD>комірка D :-)</TD>
<TD>комірка таблиці E :-)</TD>
<TD>комірка F :-)</TD></TR>
</TABLE>
```

```
</BODY>
</HTML>
```

14. У текстовому редакторі створіть HTML-документ нижченаведеного змісту і збережіть його у вигляді файла під ім'ям **web14.html**.

```
<HTML>
<HEAD>
<TITLE>Таблиця</TITLE>
</HEAD>
<BODY>
<H2 ALIGN =Center>Таблиця</H2>
<TABLE BORDER>
<TR>
<TD ALIGN=Center ROWSPAN=2 COLSPAN=2>Великий комір- ка
A</TD>
<TD>Маленька комірка 1</TD>
<TD>Маленька комірка 2</TD>
</TR>
<TR>
<TD>Маленька комірка 3</TD>
<TD>Маленька комірка 4</TD>
</TR>
<TR>
<TD ALIGN=Center ROWSPAN=2 COLSPAN=2>Велика комірка
C</TD>
<TD ALIGN=Center ROWSPAN=2 COLSPAN=2>Велика комірка
D</TD>
</TR>
<TR></TR>
</TABLE>
</BODY>
    </HTML>
```

15. Створіть такий документ, збережіть його під ім'ям **web15.html**.

```
<HTML>
<HEAD>
```

<TITLE>Таблиця</TITLE>
</HEAD>
<BODY>
<H2 ALIGN =Center>Таблиця</H2>
<TABLE BORDER WIDTH=«50 %»>
<TR>
<TD WIDTH=80 %>Таблиця, ширина якої 50 % ширини екрана. Стовпець, ширина якого 80 % ширини таблиці</TD>
<TD>комірка 2</TD>
</TR>
<TR>
<TD>комірка 3</TD>
<TD>комірка 4</TD>
</TR>
</TABLE>
</BODY>
</HTML>

16. Створіть документ із гіперпосиланнями на раніше ство- рені документи і збережіть його.

<HTML>
<HEAD>
<TITLE>Моя персональна сторінка (заголовок)</TITLE>
</HEAD>
<BODY>
Щоб перейти на документ із красивим фоном, клацніть СЮДИ.

А щоб перейти на документ із таблицями, клацніть СЮДИ .

Нарешті, щоб перейти на документ із фреймами, клацніть СЮДИ.
</BODY>
</HTML>

17. Створіть HTML-документ, що буде мати таку структуру: Документ складається з 3 сторінок:

1-ша сторінка

Заголовок – назва документа (розташувати по центру, виділити кольором) – наприклад, «лабораторна робота».

Зображення (розташувати по центру).

Текст – теоретичні дані з попередньої лабораторної роботи (вирівнювання по краях, кожний параграф – окремим кольором.

Таблиця, що описує властивості пошукових систем. 1-й стовпчик – зображення з пошукових систем. Текст таблиці – більш дрібний порівняно з основним.

	зображення	пошукова система	запит	час пошуку	кількість знайдених документів
1.	Rambler	www.rambler.ru	«прогноз погоди на вівторок»	0.3 хв	95
2.	YAHOO!	www.yahoo.com	«програма»	0.5 хв	115

Кнопки з написами «кнопка1», «кнопка2» для вибору ко- льору тіла.

Поле, у яке можна вводити текст.

Поле, у якому можна вибрати один з варіантів (зима, весна, літо, осінь).

Посилання, що вказують:

1 – на сторінку, що містить нумерований список відповідей на контрольні питання 1-ї лабораторної роботи;

2 – на сторінку, що містить нумерований список відповідей на контрольні питання цієї лабораторної роботи.

Обидві ці сторінки мають заголовки і посилання, що дозволяють повернутися на 1-ий сторінку.

2- га сторінка містить нумерований список відповідей на контрольні питання 1-ї лабораторної роботи.

3- тя сторінка містить нумерований список відповідей на контрольні питання цієї лабораторної роботи.

Запитання для самоперевірки

1. Що таке браузер?
2. Які елементи містить html-документ?

3. Чим відрізняються парні теги від непарних?
4. Які елементи html-документа є обов’язковими?
5. Які списки і як саме можна подавати в html-документах?
6. Як вставити в html-документ графічні зображення?

ЛАБОРАТОРНА РОБОТА 3

Тема. Використання рисунків, відео та звуку на HTML-сторінці

Мета: освоїти методику використання рисунків на HTML-сторінці.

Хід роботи

1. Копіюємо у теку HTML видані викладачем графічні фай- ли 1.jpg та 1a.gif.
2. У папці HTML створюємо текстовий документ з назвою picture.html, відкриваємо його за допомогою браузера та переходимо до редагування HTML-коду.
3. Створимо заготовку для HTML-коду Web-сторінки:

```
<html>
<head>
<title>Рисунки</title>
<META http-equiv=Content-Type content=«text/html;
charset=windows-1251»>
</head>
<body>
</body>
</html>
```

4. Додамо HTML-код для відображення на сторінці 4 однакових рисунків з різними розмірами:

```
<body>
<h2 align=center>Зміна розмірів рисунків</h2>
<img src=«1.jpg»>
<img src=«1.jpg» width=«260» height=«220»>
<img src=«1.jpg» width=«130» height=«110»>
<img src=«1.jpg» width=«300» height=«250»>
</body>
</html>
```

Вигляд сторінки у браузері повинен відповідати рис. 3.1.

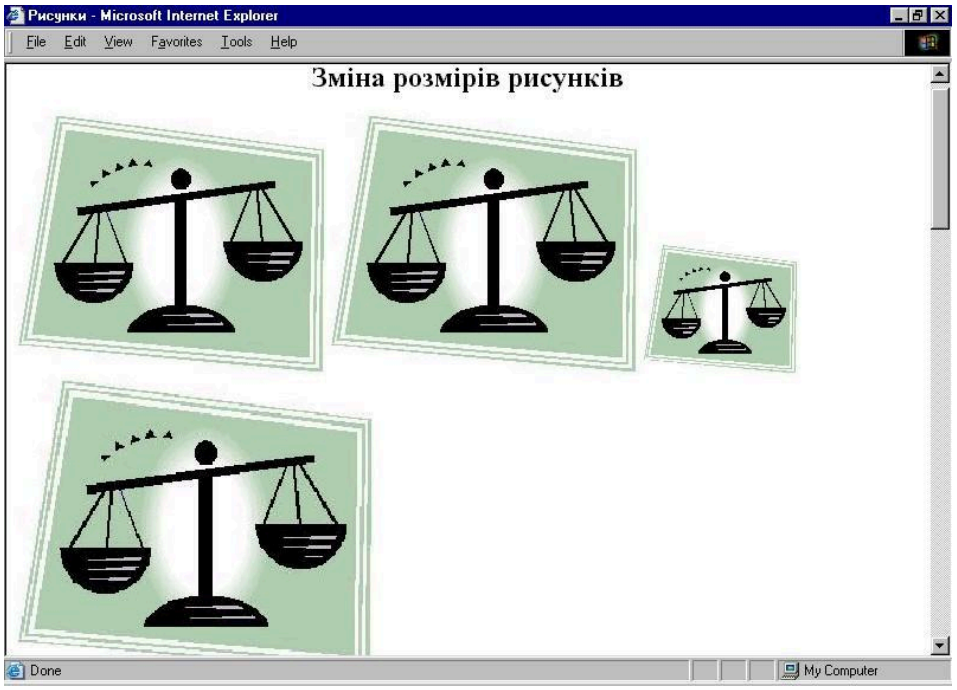


Рис. 3.1. Рисунки з різними розмірами

5. Розглянемо можливість використання границь навколо рисунків. Для цього додамо такий HTML-код:

```
...
<h2 align=center>Границі рисунків</h2>
<img src=«1.jpg» width=«130» height=«110» >
<img src=«1.jpg» width=«130» height=«110» border=«1»>
<img src=«1.jpg» width=«130» height=«110» border=«3»>
</body>
```

Відповідне відображення у вікні браузера подано на рис. 3.2.

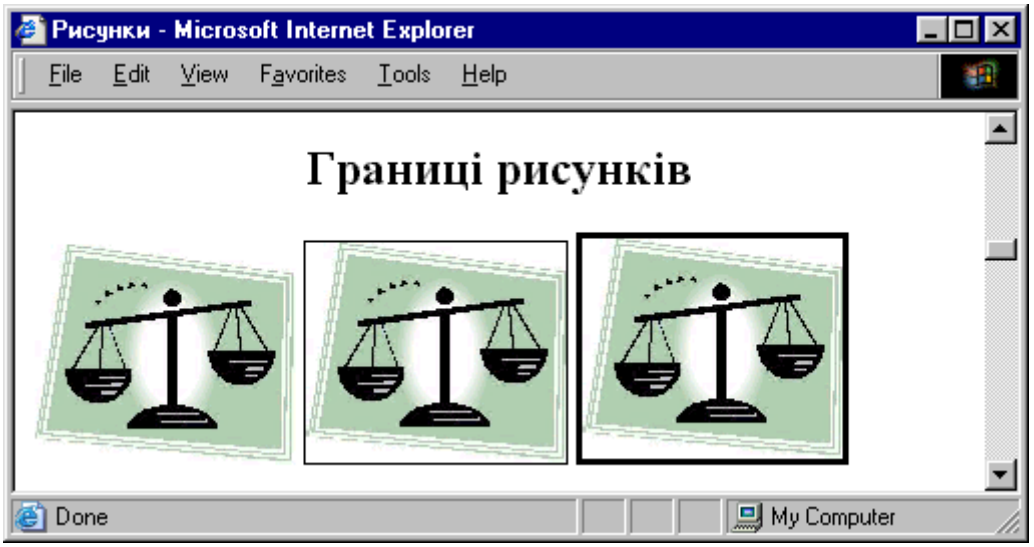


Рис. 3.2. Використання границь рисунків

6. Розглянемо можливість визначення альтернативного тек- сту у тегах рисунків. Для цього модифікуємо останній фрагмент HTML-коду:

```
...
<img src=«1.jpg» width=«130» height=«110» alt=«Рисунок.
Границь нема»>
<img src=«1.jpg» width=«130» height=«110» border=«1»
alt=«Рисунок. Товщина границь 1 піксель»>
<img src=«1.jpg» width=«130» height=«110» border=«3»
alt=«Рисунок. Товщина границь 3 пікселі»>
</body>
```

Відключимо показ рисунків у вікні браузера та проведемо оновлення нашої сторінки (рис. 3.3).

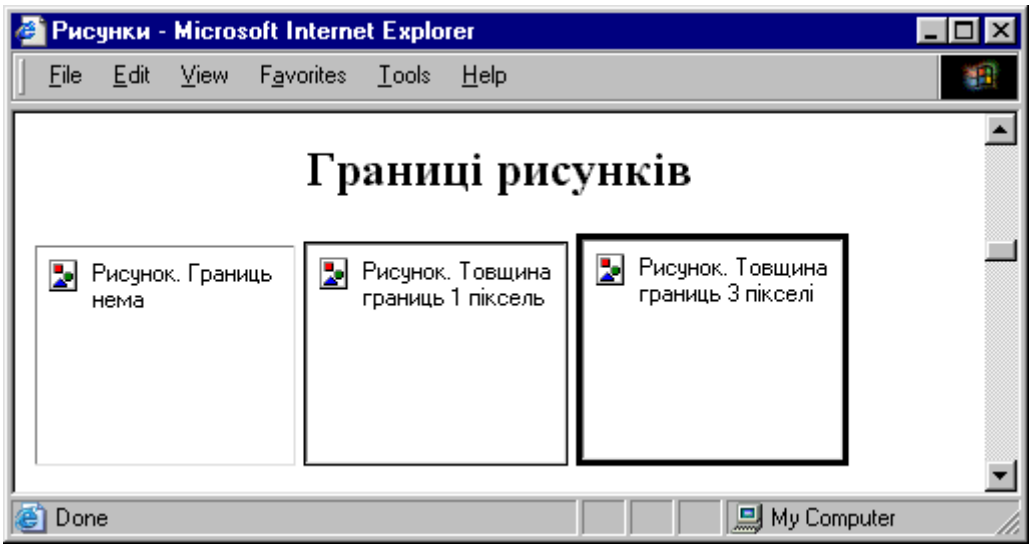


Рис. 3.3. Відображення альтернативного тексту при пе- регляді сайта з відключеним показом рисунків

7. Відновимо показ рисунків у браузері та проведемо онов- лення нашої HTML-сторінки.

8. Розглянемо можливості вертикального вирівнювання ри- сунків стосовно тексту.

9.1. Додамо HTML-код для визначення тексту та рисунків, вирівняних відносно верхньої межі:

```
...
<h2 align=center>Вирівнювання рисунків відносно тексту </h2>
```

```
<h3 align=center>Вирівнювання рисунків відносно верхнього ме- жі рядка </h3>
<p>
<img src=«1.jpg» width=«130» height=«110» border=«3» alt=«Вага»>
<img src=«1.jpg» width=«65» height=«55» border=«3» alt=«Вага» align=«top»>
По верхній межі найвищого елемента рядка
</p>
<p>
<img src=«1.jpg» width=«130» height=«110» border=«3» alt=«Вага»>
<img src=«1.jpg» width=«65» height=«55» border=«3» alt=«Вага» align=«texttop»>
По верхній межі найвищого текстового елемента рядка
</p>
</body>
```

Відповідне вікно браузера зображено на рис. 3.4.

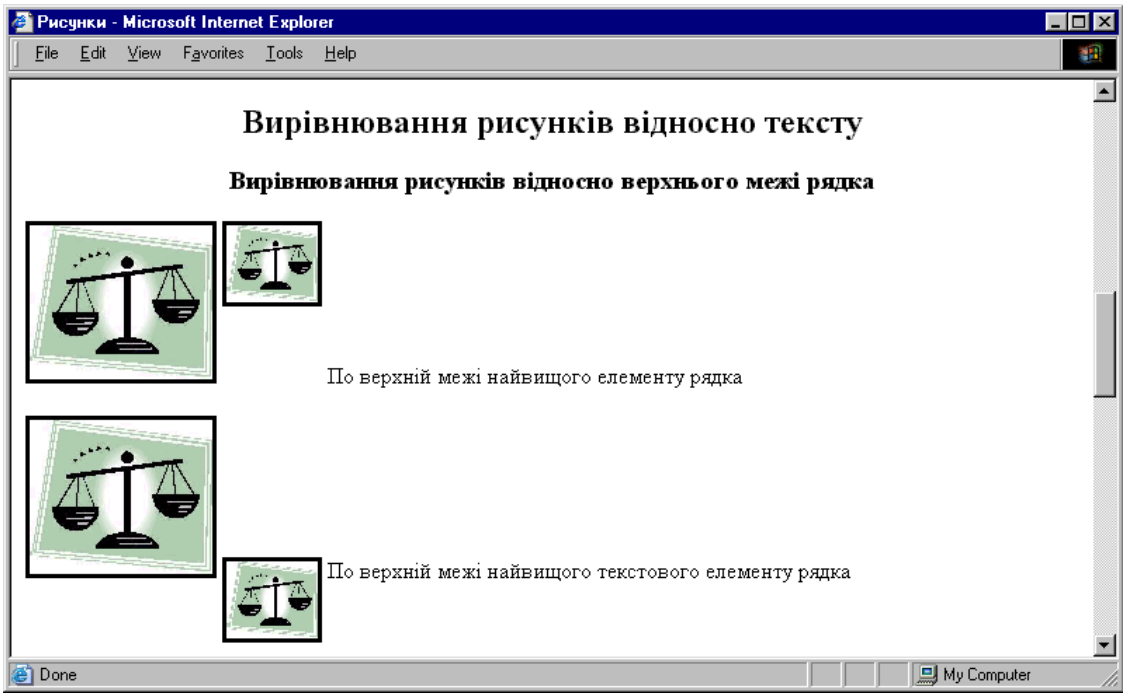


Рис. 3.4. Вирівнювання рисунків відносно верхньої межі

9.2. Додамо HTML-код для визначення тексту та рисунків, вирівняних відносно середини рядка:

```
...
<h3> Вирівнювання рисунків відносно середини рядка </h3>
<p>
<img src=«1.jpg» width=«130» height=«110» border=«3»
alt=«Вага»>
<img src=«1.jpg» width=«65» height=«55» border=«3» alt=«Вага»
align=«middle»>
По базовій лінії рядка
</p>
<p>
<img src=«1.jpg» width=«130» height=«110» border=«3»
alt=«Вага»>
<img src=«1.jpg» width=«65» height=«55» border=«3» alt=«Вага»
align=«absmiddle»>
По середині рядка
</p>
</body>
```

Відповідне вікно браузера подане на рис. 3.5.

Рис. 3.5. Вирівнювання рисунків відносно середини рядка

9.3. Додамо HTML-код для визначення тексту та рисунків, вирівняних відносно нижньої межі:

```
...
<h3 align=center> Вирівнювання рисунків відносно нижнього краю рядка </h3>
<p>
<img src=«1.jpg» width=«130» height=«110» border=«3» alt=«Вага»>
<img src=«1.jpg» width=«65» height=«55» border=«3» alt=«Вага» align=«bottom»>
Вирівнювання рисунків по базовій лінії рядка
</p>
<p>
<img src=«1.jpg» width=«130» height=«110» border=«3» alt=«Вага»>
<img src=«1.jpg» width=«65» height=«55» border=«3» alt=«Вага» align=«absbottom»>
Вирівнювання рисунків по нижній межі рядка
</p>
</body>
```

Відповідне вікно браузера подане на рис. 3.6.

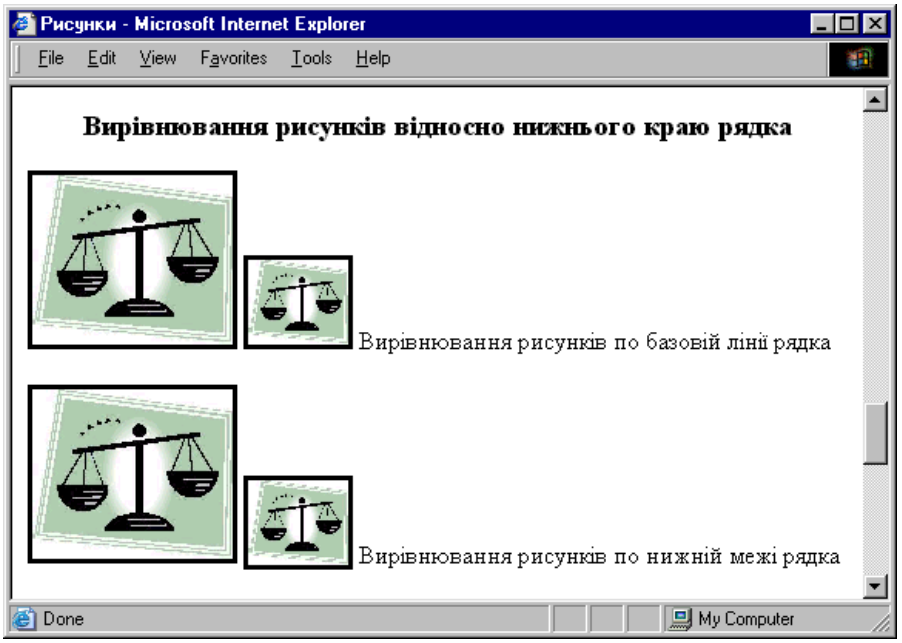


Рис. 3.6. Вирівнювання рисунків відносно нижньої межі рядка

9. Розглянемо можливість визначення плаваючих рисунків, тобто рисунків, вирівняних по лівому або правому краю рядка. Для цього додамо HTML-код:

```
...
<h3 align=center>«Плаваючі» рисунки</h3>
<p>
<img src=«1.jpg» width=«130» height=«110» border=«3» alt=«Вага»
align=«left»>
По лівому краю рядка
</p>
<br><br><br><br><br>
<p>
<img src=«1.jpg» width=«130» height=«110» border=«3» alt=«Вага»
align=«right»>
По правому краю рядка
</p>
</body>
```

Відповідне вікно браузера подано на рис. 3.7.

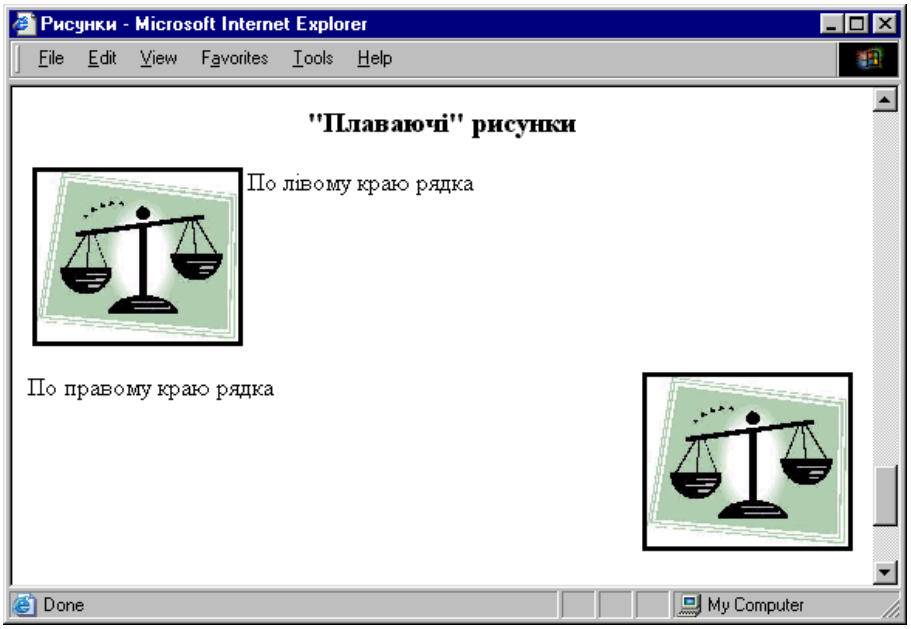


Рис. 3.7. Плаваючі рисунки

10. Розглянемо можливості відокремлення тексту від рисунків за рахунок визначення горизонтальних та вертикальних відступів. Для цього додамо HTML-код:

```
...
```

<h2 align=«center»>Відокремлення рисунків від тексту</h2>
<img src=«1.jpg» width=«130» height=«110» border=«3»
hspace=«20» vspace=«20» align=«top» alt=«Вага»>
Відстань до тексту 20 пікселів

<img src=«1.jpg» width=«130» height=«110» border=«3»
hspace=«1» vspace=«1» align=«top» alt=«Вага»>
Відстань до тексту 1 піксель
</body>

Відповідне вікно браузера зображено на рис. 3.8.

Рис. 3.8. Відокремлення тексту від рисунків за рахунок відступів

11. Додамо HTML-код для визначення альтернативного зоб- раження:

...
<h2>Використання мініатюр</h2>
<img src=«1.jpg» lowsrc=«1a.gif» width=«130» height=«110»
border=«3» alt=«Вага» >
</body>

12. Розглянемо можливість використаня на сайтах відеозображень. Для цього необхідно у папку HTML вставити файл з відео, наприклад *CLOCK.AVI,* та додати у файл picture.html HTML-код:

...
<h2>Використання відеозображень</h2>
<img width=«100» height=«100» border=«2»
dynsrc=«CLOCK.AVI»>
</body>

Моментальний знімок перегляду відео з файла показано на рис. 3.9.

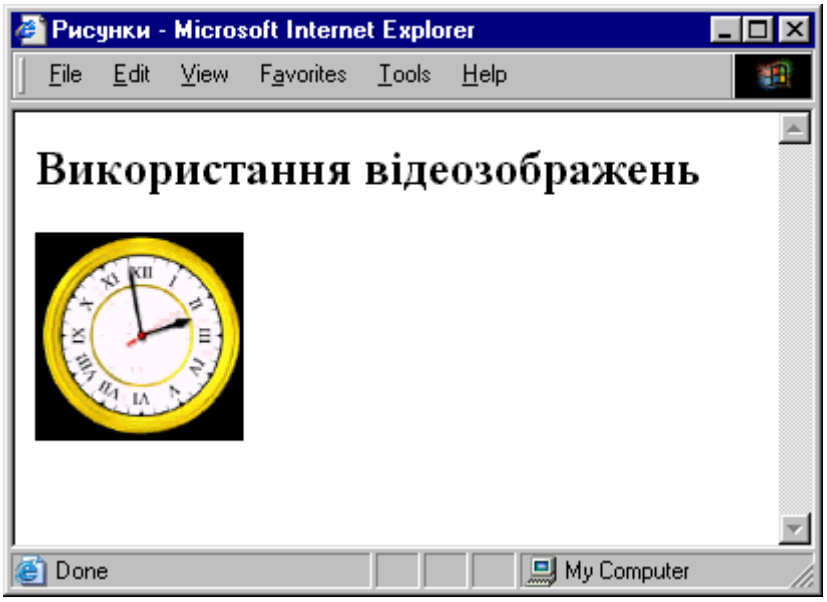


Рис. 3.9. Перегляд відео у вікні браузера

13. Модифікуємо HTML-код для прокрутки відеоролика *CLOCK.AVI* два рази:
<img width=«100» height=«100» border=«2» dynsrc=«CLOCK.AVI» loop=«2»>

Провести оновлення вікна браузера.

14. Модифікуємо HTML-код для прокрутки відеоролика *CLOCK.AVI* два рази, при цьому показ починався після наведення миші на рисунок:
<img width=«100» height=«100» border=«2» dynsrc=«CLOCK.AVI» loop=«2» start=«mouseover»>

Провести оновлення вікна браузера.

15. Модифікуємо HTML-код для того, щоб відеоролик *CLOCK.AVI* прокручувався постійно:

<img width=«100» height=«100» border=«2» dynsrc=«CLOCK.AVI» loop=«-1»>

Провести оновлення вікна браузера.

16. Розглянемо можливість використання на сайтах звуку. Для цього необхідно у папку HTML вставити звуковий файл, наприклад *START.WAV* та додати у файл picture.html HTML-код:

...

<embed src=«START.WAV» hidden=«true» autostart=«true» loop=«True»>

</body>

Провести оновлення вікна браузера.

18. Оформити у вигляді HTML-документа фотоальбом з те- матики, вказаної викладачем.

19. Оформити звіт.

Запитання для самоперевірки

1. Як визначити розмір вертикального відступу від тексту до границі рисунка?
2. Як визначити розмір горизонтального відступу від тексту до границі рисунка?
3. У чому полягає специфіка плаваючого вирівнювання рисунків?
4. Навіщо потрібно записувати альтернативний текст при визначенні рисунків?
5. Навіщо потрібні мініатюри при визначенні рисунків?
6. Як вирівняти рисунок по верхньому краю рядка?
7. У чому полягає різниця між вирівнюванням рисунка bottom та absbottom?
8. У чому полягає різниця між вирівнюванням рисунка absmiddle та middle?
9. У чому полягає різниця між вирівнюванням рисунка texttop та top?
10. Як визначити товщину границі рисунка?

ЛАБОРАТОРНА РОБОТА 4

Тема. Створення гіперпосилань
Мета роботи: навчитись створювати гіперпосилання.

Хід роботи

1. Скопіювати у папці HTML видані викладачем графічні файли Eifel_small.jpg, Eifel.jpg, Flower.jpg.
2. У папці HTML створимо текстовий документ з назвою Eifel.html та запишемо в ньому такий HTML-код:

```
<html>
<head>
<title>Ейфелева вежа</title>
</head>
<body>
<img src=«Eifel.jpg» height=«300» align=«left»>
</body>
</html>
```

3. У папці HTML створимо текстовий документ з назвою Link.html та запишемо у ньому такий HTML-код:

```
<html>
<head><title>Створення гіперпосилань</title></head>
<body>
```

У Франції є дуже багато визначних пам’яток. Спочатку ми відвідаємо Ейфелеву вежу.

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

4. Визначимо у документі Link.html текстове гіперпосилан- ня на документ Eifel.html, розміщений у одному каталозі з фай- ломLink.html. Для цього модифікуємо HTML-код та переглянемо документ (рис. 4.1):

```
...
```

У Франції є дуже багато визначних пам’яток.

Спочатку ми відвідаємо Ейфелеву ве- жу

</body>

Гіперпосилання буде завантажувати файл Eifel.html у те саме вікно браузера. Результат виконання гіперпосилання показаний на рис. 4.2.

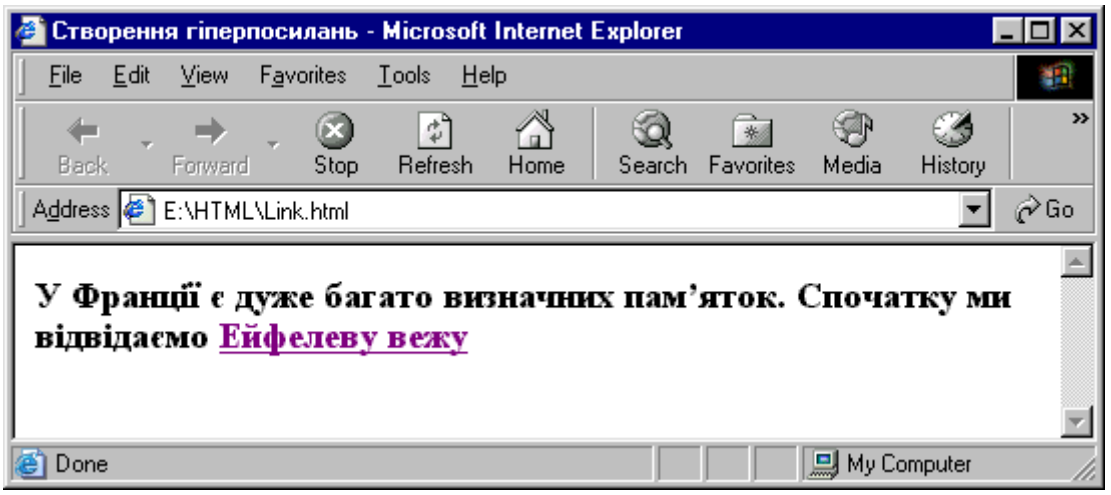


Рис. 4.1. Відображення текстового гіперпосилання у вікні браузера

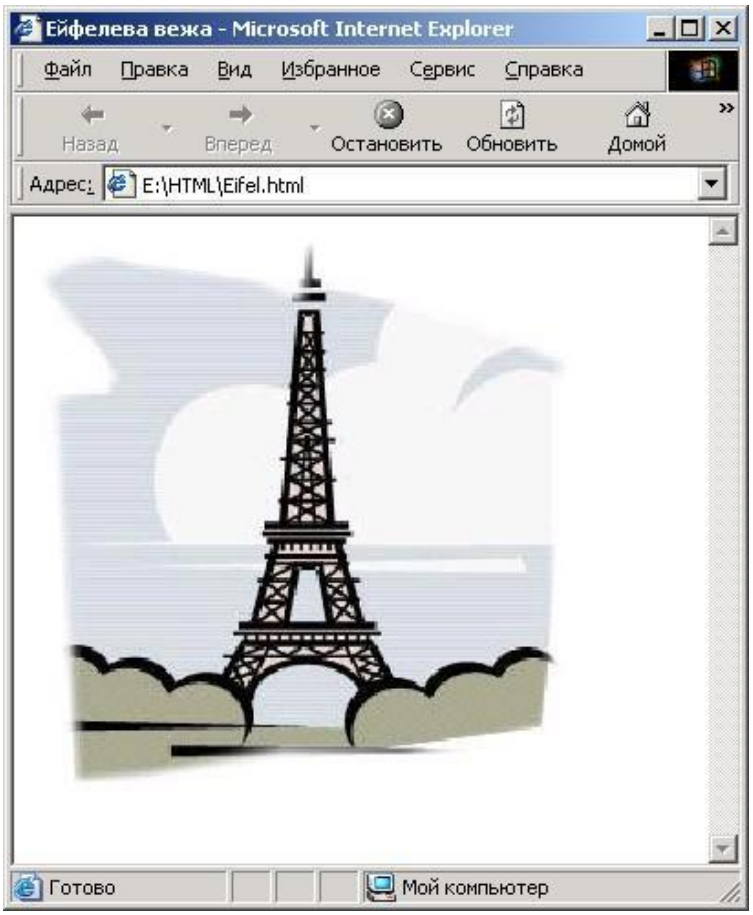


Рис. 4.2. Результат, отриманий при реалізації текстового гіперпосилання

5. Визначимо у документі Link.html графічне гіперпосилан- ня на документ Eifel.html. Додамо для цього відповідний HTML- код та переглянемо документ Link.html (рис. 4.3):

...

<img src=«Eifel_small.jpg»>

</body>

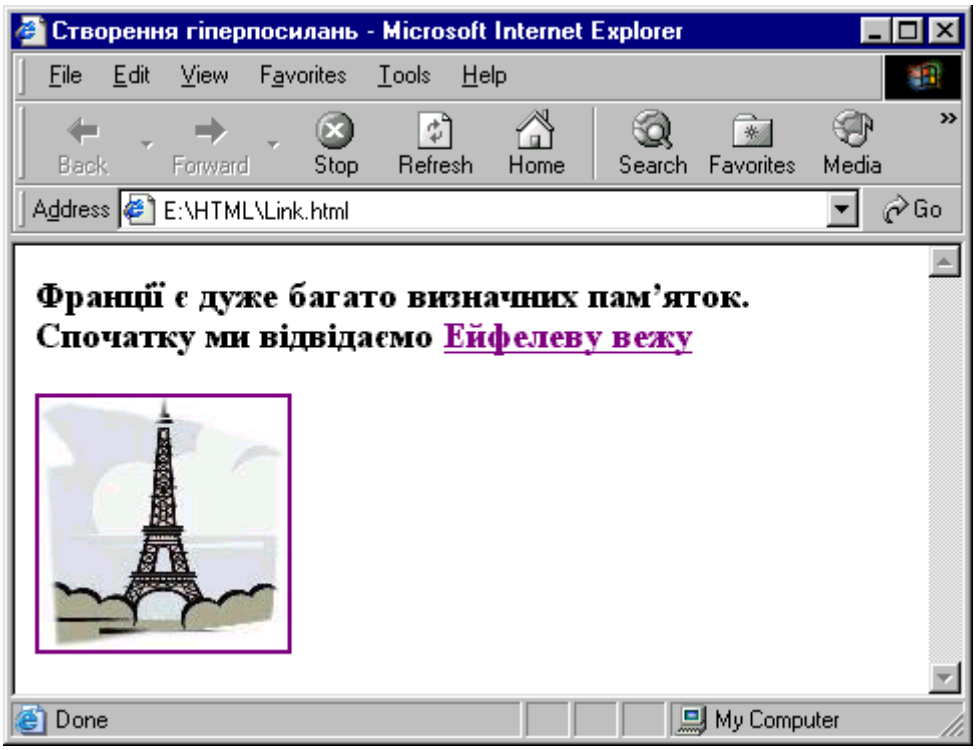


Рис. 4.3. Відображення графічного гіперпосилання у вікні браузера

Результат реалізації даного графічного гіперпосилання не повинен відрізнятись від результату тестового гіперпосилання.

6. Використання відносної адресації у гіперпосиланнях.

6.1. Створимо у папці HTML папку A. У папці A створимо текстовий документ з назвою Zamok.html.

6.2. У файлі Zamok.html визначимо такий HTML-код:

<html>
<head>
<title>Замок</title>
</head>
<body>
<img src=«Zamok.jpg» height=«300» align=«left»>

</body>
</html>

Переглянемо документ Zamok.html (рис. 4.4).

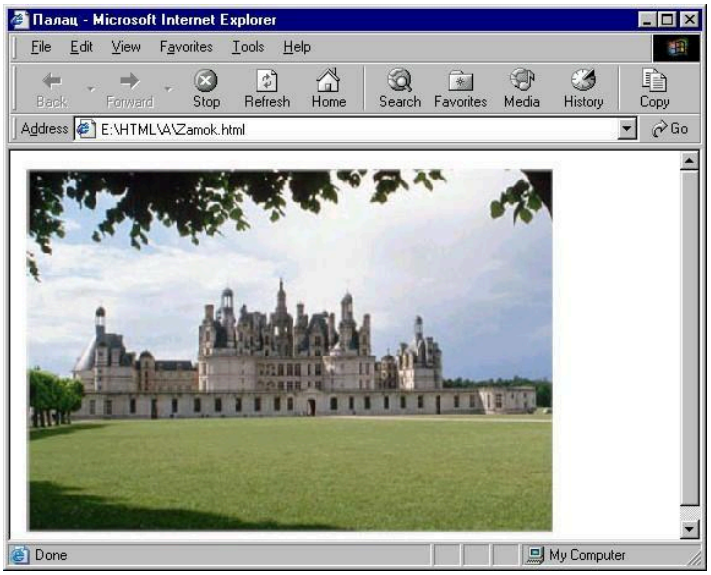


Рис. 4.4. HTML-документ Zamok.html

6.3. Визначимо у документі Link.html текстове та графічне гіперпосилання на документ Zamok.html:

```
...
<a href=«A/Zamok.html»><center>Замок</center></a>
<br><center><a href=«A/Zamok.html»><img src=«Flower.jpg»
width=«150» height=«150»></a></center>
</body>
```

Переглянемо оновлений документ Link.html (рис. 4.5).

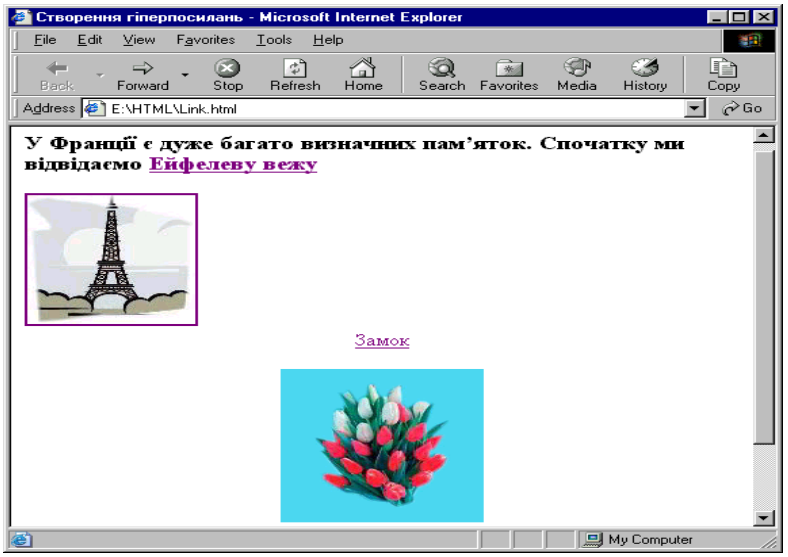


Рис. 4.5. Відображення оновленого документа Link.html

6.4. На одному рівні з папкою HTML створимо папку У. Наприклад, якщо папка HTML знаходиться у корені диску E, то і папку У створимо у корені диску E. Скопіюємо файл Zamok.html у папці B. У документі Link.html запишемо текстове гіперпосилання на документ Zamok.html, що знаходиться у папці У:

```
...
<br><a href=«../B/Zamok.html»>На документ Zamok.html,
що знаходиться у папці У</a>
</body>
```

Реалізація цього посилання буде відрізнятись від поданого на рис. 4 тільки тим, що в адресному рядку браузера буде записано інший шлях — *E:\B\Zamok.html*.

7. Використання абсолютної адресації у гіперпосиланнях.

7.1. Використовуючи абсолютну адресацію, додамо у документі Link.html текстове гіперпосилання на HTML-документ Zamok.html, що знаходиться у папці B, тобто за адресою E:\B\Zamok.html:

```
...
<br><a href=«=«E://B/Zamok.html»>На документ Zamok.html,
що знаходиться у папці У</a>
</body>
```

Реалізація цього посилання не повинна відрізнятися від реалізації попереднього посилання.

7.2. Визначимо у документі Link.html текстове та графічне гіперпосилання на сайт, що знаходиться за адресою *http//narod.yandex.ru*:

```
...

<br><a href=«http//narod.yandex.ru»>Замок</a>
<br><a href=«http//narod.yandex.ru»><img src=«Flower.jpg»></a>
</body>
```

Реалізація кожного з цих посилань показана на рис. 4.6.

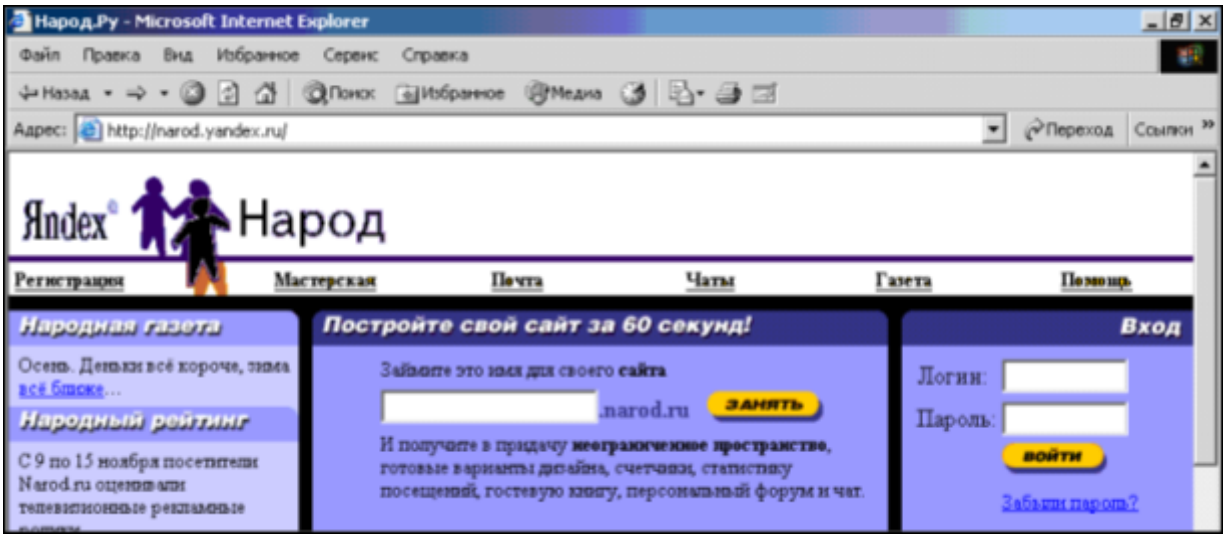


Рис. 4.6. Перехід по гіперпосиланню на сайт за адресою *http//narod.yandex.ru*

8. Використаннягіперпосилань у середині HTML-до- кумента.

8.1. У папці HTML створимо текстовий документ з назвою Recepti.html та запишемо у ньому такий HTML-код:

```
<html>
<head><title>Створення внутрішнього гіперпосилан-
ня</title></head>
<body>
<h1>Рецепти</h1>
<h3>1. Кошики з кавовим кремом </h3>
<h3>2. Тістечка «Наполеон»</h3>
<h3>3. Яблука у заварному тісті </h3>
<h2 align=«center»>1. Кошики з кавовим кремом </h2>
<h3>Тісто:</h3>
<p align=«justify»>300 г борошна, 100 г цукру, 150 г вершкового масла або маргарину.</p>
<h3>Крем:</h3>
<p align=«justify»>1 ст. ложка борошна, 2 жовтки, 100 г цукру, 100 г вершкового масла, 3 ст. ложки настою міцної кави, 100 г посічених горіхів.</p>
...
<h2 align=«center»>3. Яблука у заварному тісті </h2>
<h3>Тісто:</h3>
```

80 г розтопленого вершкового масла, 1 склянка води, 150 г борошна, 3 яєць, на кінчик ножа
<h3>Начинка:</h3>
1 кг яблук, 1 ст. ложка варення.
<h3>Приготування </h3>
До гарячого розтопленого масла долити гарячої води і закип’ятити...
</body>
</html>

8.2. Створюємо у документі Recepti.html посилання на текст третього пункту документ – «Яблука у заварному тісті». Для цього:

– розмістимо безпосередньо перед третім пунктом «якір» з ім’ям *apple*:

...
<h2 align=«center»>3. Яблука у заварному тісті </h2>
...

– модифікуємо HTML-код файла Recepti.html для визначен- ня безпосередньо внутрішнього посилання. При цьому у параме- трі href вказуємо ім’я “якоря” з префіксом #:

...
<h3><ahref=«#apple»>3. Яблука у заварному тісті</h3>
...

Відображення документа Recepti.html одразу після завантаження та після реалізації визначеного нами внутрішнього посилання показані на рис. 4.7 та 4.8.

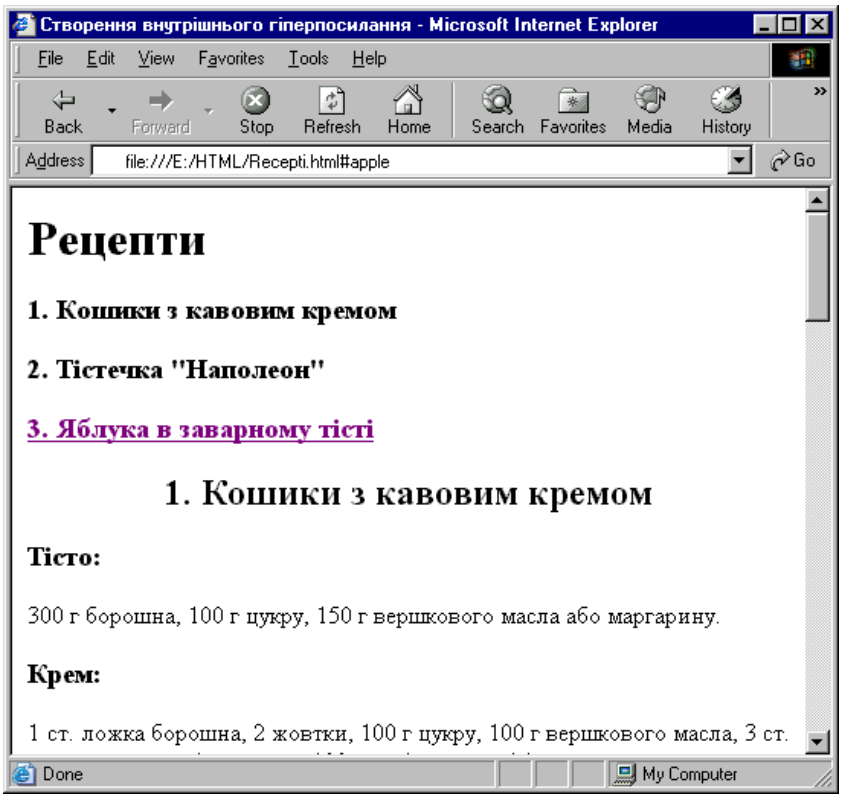


Рис. 4.7. Відображення документа Recepti.html відразу після завантаження

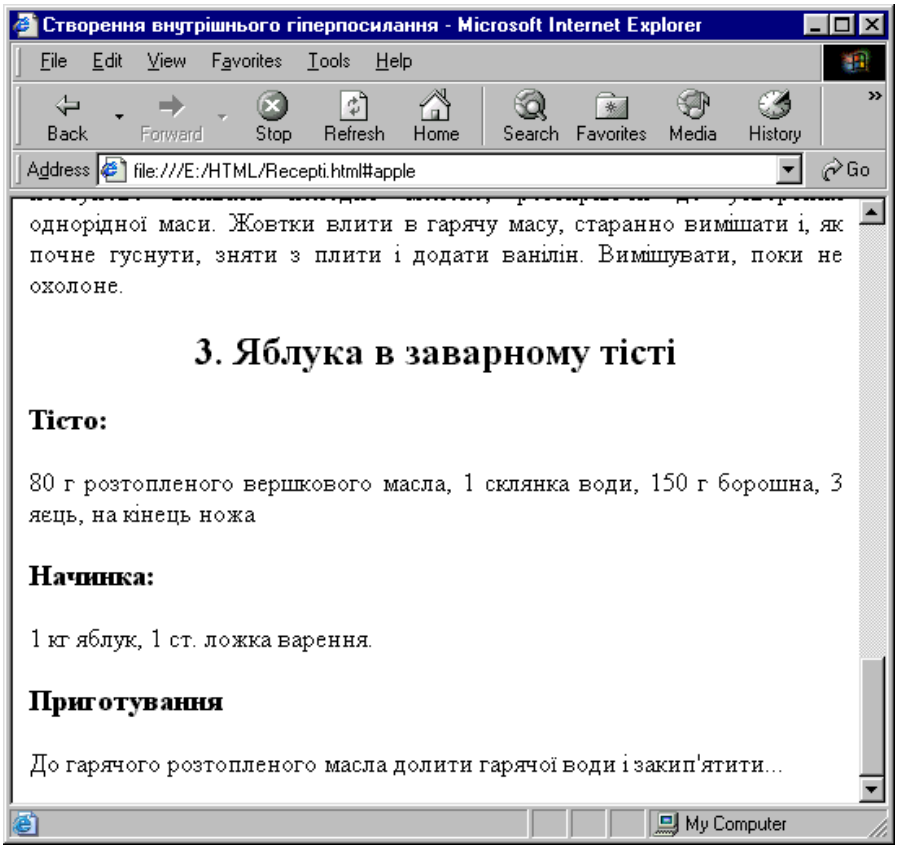


Рис. 4.8. Відображення документа Recepti.html після реалізації внутрішнього посилання

9. Визначимо у документі Link.html текстове гіперпосилан- ня на рисунок Flower.jpg, що знаходиться у папці HTML:

```
...
<br><a href=«Flower.jpg»>Квіти</a>
</body>
```

Реалізація даного посилання показана на рис. 4.9.

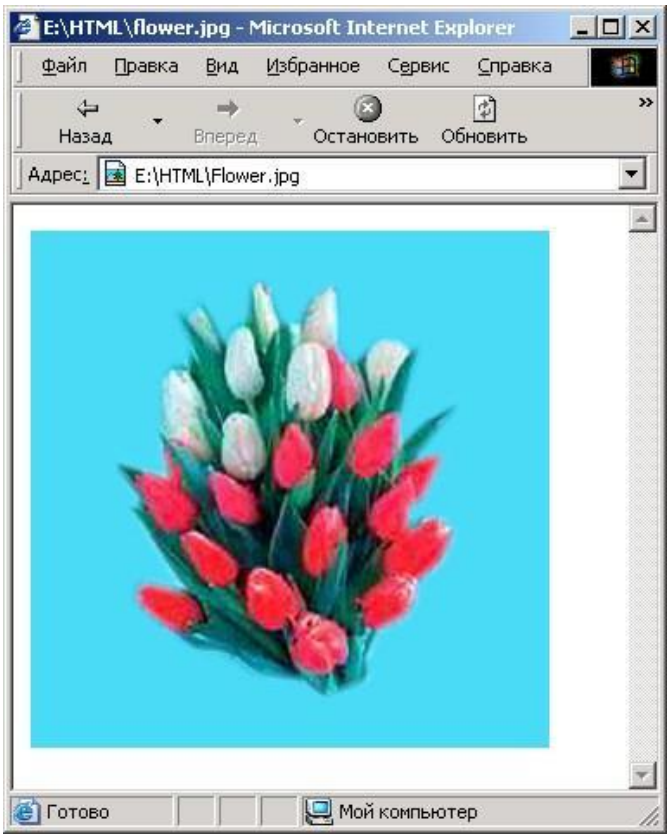


Рис. 4.9. Перехід по текстовому гіперпосиланню на рисунок Flower.jpg

10. Визначимо у документі Link.html текстове гіперпосилання на адресу електронної пошти:

```
...
<br><a href=«mailto:pochta@ukr.net «>Пошта</a>
</body>
```

Реалізація даного посилання у випадку використання як пош- товий клієнт програми «OutlookExpress» показана на рис. 4.10.

11. Створити три папки з назвами A,B,C. У папці A створити HTML-документ 1.html, у папці У − 2.html, а у папці C −

3.html. У кожному із цих документів визначити гіперпосилання на два інших документи як з абсолютною, так із відносною

адресацією. Крім того, документ 1.html повинен завжди відкриватись у новому вікні браузера.

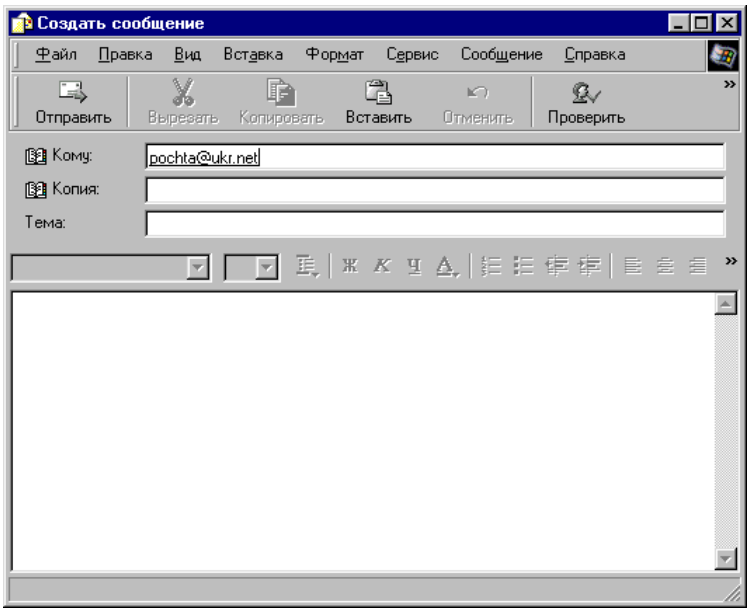


Рис. 4.10. Реалізація гіперпосилання на адресу електронної пошти pochta@ukr.net

12. Оформити звіт.

Запитання для самоперевірки

1. Як відкрити гіперпосилання у новому вікні браузера?
2. Як визначити якір у HTML-документі?
3. Як використовується у гіперпосиланнях абсолютна адре- сація?
4. Як використовується у гіперпосиланнях відносна адресація?
5. Як відкрити гіперпосилання у тому самому вікні браузера?
6. Правила запису текстових гіперпосилань.
7. Правила запису графічних гіперпосилань.
8. Як визначити гіперпосилання на рисунок?
9. Яквизначити гіперпосилання у середині HTML- документа?
10. Як записати гіперпосилання на адресу електронної пошти?

ЛАБОРАТОРНА РОБОТА 5

Тема. Створення списків

Мета: опанувати основні прийоми роботи зі створення списків.

Хід роботи

1. Скопіюємо у папці HTML виданий викладачем графічний файл star.jpg.

2. У папці HTML створити текстовий документ з назвою List.html, відкрити його за допомогою браузера та перейдемо до редагування HTML-коду.

3. Створимо маркірований список, елементами якого будуть місяці року. Для цього у файлі List.html записуємо HTML-код:

```
<html>
<head>
<title>Створення списків</title>
</head>
<body>
<ul>
<b>Місяці року:</b>
<li>Січень
<li>Лютий
<li>Березень
<li>Квітень
<li>Травень
<li>Червень
<li>Липень
<li>Серпень
<li>Вересень
<li>Жовтень
<li>Листопад
<li>Грудень
</ul>
</body>
```

</html>

Перегляд списку у браузері повинен відповідати рис. 5.1.

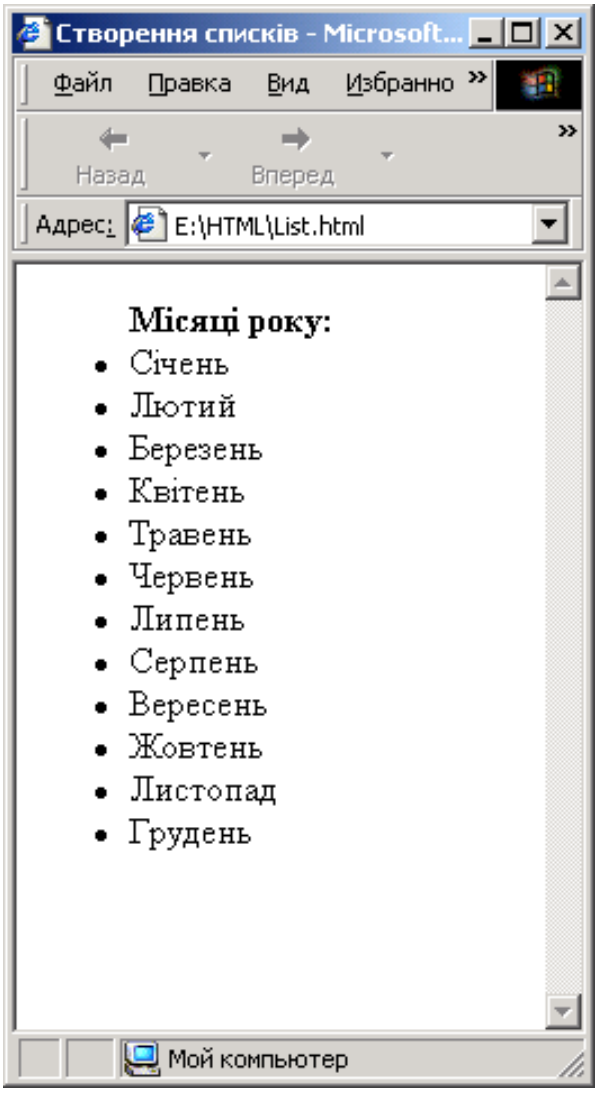


Рис. 5.1. Відображення браузером маркірованого списку

4. Розглянемо можливість зміни типу маркера. Необхідно відобразити маркери заповненими квадратиками для перших чотирьох елементів списку, відобразити маркери незаповненими колами для інших чотирьох елементів списку та відобразити маркери заповненими колами для останніх чотирьох елементів списку. Реалізуємо зміну типу маркерів за допомогою параметра *type* тега <*li*>. Для цього модифікуємо код списку:

```
…
<ul>
<b>Місяці року:</b>
<li type=«square»>Січень
<li type=«square»>Лютий
```

<li type=«square»>Березень
<li type=«square»>Квітень
<li type=«circle»>Травень
<li type=«circle»>Червень
<li type=«circle»>Липень
<li type=«circle»>Серпень
<li type=«disk»>Вересень
<li type=«disk»>Жовтень
<li type=«disk»>Листопад
<li type=«disk»>Грудень

</body>

Переглянемо цей маркірований список у браузері (рис. 5.2).

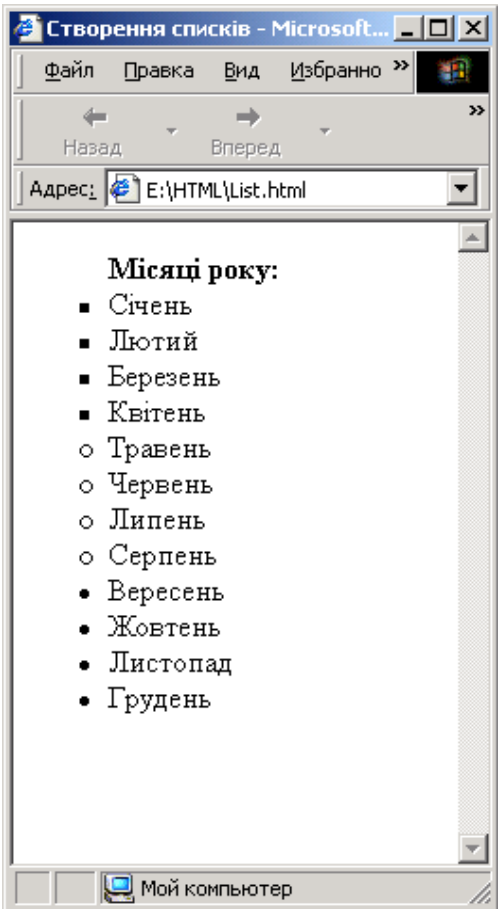


Рис. 5.2. Відображення маркірованого списку, для елементів якого визначено різні значення параметра type

5. Створимо список з графічними маркерами. При цьому можна обійтись без тегів <*li*>. Перед кожним елементом списку

необхідно вставити графічне зображення. Тоді елементи списку відокремлюємо один від одного за допомогою тегу примусового переводу рядка <*br*>. Графічний файл star.jpg повинен знаходиться у тому самому каталозі, що і наш файл List.jpg, тобто у папці HTML на диску E. Записуємо HTML-код:

```
…
<ul>
<b>Місяці року:</b><br><br>
<imgsrc=«star.jpg»>Січень<br>
<imgsrc=«star.jpg»>Лютий<br>
<imgsrc=«star.jpg»>Березень<br>
<img src=«star.jpg»>Квітень<br>
<img src=«star.jpg»>Травень<br>
<img src=«star.jpg»>Червень<br>
<img src=«star.jpg»>Липень<br>
<img src=«star.jpg»>Серпень<br>
<img src=«star.jpg»>Вересень<br>
<img src=«star.jpg»>Жовтень<br>
<img src=«star.jpg»>Листопад<br>
<img src=«star.jpg»>Грудень
</ul>
</body>
```

Переглянемо цей маркірований список у браузері (рис. 5.3).

6. Створимо нумерований список. Реалізуємо це за допомо- гою тегу – контейнера <*ol*></*ol*>. Елементами нумерованого списку будуть знову місяці року. Записуємо відповідний HTML- код та переглянемо цей нумерований список у браузері (рис. 5.4):

```
…
<ol>
<b>Перший квартал включає такі місяці року:</b>
<li>Січень
<li>Лютий
<li>Березень
</ol>
```

</body>

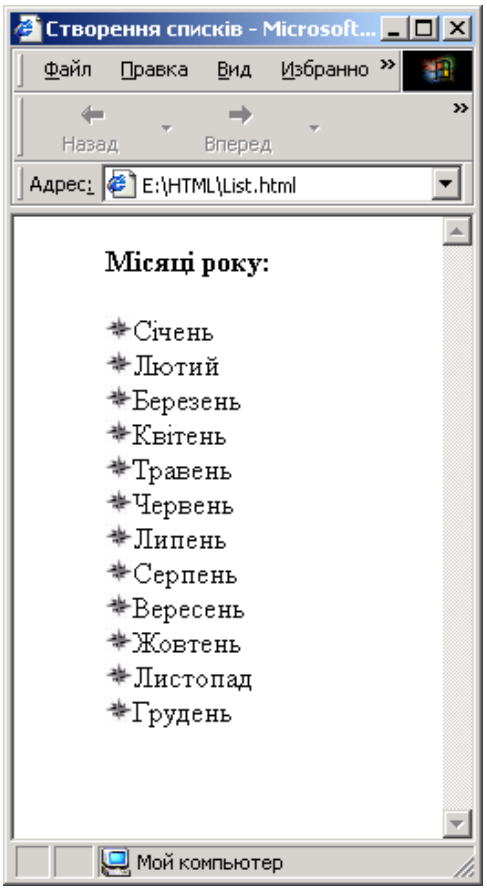


Рис. 5.3. Відображення маркірованого списку з графічними маркерами

7. Визначимо різні види нумерації списку. Реалізуємо це за допомогою параметра *type* тегу *<li>*. Записуємо HTML-код для вказаного списку та переглянемо його у браузері (рис. 5.5):

```
...
<ol>
<b>Квартали включають такі місяці року:</b>
<li type=«I»>Січень, лютий, березень
<li type=«i»>Квітень, травень, червень
<li type=«A»>Липень, серпень, вересень
<li type=«a»>Жовтень, листопад, грудень
</ol>
</body>
```

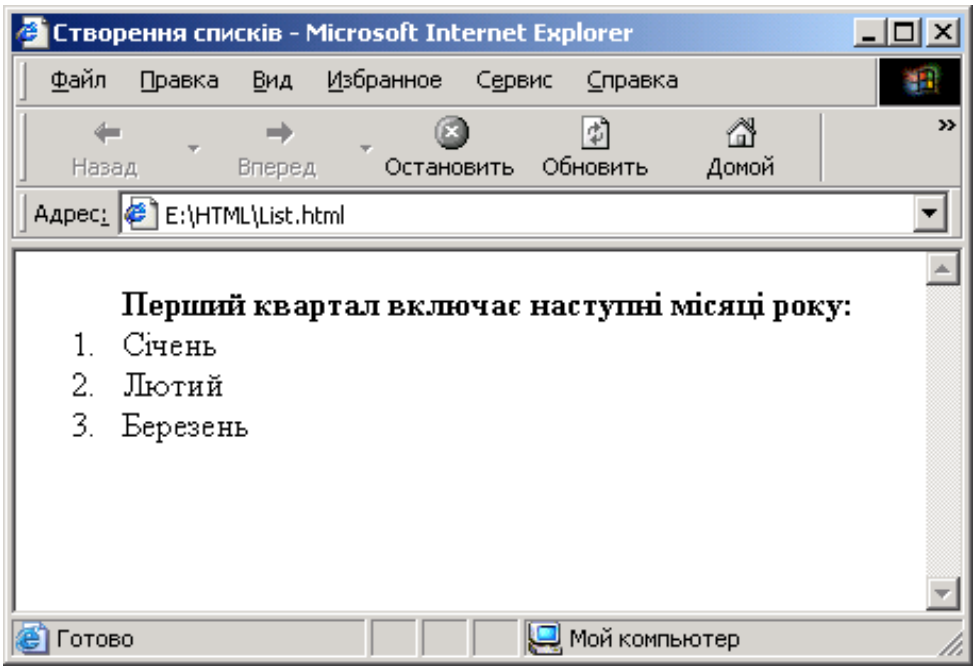


Рис. 5.4. Відображення нумерованого списку

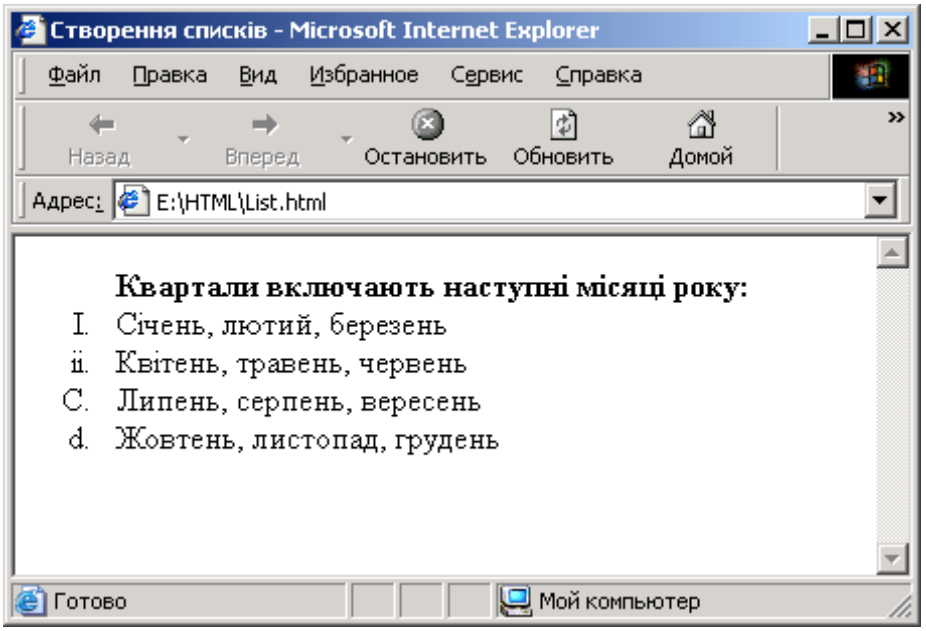


Рис. 5.5. Відображення нумерованого списку, для якого визначено різні види нумерації списку

8. Визначимо нумерований список, що складається з п’яти елементів. У цьому списку нумерація починається з 1995 та у третьому рядку списку початок нумерації змінюється на 2003. Записуємо відповідний HTML-код:

```
…
<ol start=«1995»>
<li>
```


<li value=«2003»>

</body>

Переглянемо цей нумерований список у браузері (рис. 5.6).

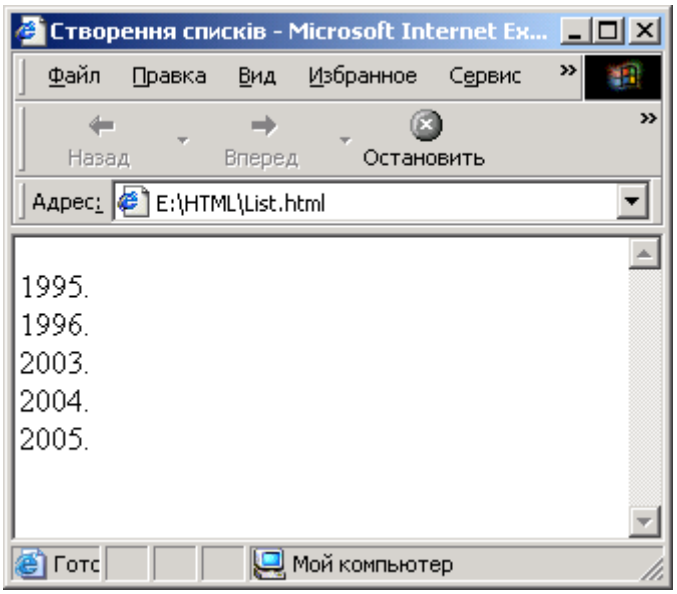


Рис. 5.6. Відображення нумерованого списку, для якого визначено заміну початку нумерації списку та заміну нумерації пунктів у середині списку

9. Створимо список визначень. У першій частині пункту списку запишемо термін, який визначає темперамент людини. Реалізуємо це за допомогою тегу <*dt*>. А у другій частині пункту списку запишемо пояснення, яке розкриває значення терміна. Це реалізуємо за допомогою тегу <*dd*>. Записуємо HTML-код для такого списку визначень:

…
<dl>
<h3 align=«center»> Класифікація типових темпераментів людини,
 заснована на поглядах Гіппократа </h3>
<dt> Флегматик
<dd> Пасивний, досить працездатний, повільно пристосовується;
 настрій стійкий, мало піддається зовнішньому впливу;

 млявість емоційних реакцій та повільність у вольовій діяльності

<dt> Сангвінік
<dd> Активний, енергійний, легко пристосовується;
 живість та рухливість емоційних реакцій, швидкість та сила вольових проявів

<dt> Холерик
<dd> Активний, досить енергійний, наполегливий;
 поривчастість та сила емоційних реакцій, бурхливі вольові прояви

<dt> Меланхолік
<dd> Пасивний, легко стомлюється, важко пристосовується;
 слабість вольових проявів та перевага подавленого настрою, невпевненість у собі
</dl>
</body>

Переглянемо цей список визначень у браузері (рис. 5.7).

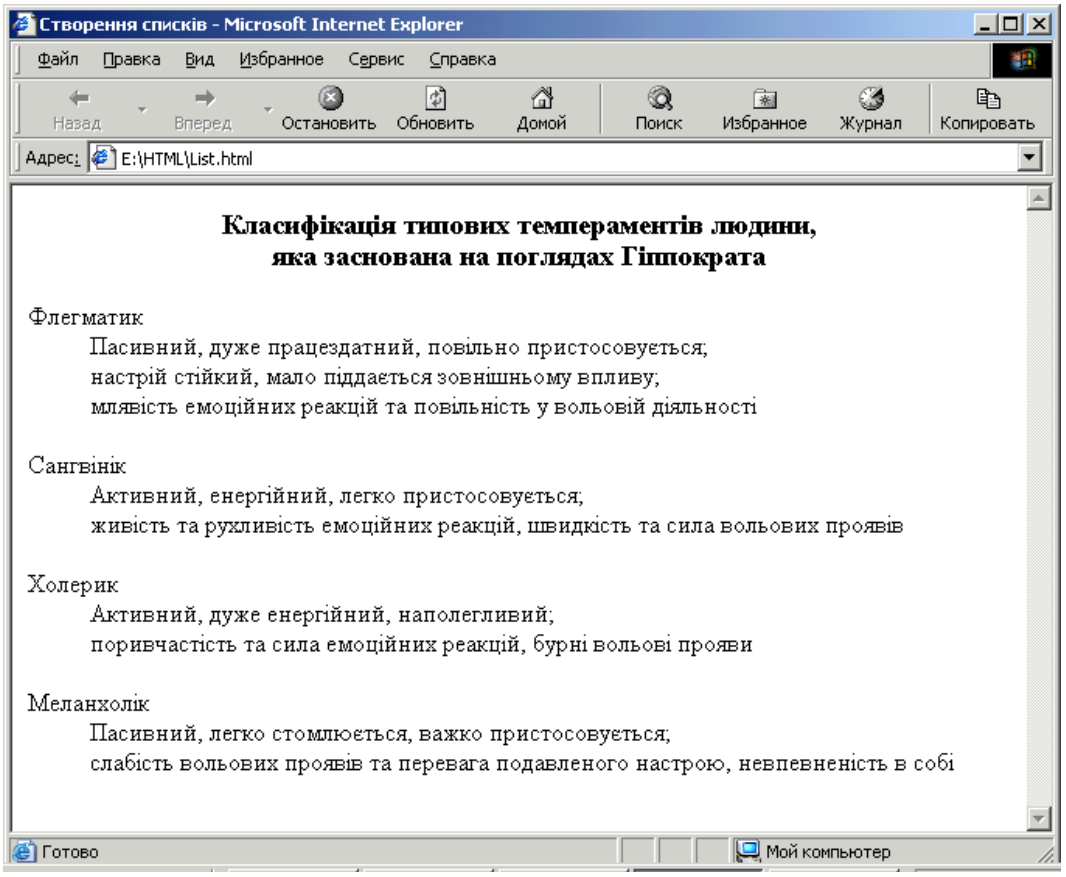


Рис. 5.7. Відображення списку визначень, який розкриває класифікацію темпераментів людини

10. Створимо вкладений список, показує супутники деяких планет. Розглянемо випадок, коли у кожен елемент маркірованого списку вкладений свій нумерований список. Записуємо відповідний HTML-код:

```
…
<ul>
<b>Супутники деяких планет</b>
<li>Земля
<ol>
<li>Луна
</ol>
<li>Марс
<ol>
<li>Фобос
<li>Деймос
</ol>
<li>Уран
<ol>
<li>Аріель
<li>Умбрі ель
<li>Тітанія
<li>Оберон
<li>Міранда
</ol>
<li>Нептун
<ol>
<li>Тритон
<li>Нереїда
</ol>
</ul>
</body>
```

Переглянемо цей вкладений список у браузері (рис. 5.8).

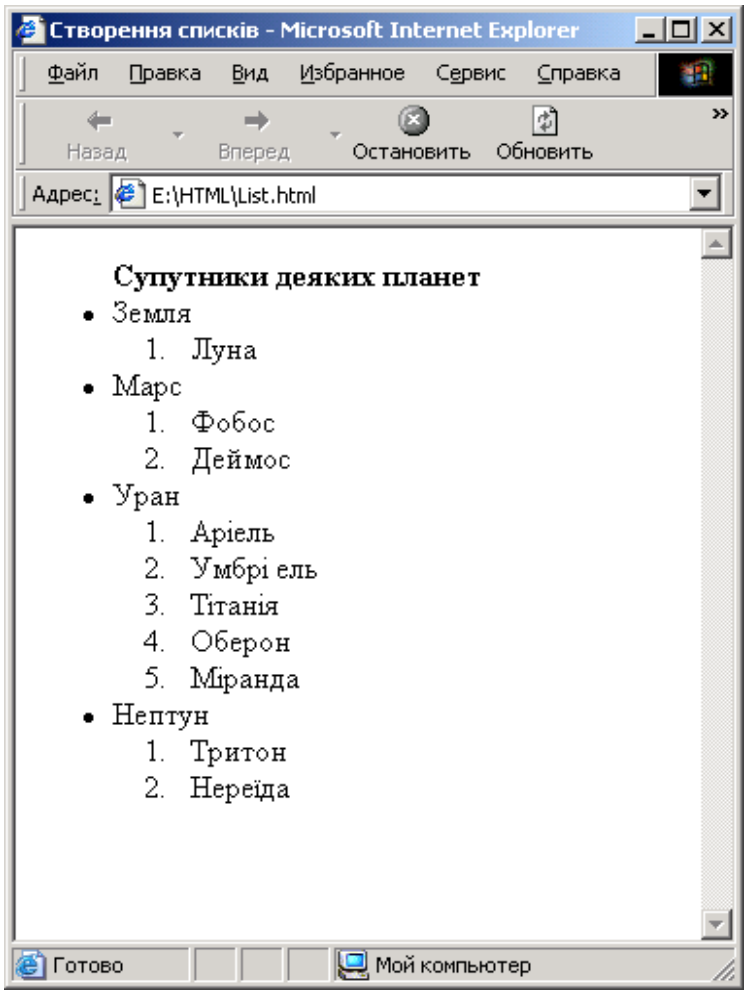


Рис. 5.8. Відображення вкладеного списку, показує супутники деяких планет

11. Визначити список відповідно до зразка, поданого на рис. 5.9.

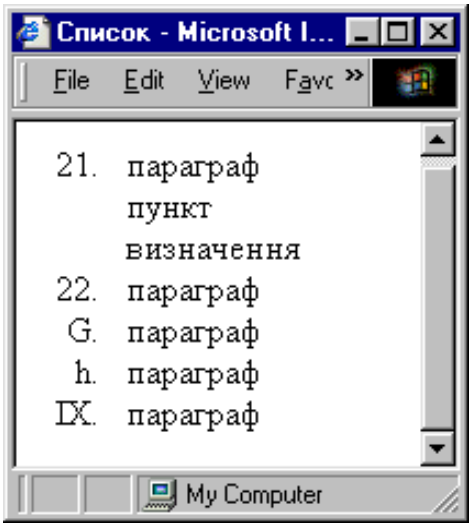


Рис. 5.9. Зразок списку

12. Оформити звіт.

Запитання для самоперевірки

1. Як визначається нумерований список?
2. Як визначається маркірований список?
3. Як визначається список визначень?
4. Як змінити тип маркера у маркірованому списку?
5. Як змінити порядок нумерації у нумерованому списку?
6. Як визначити список з маркерами у вигляді рисунків?
7. Як почати нумерований список з довільного номера?
8. Як визначаються вкладені списки?
9. Навіщо використовуються списки?
10. Як пронумерувати список за допомогою прописних англійських літер?

ЛАБОРАТОРНА РОБОТА 6

Тема. Створення таблиць
Мета: навчитись створювати таблиці.

Хід роботи

1. Скопіювати у папку HTML видані викладачем графічні файли Zamok.jpg та Tochka.jpg, Yellow.jpg.
2. У папці HTML створити текстовий документ з назвою Table.html, відкрити його за допомогою браузера та перейти до редагування HTML-коду.
3. Створити таблицю, яка складається із двох рядків та трьох стовпчик і містить текстову інформацію. Структура таблиці подана на рисунку 6.1.

11	12	13
21	22	23

Рис. 6.1. Таблиця, яку необхідно відобразити у вікні браузера

Записуємо відповідний HTML-код файлу Table.html з цієї таблиці (границі поки що невидимі) та переглянемо її у браузері (рис. 6.2):

```
<html>
<head>
<title>Створення таблиць </title>
</head>
<body>
<table>
<tr>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>21</td>
```

<td>22</td>
<td>23</td>
</tr>
</table>
</body>
</html>

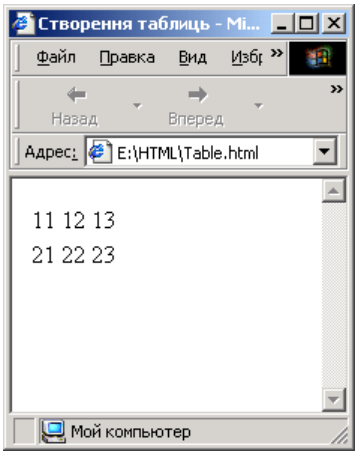


Рис. 6.2. Відображення таблиці, що складається з двох рядків та трьох стовпчиків у вікні браузера

4. Визначимо границі таблиці. Реалізуємо це за допомогою параметра border. Колір границі задається параметром bordercolor=«колір_границі». Для визначення таблиці із границями чорного кольору, товщина яких дорівнює 3 пікселям, необхідно записати такий код:

...
<table border=«3»bordercolor=«#000000» >
<tr>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
</table>
</body>

Переглянемо цю таблицю у браузері (рис. 6.3).

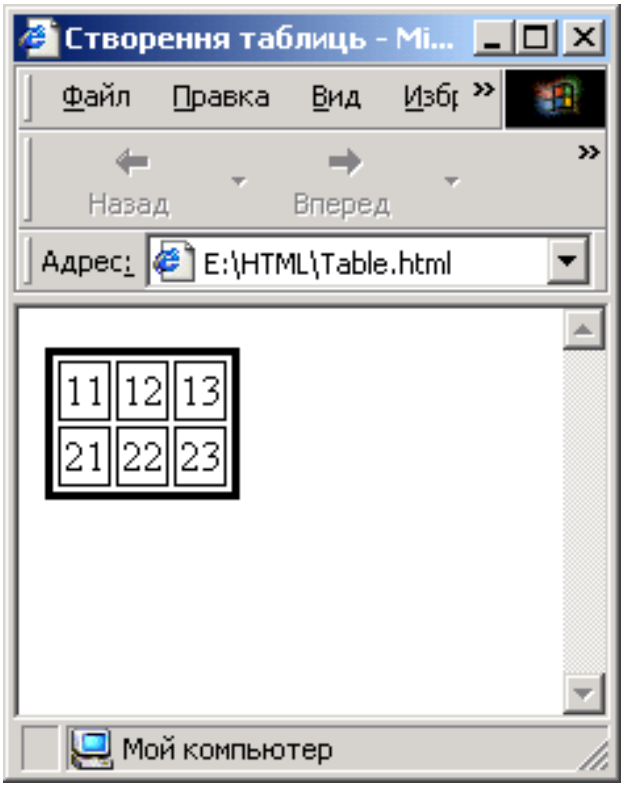


Рис. 6.3. Відображення таблиці з границями товщиною 3 пікселі у вікні браузера

5. Визначимо фон таблиці. Реалізуємо за допомогою пара- метра bgcolor =« колір_фону». Записуємо HTML-код для таблиці із сірим кольором фону та переглянемо її у браузері (рис. 6.4):

```
…
<tableborder=«3» bordercolor=«#000000» bgcolor=«#999999»>
<tr>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
</ table>
</body>
```

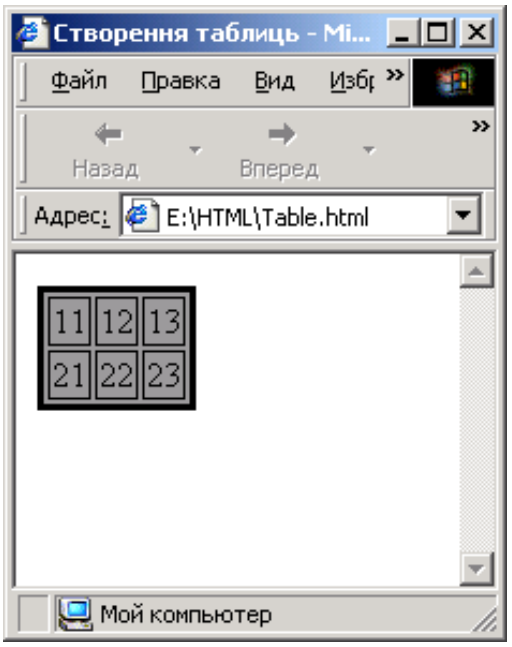


Рис. 6.4. Відображення таблиці із сірим кольором фону

6. Записуємо HTML-код для таблиці із білими та сірими ко- льорами фону для сусідніх комірок та переглянемо її у браузері (рис. 6.5):

```
…
<table border=«3» bordercolor=«#000000»>
<tr>
<td bgcolor=«#ffffff»>11</td>
<td bgcolor=«#999999»>12</td>
<td bgcolor=«#ffffff»>13</td>
</tr>
<tr>
<td bgcolor=«#999999»>21</td>
<td bgcolor=«#ffffff»>22</td>
<td bgcolor=«#999999»>23</td>
</tr>
</ table>
</body>
```

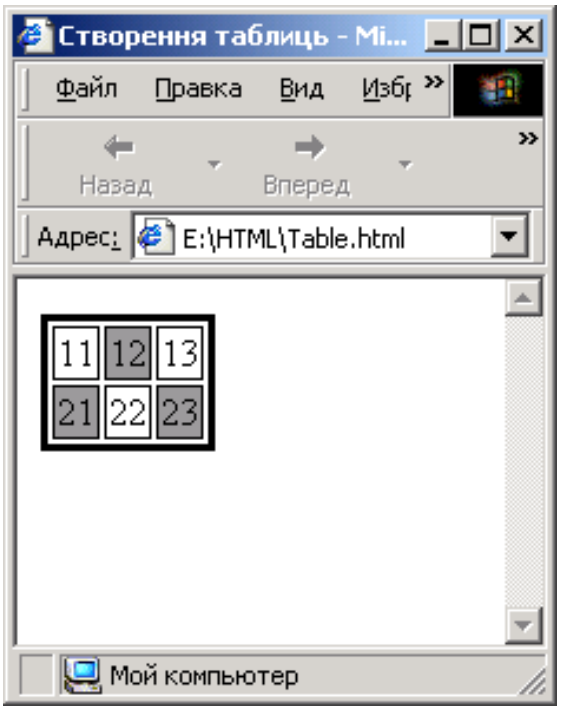


Рис. 6.5. Відображення таблиці з білими та сірими кольорами фону для сусідніх комірок

7. Визначимо висоту рядків та ширину стовпців. Створимо таблицю із рядками висотою 40 та стовбцями шириною стовбців 60 пікселів. Ширина стовпців та висота рядків задається за допо- могою параметрів width та height відповідно. Записуємо HTML- код для заданої таблиці та переглянемо її у браузері (рис. 6.6):

```
…
<table border=«3» bordercolor=«#000000»>
<tr>
<tdheight = «40» width = «60» bgcolor = «#ffffff»>11</td>
<tdwidth = «60» bgcolor = «#999999»>12</td>
<td width = «60» bgcolor = «#ffffff»>13</td>
</tr>
<tr>
<td height = «40» width = «60» bgcolor = «#999999»>21</td>
<td width = «60» bgcolor = «#ffffff»>22</td>
<td width = «60» bgcolor = «#999999»>23</td>
</tr>
</table>
</body>
```

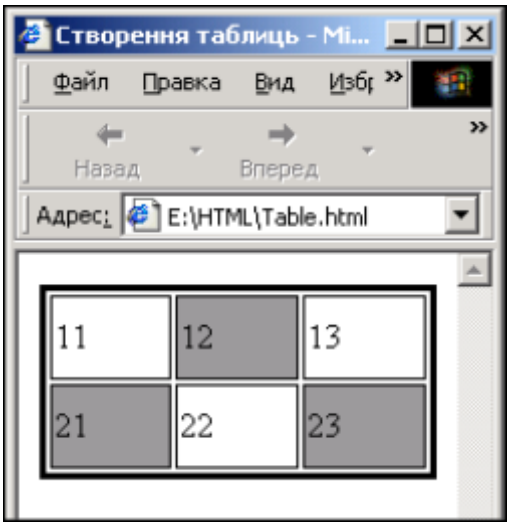


Рис. 6.6. Відображення таблиці з висотою рядків 40 та шириною стовпчиків 60 пікселів

8. Визначимо висоту та ширину для всієї таблиці, тоді всі комірки (стовпчики) і рядки поділять даний їм простір порівну, якщо не задавати особисто цього простору у відсотках чи пікселях. Записуємо HTML-код для такої таблиці та переглянемо її у браузері (рис. 6.7):

```
…
<table border=«3» bordercolor=«#000000» bgcolor =
«#999999» height = «200» width = «200»>
<tr>
<td>11</td><td>12</td><td>13</td>
</tr><tr>
<td>21</td><td>22</td><td>23</td>
</tr>
</table>
</body>
```

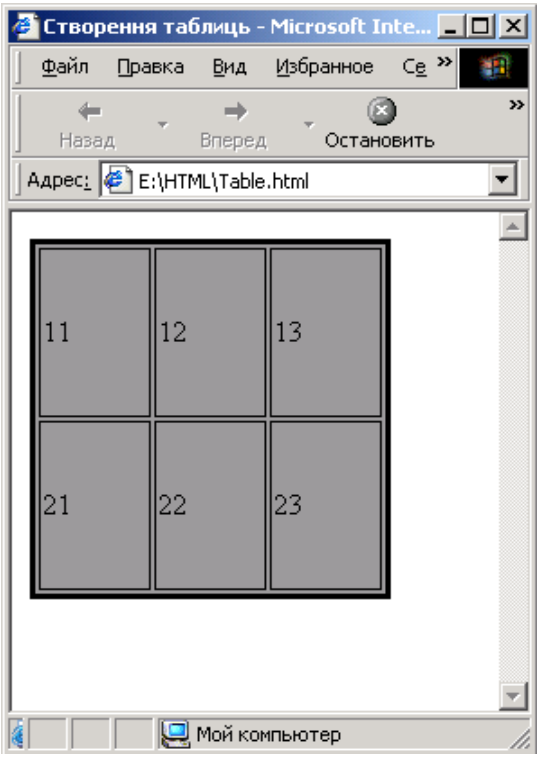


Рис. 6.7. Відображення таблиці з висотою 200 та шириною 200 пікселів

9. Для таблиці, поданої на рис. 6.7, визначимо висоту рядків і ширину стовпчиків у відсотках до висоти та ширини всієї таб- лиці. Записуємо необхідний HTML-код і переглянемо таблицю у браузері (рис. 6.8):

…
<table border=«3» bordercolor=«#000000» height = «200» width = «200» >
<tr>
<td height = «20%» width = «20%» bgcolor = «#ffffff»>11</td>
<td width = «30%» bgcolor = «#999999»>12</td>
<td width = «50%» bgcolor = «#ffffff»>13</td>
</tr><tr>
<td height=«80%» width= «20%» bgcolor = «#999999»>21 </td>
<td width = «30%» bgcolor = «#ffffff»>22</td>
<td width = «50%» bgcolor = «#999999»>23</td>
</tr>
</table>
</body>

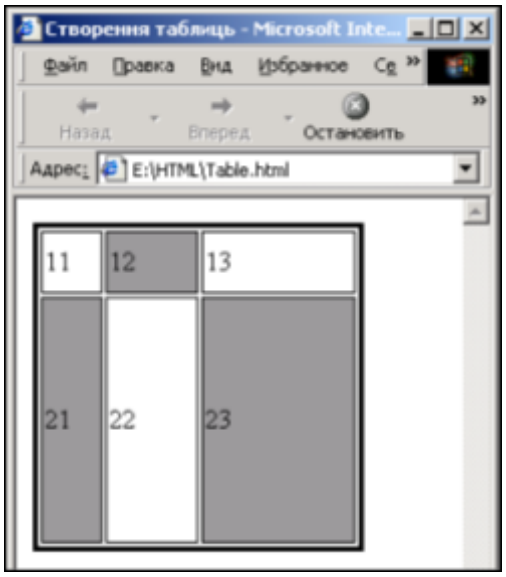


Рис. 6.8. Відображення таблиці з висотою і шириною стовпчиків, визначених у відсотках до висоти та ширини всієї таблиці

10. Зробимо вертикальне і горизонтальне вирівнювання зміс- ту таблиці. Таблиця, яку необхідно отримати, зображена на рис. 9.

Рис. 6.9. Таблиця, яку необхідно відобразити у вікні браузера

Реалізуємо це за допомогою параметра valign=«middle» (top, bottom) для вертикального вирівнювання і параметра align=«center» (left, right) для горизонтального вирівнювання. Записуємо HTML- код для заданої таблиці та переглянемо її у браузері (рис. 6.10):

…

<table border=«3» bordercolor=«#000000» >

<tr>

<td height=«40» width=«60» valign=«top» align=«left» bgcolor =

«#ffffff»>11</td>

```
<td width=«60» valign=«middle» align=«center» bgcolor =
«#999999»>12</td>
<td width=«60» valign=«bottom» align=«right» bgcolor =
«#ffffff»>13</td>
</tr>
<tr>
<td height=«40» width=«60» valign=«top» align=«left» bgcolor
=«#999999»>21</td>
<td width=«60» valign=«middle» align=«center» bgcolor =
«#ffffff»>22</td>
<td width=«60» valign=«bottom» align=«right» bgcolor =
«#999999»>23</td>
</tr>
</table>
</body>
```

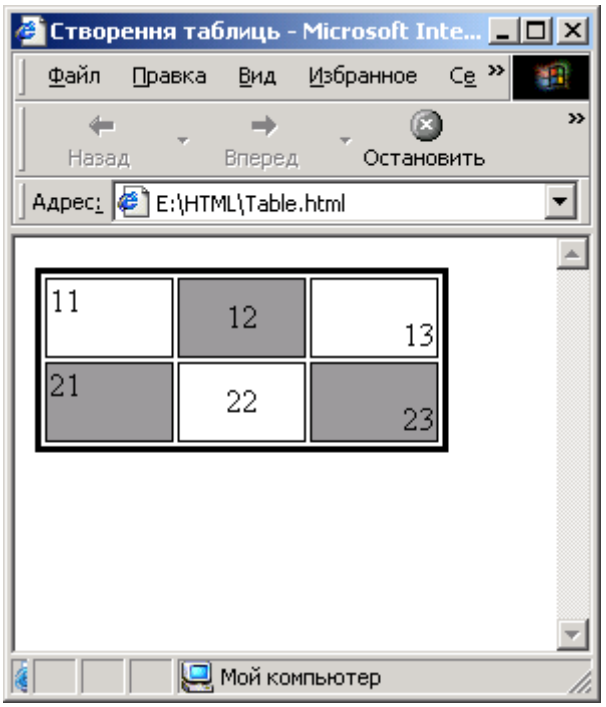


Рис. 6.10. Відображення таблиці з виконаним вирівнюванням змісту її комірок у вікні браузера

11. Об’єднаємо у першому рядку таблиці (рис. 6.10) перші дві комірки та задамо для них сірий колір фону. Для цього використаємо параметр colspan, який визначає кількість стовпчиків в об’єднаній комірці. Записуємо необхідний HTML-код:

…

```
<table border=«3» bordercolor=«#000000» >
<tr>
<td height=«40» colspan=«2» valign=«top» align=«left» bgcolor =
«#999999»>11</td>
<td width=«60» valign=«bottom» align=«right» bgcolor =
«#ffffff»>13</td>
</tr>
<tr>
<td height=«40» width=«60» valign=«top» align=«left» bgcolor
=«#999999»>21</td>
<td width=«60» valign=«middle» align=«center» bgcolor =
«#ffffff»>22</td>
<td width=«60» valign=«bottom» align=«right» bgcolor =
«#999999»>23</td>
</tr>
</table>
</body>
```

Відповідна таблиця подана на рис. 6.11.

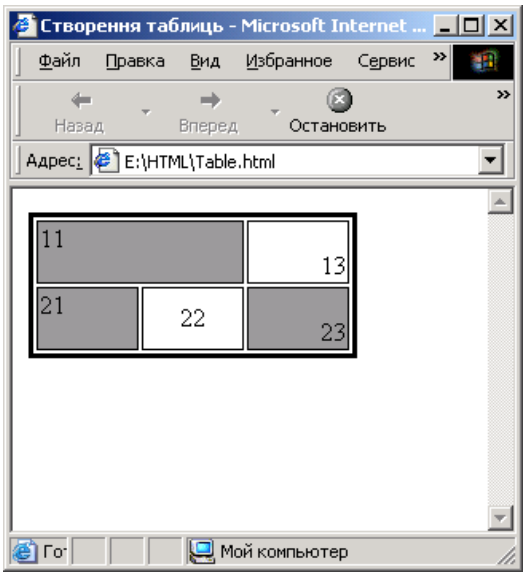


Рис. 6.11. Відображення таблиці з об'єднаними у першому рядку першими двома комірками, для яких заданий сірий колір фону

12. Використання параметрів cellspacing та cellspadding.

12.1. Встановимо відстань між комірками таблиці, поданої

на рис. 11, яка дорівнює 0 пікселів (це білий простір між комір-

ками таблиці). Реалізуємо це за допомогою параметра cellspacing. Записуємо необхідний HTML-код:

```
…
<table border=«3» bordercolor=«#000000» cellspacing=«0» >
<tr>
<td height=«40» colspan=«2» valign=«top» align=«left» bgcolor =
«#999999»>11</td>
<td width=«60» valign=«bottom» align=«right» bgcolor =
«#ffffff»>13</td>
</tr>
<tr>
<td height=«40» width=«60» valign=«top» align=«left» bgcolor
=«#999999»>21</td>
<td width=«60» valign=«middle» align=«center» bgcolor =
«#ffffff»>22</td>
<td width=«60» valign=«bottom» align=«right» bgcolor =
«#999999»>23</td>
</tr>
</table>
</body>
```

Відповідна таблиця показана на рис. 6.12.

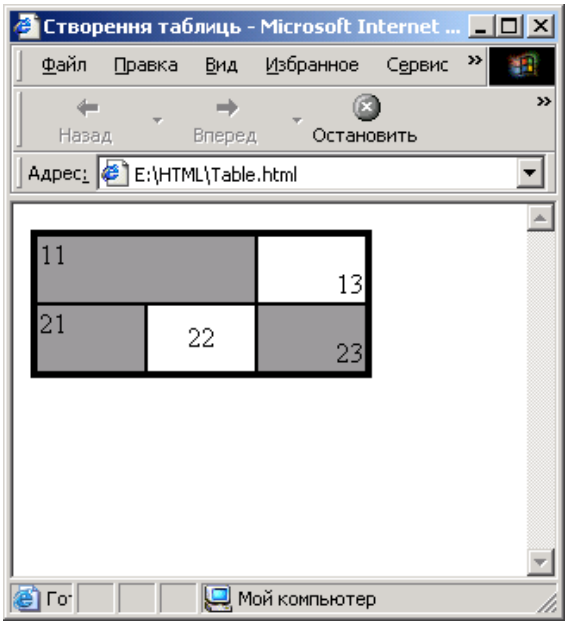


Рис. 6.12. Відображення таблиці, у якій відстань між комірками дорівнює 0 пікселів

12.2. Змінимо значення параметра cellspacing з 0 на 7. Нова таблиця показана на рис. 6.13.

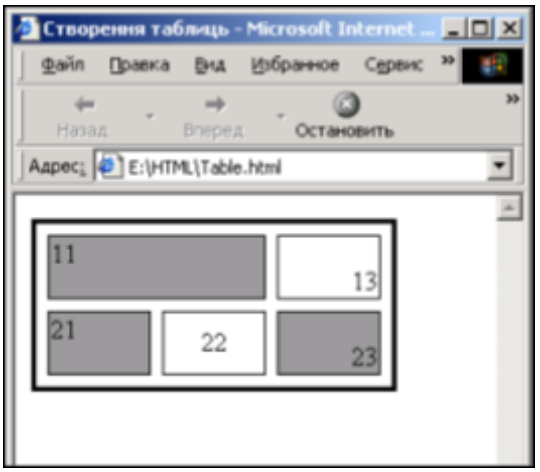


Рис. 6.13. Відображення таблиці, відстань між комірками дорівнює 7 пікселів

12.3. Розглянемо наслідки використання параметра cellpadding, що визначає відстань між границями комірок і даними, що знахо- дяться у цих комірках. Для наочності результатів спочатку за до- помогою параметра valing вирівняємо зміст комірок верхнього ряду доверху, а нижнього – донизу:

```
...
<table border=«3» bordercolor=«#000000»cellpadding=«0» >
<tr>
<td height=«40» colspan=«2» valign=«top» align=«left» bgcolor =
«#999999»>11</td>
<td width=«60» valign=«top» align=«right» bgcolor = «#ffffff»>13</td>
</tr>
<tr>
<td height=«40» width=«60» valign=«bottom» align=«left» bgcolor
=«#999999»>21</td>
<td width=«60» valign=«bottom» align=«center» bgcolor =
«#ffffff»>22</td>
<td width=«60» valign=«bottom» align=«right» bgcolor =
«#999999»>23</td>
</tr>
</table>
</body>
```

Відповідна таблиця показана на рис. 6.14.

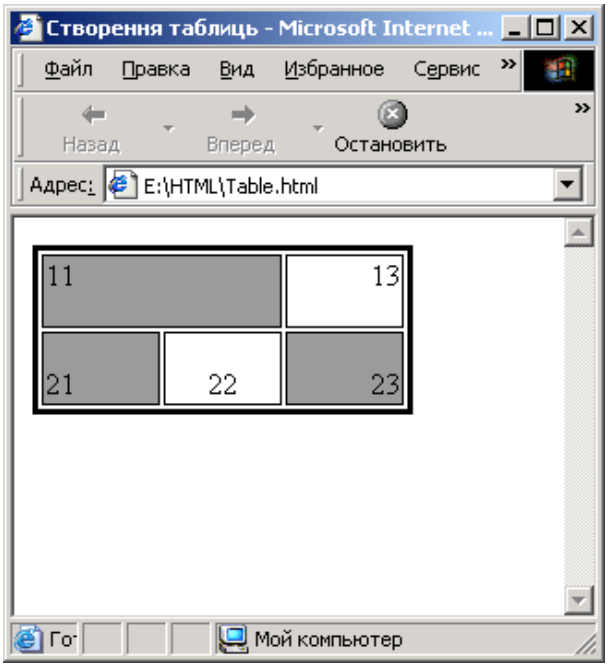


Рис. 6.14. Відображення таблиці, для якої параметр cellpadding дорівнює 0 пікселів

Змінимо значення параметра cellpadding з 0 на 7. Нова таблиця подана на рис. 6.15.

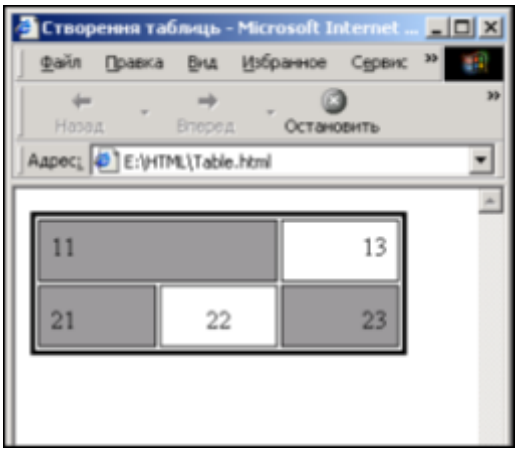


Рис. 6.15. Відображення таблиці, для якої параметр cellpadding дорівнює 7 пікселів

13. Визначимо фоновий рисунок таблиці. Реалізуємо це за до- помогою параметра background. HTML-код при цьому буде таким:

…

<table height = «200» width = «200» background=«Zamok.jpg»>

<tr>

<td></td><td>Замок</td><td></td>
</tr>
<tr>
<td></td><td></td><td></td>
</tr>
</table>
</body>

Рис. 6.16. Відображення таблиці, для якої визначено параметр background

14. Вкладені таблиці. Створимо таблицю, яка складається з одного ряду та трьох комірок. У третю комірку додамо ще одну таблицю, яка складається з чотирьох рядків та двох стовпчиків. HTML-код для такої таблиці такий:

…
<table align=«center»>
<tr height=«200»>
<tdwidth=«200» valign=«top»
background=«Zamok.jpg»
align=«center»><h1>

ЗВІТ</h1></td>
<td width=«50» background=«Tochka.jpg»></td>
<td width=«300» valing=«top» background=«Yellow.jpg»
align=«center»>

```
<h3>Таблиця, яка показує успішність групи ОА-Д-2:</h3>
<table cellspasing=«3» border=«3»
bordercoler=«black» background=«Yellow.jpg»>
<tr height=«40»>
<td width=«200» align=«center»> п’ятірок </td>
<td width=«100» align=«center»>10%</td>
</tr>
<tr height=«40»>
<td width=«200» align=«center» >четвірок </td>
<td width=«100» align=«center» >70% </td>
</tr>
<tr height=«40»>
<td width=«200» align=«center»>трійок </td>
<td width=«100» align=«center» >20% </td>
</tr>
<tr height=«40»>
<td width=«200» align=«center»>двійок </td>
<td width=«100» align=«center» >немає </td>
</tr>
</table>
<br><h3>Викладач</h3>
</td>
</tr>
</table>
</body>
```

Переглянемо цю таблицю у браузері (рис. 6.17).

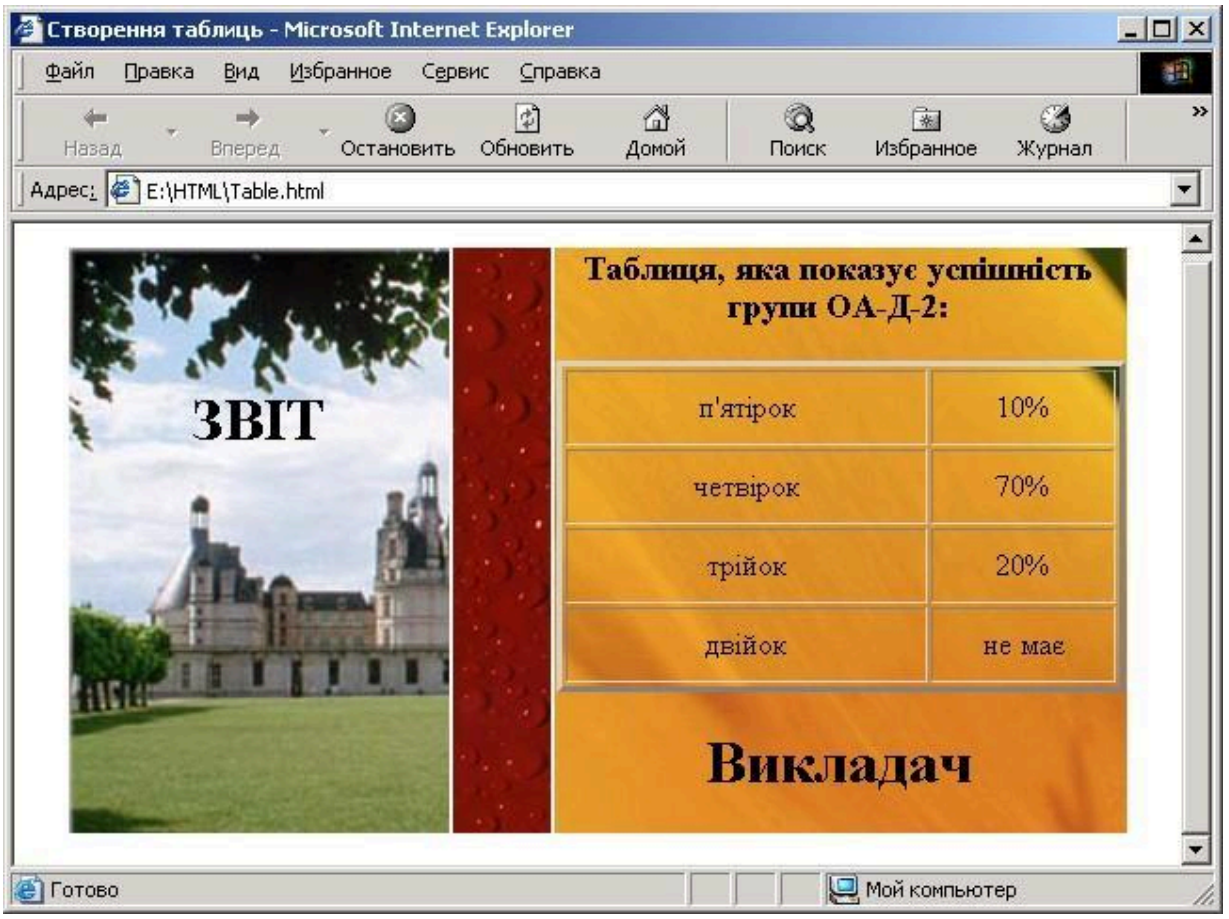


Рис. 6.17. Відображення таблиці, яка складається з одного ряду та трьох комірок і містить вкладену таблицю у третій комірці

14. У HTML-документі визначити таблицю, зміст та форматування якої відповідають таблиці, поданої на рис. 6.18.

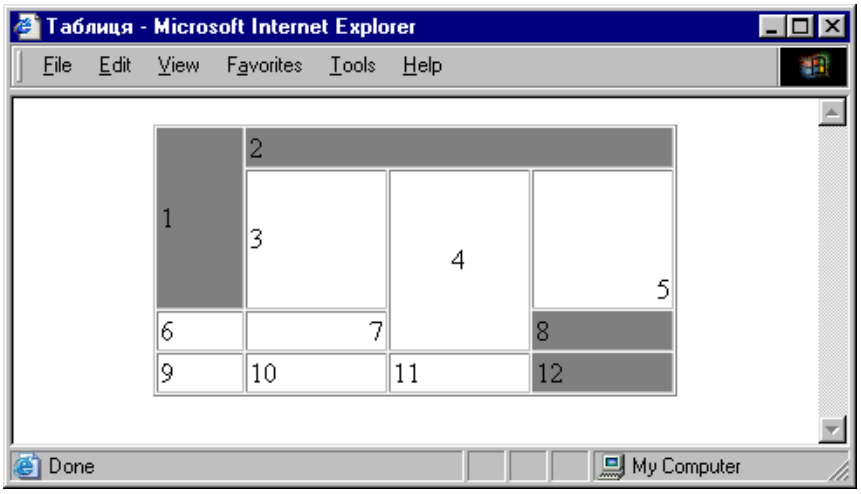


Рис. 6.18. Завдання для визначення таблиці

15. Оформити звіт.

Запитання для самоперевірки

1. Навіщо використовують таблиці?
2. Як визначити кількість рядків у таблиці?
3. Як визначити кількість стовпчиків у таблиці?
4. Як визначити горизонтальне вирівнювання змісту комірок таблиці?
5. Чи можна визначити вертикальне вирівнювання змісту рядків таблиці?
6. Чи можна визначити ширину окремої комірки таблиці?
7. Як визначити ширину таблиці?
8. Чи можна визначити висоту таблиці?
9. Як встановити відстань між комірками таблиці?
10. Як встановити товщину границь таблиці?

ЛАБОРАТОРНА РОБОТА 7

Тема. Створення фреймів
Мета: навчитися створювати фрейми.

Хід роботи

1. Скопіювати у папку HTML виданий викладачем графічний файл Eifel.jpg.
2. У папці HTML створити текстовий документ з назвою Frame.html.
3. У файлі Frame.html визначимо чотири фрейми зі структу- рою, показаною на рис. 7.1.

Ім’я першого фрейма logo, він займає всю ширину сторінки, у який завантажений файл logo.html. Далі ідуть два центральні фрейми: menu, у який завантажений файл menu.html, та content, у який завантажений файл content.html. Фрейм menu займає 25 % у ширину, а фрейм content – весь простір, що залишився.

<table>
<tr><td colspan="2">logo</td></tr>
<tr><td>menu</td><td>content</td></tr>
<tr><td colspan="2">bottom</td></tr>
</table>

Рис. 7.1. Розміщення фреймів у вікні браузера

Останній фрейм bottom займає нижню частину екрана, у який завантажений файл bottom.html. У файл Frame.html запишемо HTML-код для визначення фреймової структури:

```
<html>
<head>
<title>Створення фреймів</title>
</head>
<frameset rows=«25%,50%,25%»>
<frame src=«logo.html»>
<frameset cols=«25%,*»>
```

```
<frame src=«menu.html»>
<frame src=«content.html»>
</frameset>
<frame src=«bottom.html»>
</frameset>
</html>
```

4. У папці HTML створимо текстові документи з назвами logo.html, menu.html, content.html, bottom.html та запишемо у них такий HTML-код.

4.1. HTML-код документа logo.html:

```
<html>
<head>
<title>logo</title>
</head>
<body >
<img src=«Eifel.jpg» height=«110» align=«left»>
<h2 align=«center»>ТУРИСТИЧНА ФІРМА «АРС»</h2>
</body>
</html>
```

4.2. HTML-код документа menu.html:

```
<html>
<head>
<title>menu</title>
</head>
<body>
Новини<br><br
> Ціни<br><br>
Погода
</body>
</html>
```

4.3. HTML-код документа content.html:

```
<html>
<head>
<title>content</title>
```

```
</head>
<body>                                          <center>Зміст
                                                документа</center>

</body>
</html>
```

4.4. HTML-код документа bottom.html:

```
<html>
<head>
<title>bottom</title>
</head>
<body>
        <center>Нижній колонтитул документа</center>
</body>
</html>
```

Відображення фреймів (файла Frame.html) у браузері повинно відповідати рис. 7.2.

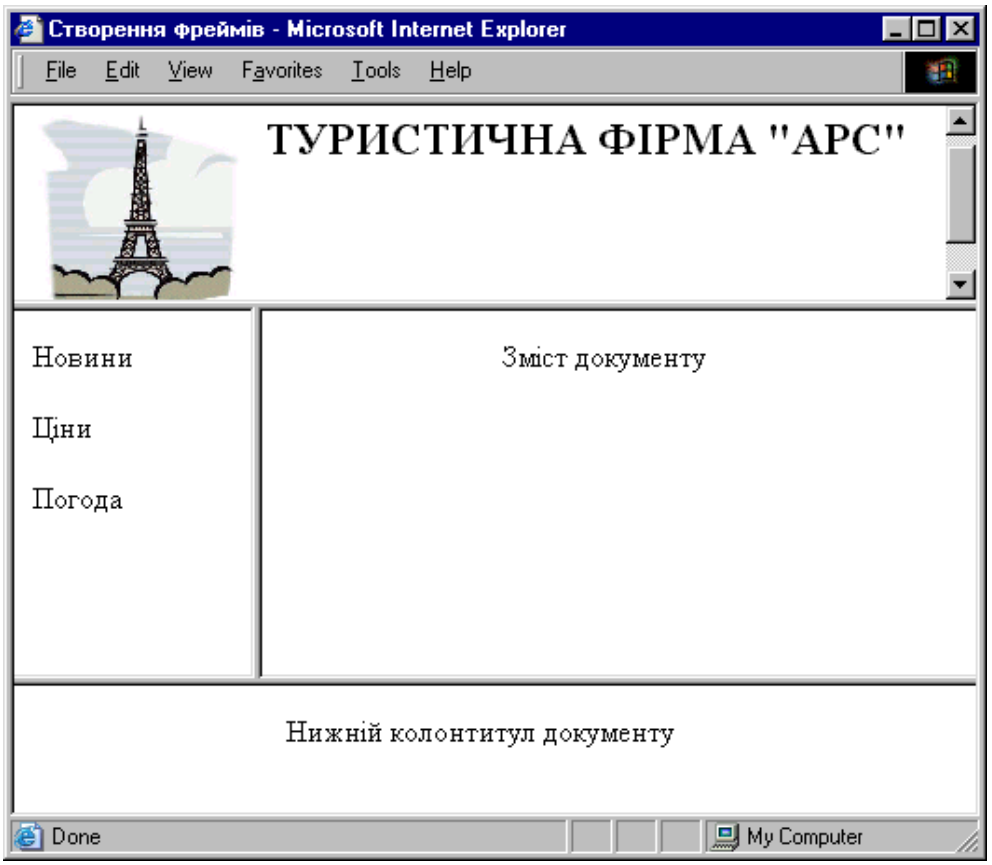


Рис. 7.2. ВідображенняHTML-документа з чотирма фрей-

мами у вікні браузера

5. Зробимо так, щоб смуга прокрутки у фреймі logo була ві- дсутня. Реалізуємо це за допомогою параметра *scrolling* тегу <*frame*>, значення якого у даному випадку буде дорівнювати «*no*». Корегуємо HTML-код файла Frame.html:

```
<frameset rows=«25%,50%,25%»>
<frame src=«logo.html»scrolling=«no»>
<frameset cols=«25%,*»>
<frame src=«menu.html»>
<frame src=«content.html» >
</frameset>
<frame src=«bottom.html»>
</frameset>
```

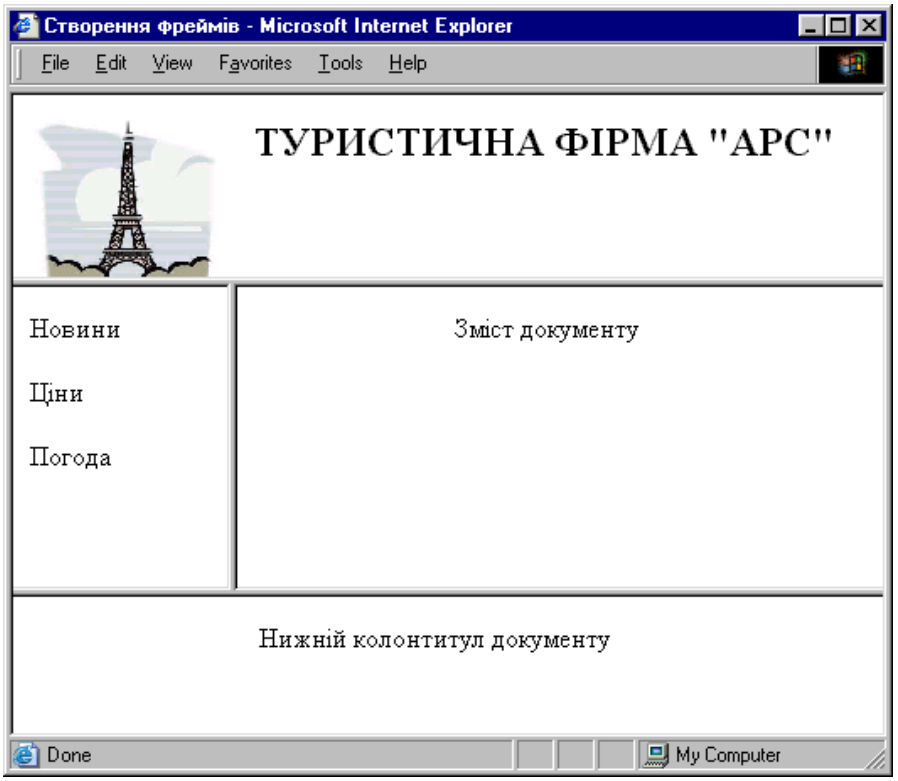


Рис. 7.3. Відображення у браузері чотирьох фреймів, у яких відсутні смуги прокрутки

6. Зробимо так, щоб межі між фреймами були відсутні. Реалі- зуємо це за допомогою параметра *frameborder* тегу <*frame*>, зна- чення якого у цьому випадку дорівнює *0*. Корегуємо HTML-код:

```
<frameset rows=«25%,50%,25%»>
<frame src=«logo.html» scrolling=«no» frameborder=0>
<frameset cols=«25%,*»>
```

<frame src=«menu.html» frameborder=0>
<frame src=«content.html» frameborder=0>
</frameset>
<frame src=«bottom.html» frameborder=0>
</frameset>

Переглянемо цей документ у вікні браузера (рис. 7.4).

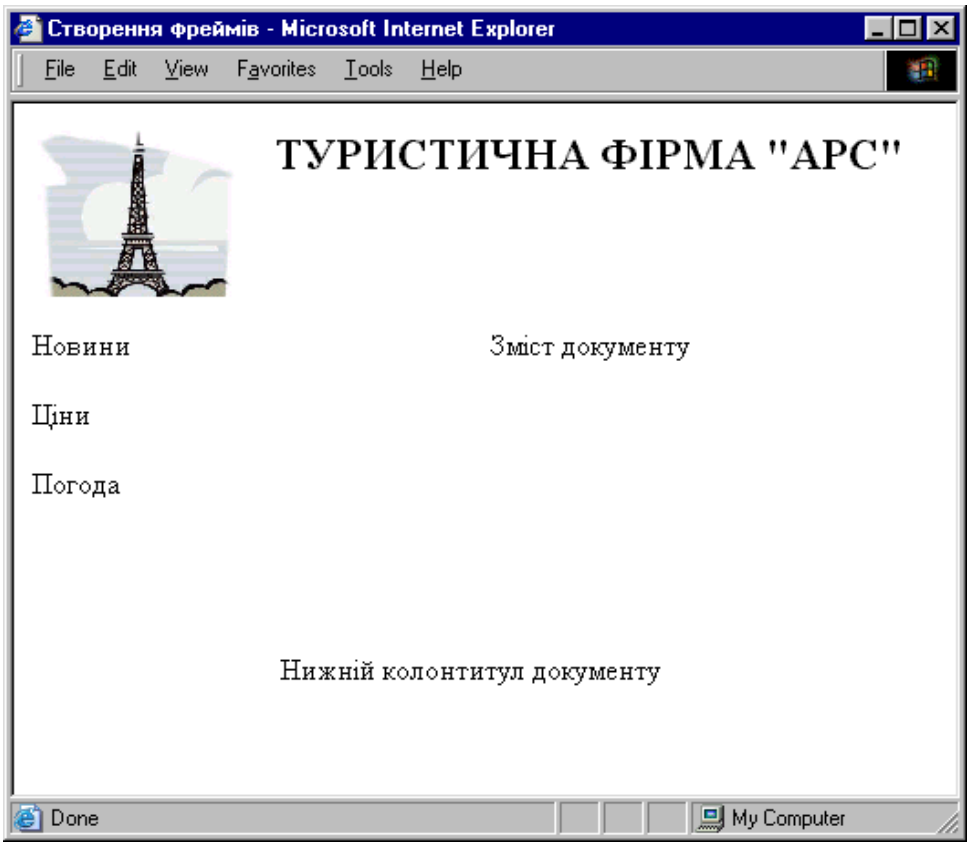


Рис. 7.4. Відображення у браузері чотирьох фреймів, для яких визначена відсутність межі між фреймами

7. Змінимо значення параметра *frameborder* з *0* на *1*, щоб визначити наявність межі між фреймами. А також визначимо зеле- ний колір межі між фреймами – це робимо за допомогою параметра *bordercolor* тегу *<frame>*. Записуємо відповідний HTML-код:

<frameset rows=«25 %,50 %,25 %»>
<frame src=«logo.html» scrolling=«no»
frameborder=1 bordercolor=«green»>
<frameset cols=«25 %,*»>
<frame src=«menu.html» frameborder=1 bordercolor=«green»>
<frame src=«content.html» frameborder=1 bordercolor=«green»>

</frameset>
<frame src=«bottom.html» frameborder=1 bordercolor=«green»>>
</frameset>

Переглянемо документ Frame.html у браузері (рис. 7.5).

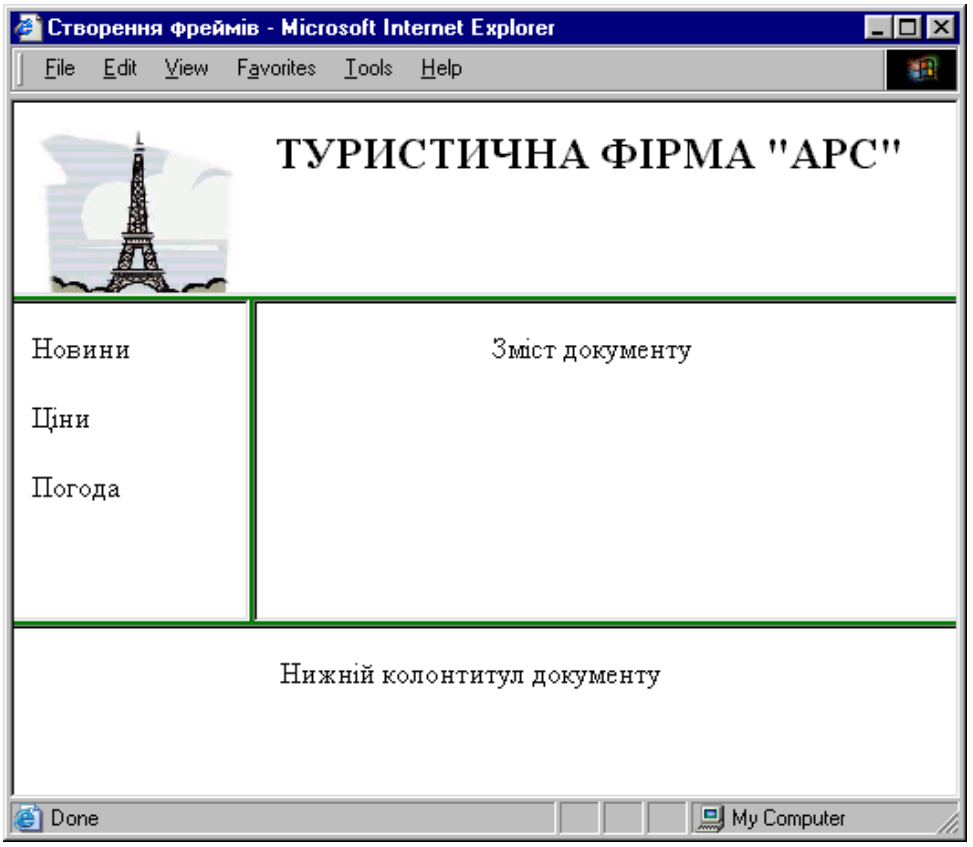


Рис. 7.5. Відображення у браузері чотирьох фреймів, для яких визначена наявність межі між фреймами та зелений колір меж

8. Визначимо простір усередині кожного фрейма, тобто по- ля, у межах яких не може бути розміщена ніяка інформація. Реа- лізуємо це за допомогою параметрів *marginheight* та *marginwidth*, значення яких буде дорівнювати *1* пікселю.

Запишемо HTML-код та переглянемо документ у вікні брау- зера (рис. 7.6):

<frameset rows=«25%,50%,25%»>
<frame src=«logo.html» scrolling=«no» frameborder=1
bordercolor=«green» margingheight=«1» marginwidth=«1»>
<frameset cols=«25%,*»>
<frame src=«menu.html» scrolling=«no» frameborder=1
bordercolor=«green»margingheight=«1» marginwidth=«1»>

<frame src=«content.html» frameborder=1 bordercolor=«green» margingheight=«1» marginwidth=«1»>
</frameset>
<frame src=«bottom.html» frameborder=1 bordercolor=«green»margingheight=«1» marginwidth=«1»>
</frameset>

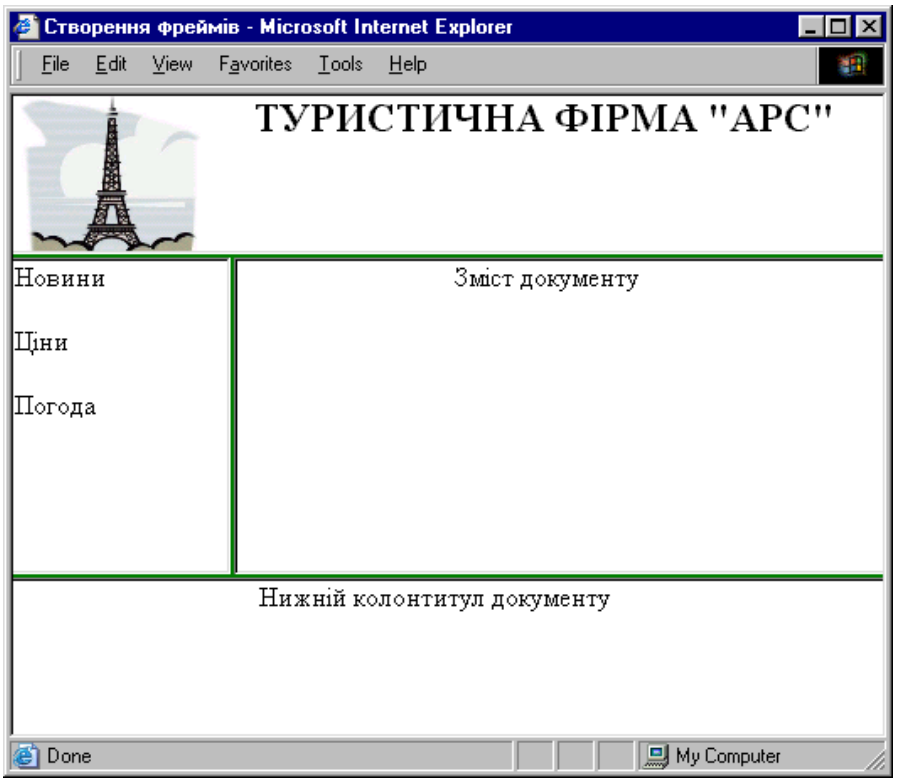


Рис. 7.6. Відображення у браузері чотирьох фреймів, для яких визначено параметри marginheight та marginwidth, значення яких дорівнює 1 пікселю

Змінимо значення параметрів *marginheight* та *marginwidth* з *1* на *20* пікселів для фреймів logo та menu. Переглянемо новий документ на рис. 7.7.

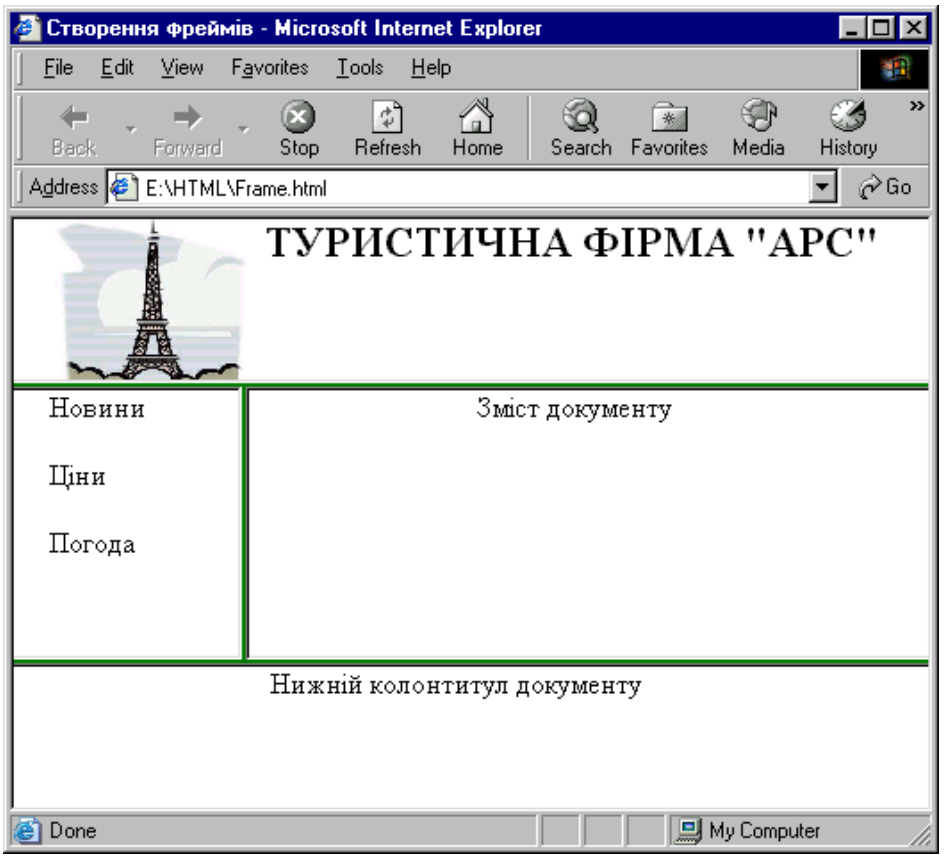


Рис. 7.7. Відображення у браузері чотирьох фреймів, для яких визначено параметри marginheight та marginwidth, значення яких дорівнює 20 пікселів

9. Заборонимо користувачам змінювати розмір фреймів, що може порушити структуру спроектованих нами фреймів. Реалі- зуємо це за допомогою параметра noresize тегу *<frame>*, який не потребує ніяких значень.

10. Визначимо взаємодію між фреймами.

У файлі menu.html створимо гіперпосилання, перехід по яких буде завантажувати файл з іменем example.html у визначений фрейм. Для цього:

10.1. Створимо файл example.html у папці HTML та запишемо для нього HTML-код:

```
<html><head><title>example</title></head>
<body>
Текст документа
</body>
</html>
```

10.2. Задамо ім’я для фрейма, на який буде гіперпосилання (це фрейм content). Скорегуємо HTML-код для файла Frame. html – фрейму content дамо ім’я «A»:

```
<frameset rows=«25 %,50 %,25 %»>
<frame src=«logo.html» scrolling=«no» frameborder=1
bordercolor=«green» margingheight=«20» marginwidth=«20»>
<frameset cols=«25 %,*»>
<frame src=«menu.html» scrolling=«no» frameborder=1
bordercolor=«green»margingheight=«20» marginwidth=«20»>
<frame src=«content.html» frameborder=1 bordercolor=«green»
margingheight=«20» marginwidth=«20» name=«A»>
</frameset>
<frame src=«bottom.html» frameborder=1
bordercolor=«green»margingheight=«20» marginwidth=«20»>
</frameset>
```

10.3. Скорегуємо HTML-код для файла menu.html: додамо ще один пункт у меню (назвемо його «Повідомлення»), визначимо гіперпосилання на файл example.html:

```
<html>
<head><title>menu</title></head>
<body>
<a href=«example.html»target=«A»>Новини</a><br><br>
<a href=«example.html»target=«_blank»>Ціни</a><br><br>
<a href=«example.html»target=«_top»>Погода</a><br><br>
<a href=«example.html»target=«_self»>Повідомлення</a>
</body>
</html>
```

Документ menu.html має чотири гіперпосилання на файл з іменем example.html, але з різним значенням параметра target. Перше гіперпосилання зі значенням *target=«A»* буде завантажувати файл example.html у фрейм з іменем *«A»*. Друге гіперпосилання зі значенням *target=«_blank»* створить нове вікно без імені та завантажить туди необхідний документ. Третє гіперпосилання зі значенням *target=«_top»* завантажить документ у повне вікно замість усієї фреймової структури. Четверте гіперпосилання зі

значенням *target=«_self»* завантажить документ у фрейм menu замість документа із гіперпосиланнями. Переглянемо оновлений файл Frame.html у вікні браузера (рис. 7.8).

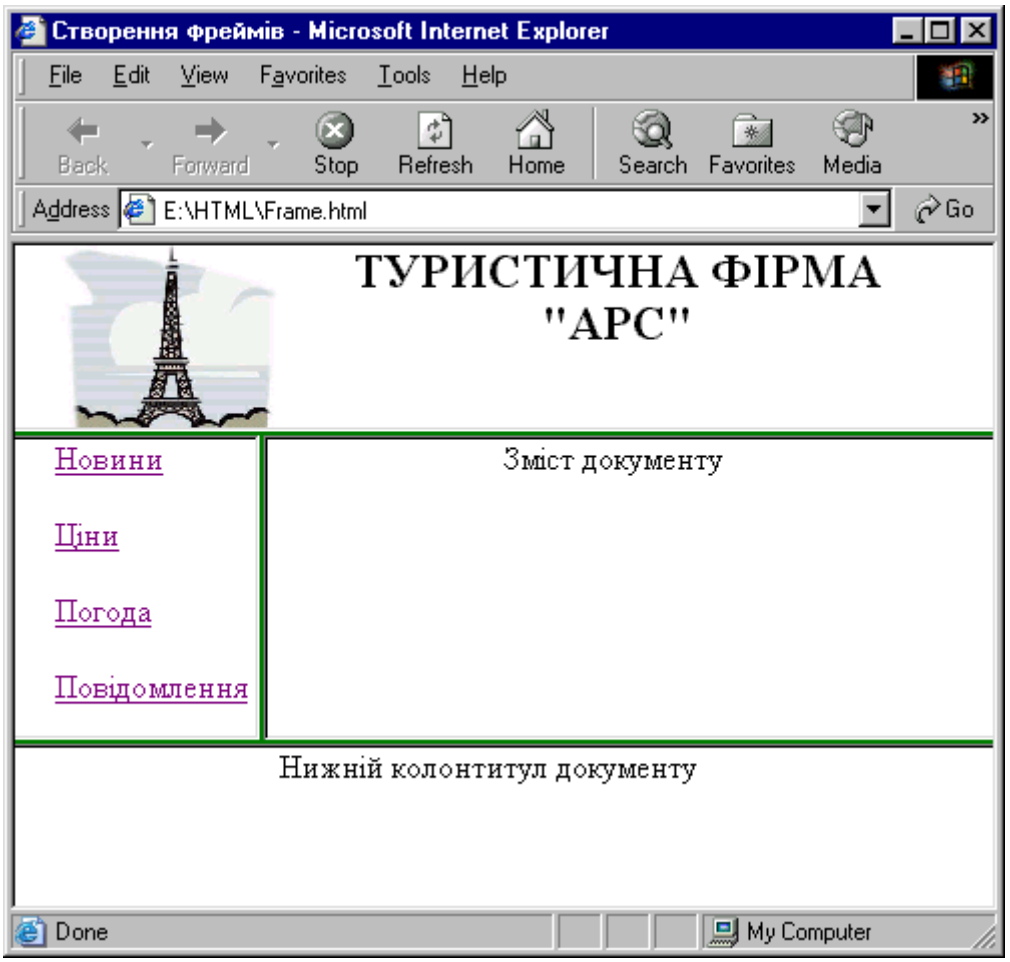


Рис. 7.8. Відображення оновленого документа Frame.html – у фреймі menu додано один пункт меню та визначено гіперпо- силання

Переглянемо у вікні браузера ситуації після реалізації чотирьох посилань на рис. 7.9, 7.10, 7.11.

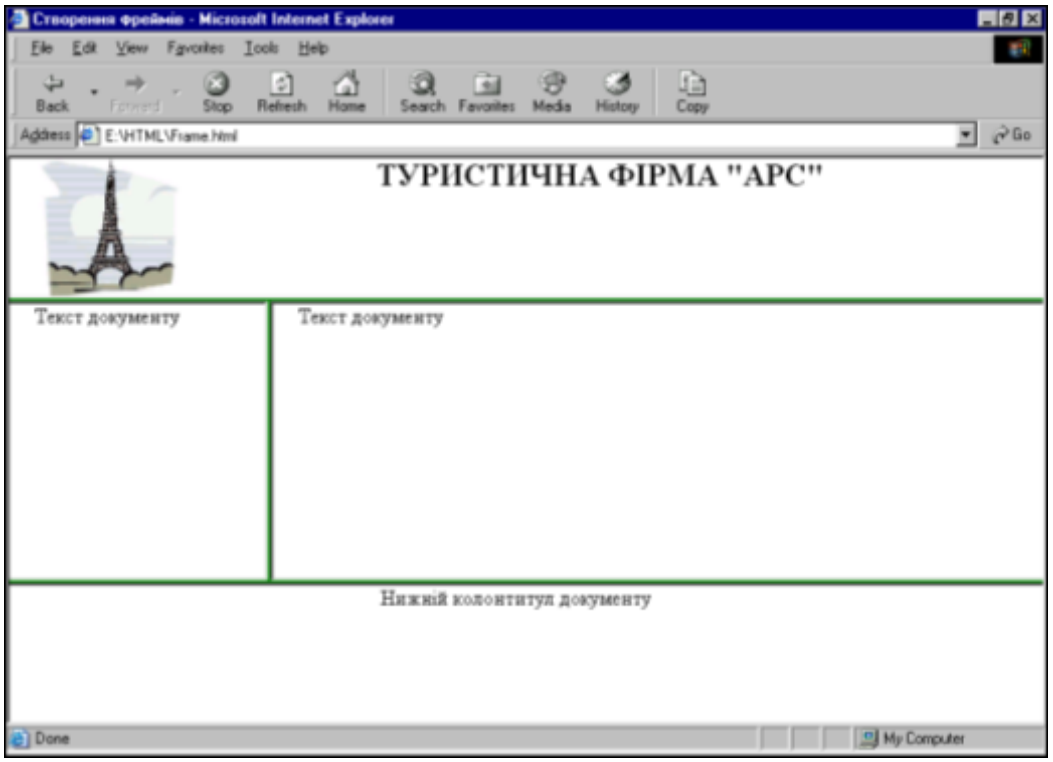


Рис. 7.9. Ситуація, отримана після послідовної реалізації першого та четвертого посилань

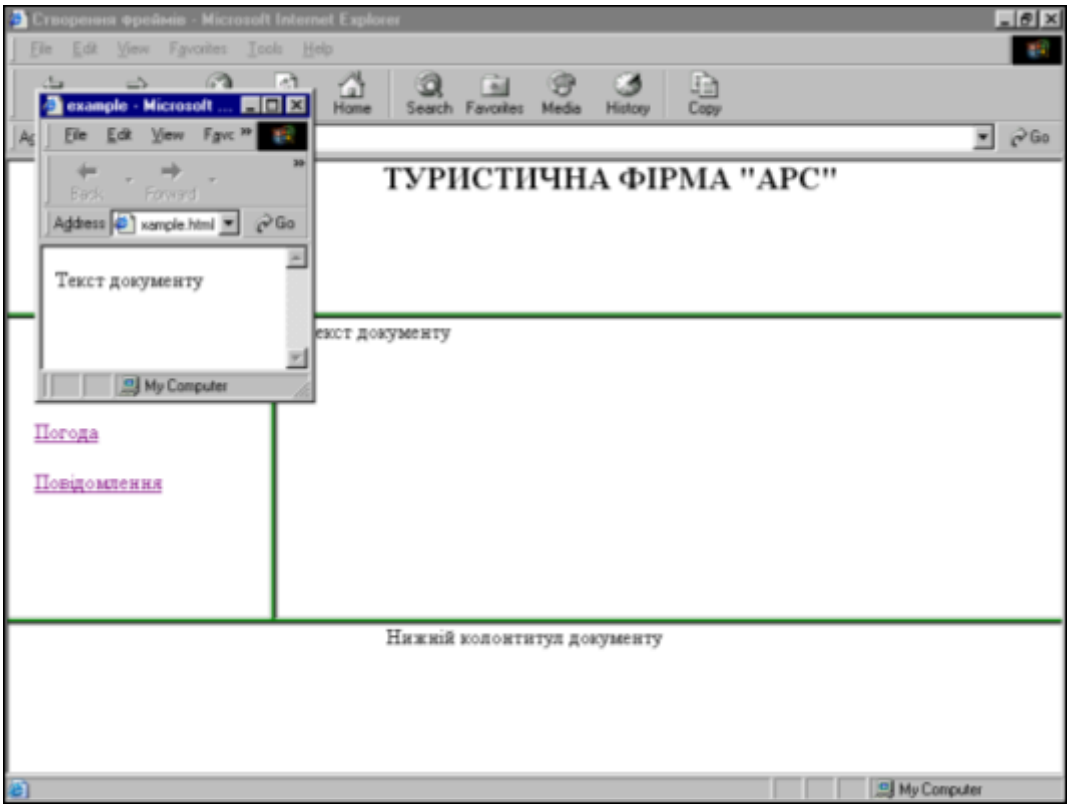


Рис. 7.10. Ситуація, отримана після реалізації другого посилання

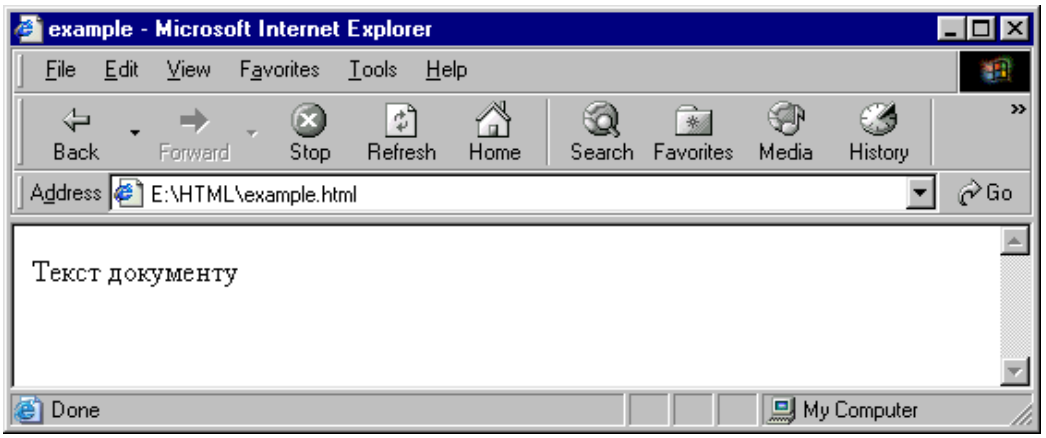


Рис. 7.11. Ситуація, отримана після реалізації третього посилання

Повторимо друге посилання – файл example.html відкриється у новому вікні браузера. Реалізуємо третє посилання – нове вік- но браузера не створюється, файл example.html відкривається у тому самому вікні, де був фрейм. Повертаємось до фреймової структури за допомогою кнопки Back. Зверніть увагу, що кнопка Back активна на рис. 11 на відміну від рис 7.10.

11. За допомогою парного тегу <*iframe*> у документі logo1.html визначимо плаваючий фрейм, у якому буде відображатись HTML-файл float.html. Для цього:

11.1. У папці HTML створюємо файл float.html та записуємо у ньому такий HTML-код:

```
<html>
<head>
<title>float</title>
</head>
<body>
<ul>
<h2>Тури:</h2>
<li>ОАЕ
<li>Болгарія
<li>Італія
<li>Таїланд
</ul>
</body>
</html>
```

11.2. Створюємо файл logo1.html, який є копією файла logo.html та буде розміщений у папці HTML. Корегуємо HTML-код документа logo1.html:

```
<html>
<head>
<title>logo1</title>
</head>
<body>
<img src=«Eifel.jpg» height=«110» align=«left»>
<h2 align=«center»>ТУРИСТИЧНАФІРМА «АРС»</h2>
<iframe src=«float.html» height=«150» width=«270» scrolling=«yes»
align=«right» valign=«bottom» hspace=«10» vspace=«10»>
Ваш браузер не дозволяє відображати плаваючі фрейми
</iframe>
</body>
</html>
```

Переглянемо результат відображення цього HTML-коду у вікні браузера (рис. 7.12).

Якщо браузер не підтримує концепцію плаваючих фреймів, то у цьому випадку замість змісту документа float.html у ньому буде відображено текст. Ваш браузер не дозволяє відображати плаваючі фрейми.

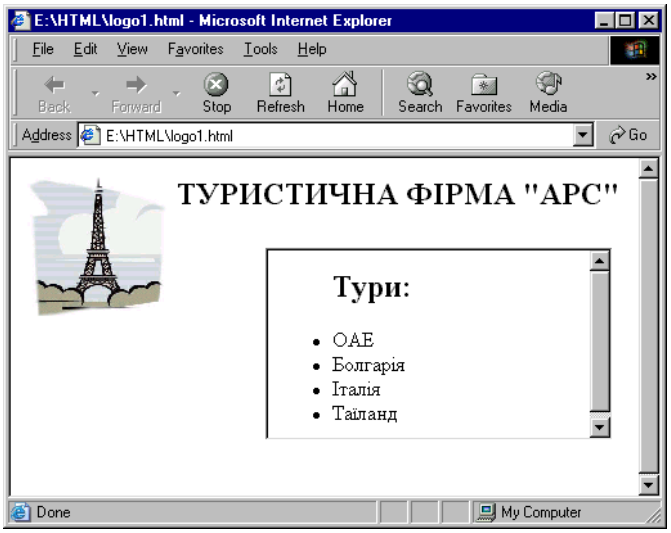


Рис. 7.12. Відображення плаваючого фрейма у вікні браузера

12. Визначити фрейм зі структурою, вказаною викладачем.
13. Оформити звіт.

Запитання для самоперевірки

1. Навіщо використовуються фрейми?
2. Як визначити кількість горизонтальних фреймів?
3. Як визначити кількість вертикальних фреймів?
4. Як визначити розмір фреймів в абсолютних величинах?
5. Як визначити розмір фреймів у відносних величинах?
6. Як встановити наявність границь між фреймами?
7. Як визначити, який фрейм буде метою гіперпосилання?
8. Як визначити полосу прокрутки у фреймі?
9. Як встановити розміри плаваючого фрейма?
10. Як заборонити користувачеві змінювати розміри фреймів?

ЛАБОРАТОРНА РОБОТА 8

Тема. Форматування елементів Web-сторінки за допомогою каскадних таблиць стилів

Мета: освоїти практичні навички використання каскадних таблиць стилів.

Хід роботи

1. Підготовча робота. У папці HTML створимо файл css.html та запишемо у ньому такий HTML-код:

```
<html><head>
<meta http-equiv=«Content-Type» content=«text/html;
charset=windows-1251»>
<style type=«text/css»></style>
<title>Використання каскадних таблиць стилів</title>
</head><body>
<h1>Заголовок першого рівня №1</h1>
<p>Текст першого абзацу</p>
<h1>Заголовок першого рівня №2</h1>
<h2>Заголовок другого рівня</h2>
<p>Текст другого абзацу</p>
</body></html>
```

Відповідне вікно браузера подане на рис. 8.1.

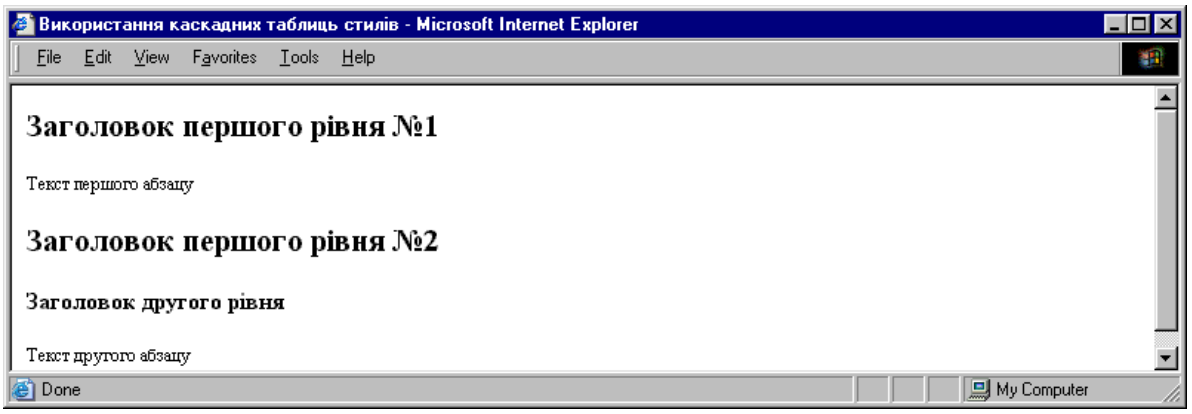


Рис. 8.1. Відображення заголовка першого рівня до використання стилів

2. Визначимо стиль, що приводить до появи границь товщиною 1 піксель навколо всіх елементів *h1* та вирівнювання цих елементів по центру:

```
<style type=«text/css»>
h1 {border-width: 1; border: solid; text-align: center}
</style>
```

Після застосування такого стилю вікно відображення заголовків першого рівня зміниться відповідно до рис. 8.2.

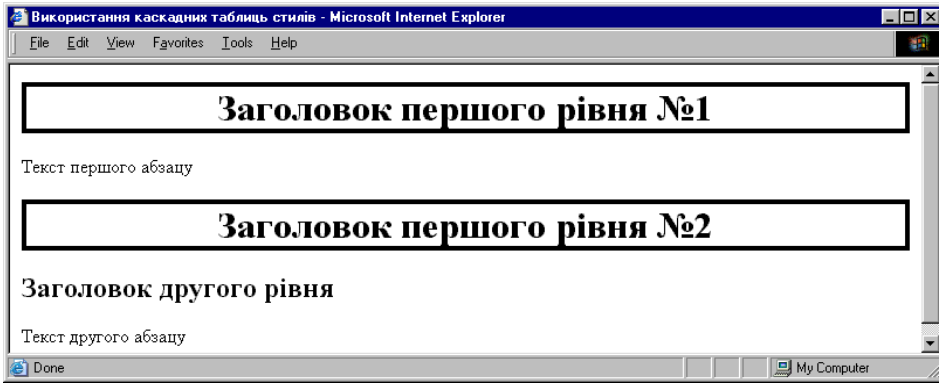


Рис. 8.2. Застосування стилів у заголовках першого рівня

3. Модифікуємо визначення стилю таким чином, щоб він використовувався тільки елементами *h1*класу *m*:

h1.m {border-width: 1; border: solid; text-align: center}

4. Першому елементу *h1* призначимо клас *m*:

<h1 class='m'>Заголовок першого рівня №1</h1>

Оновлене відображення HTML-документа у вікні браузера повинно відповідати рис. 8.3.

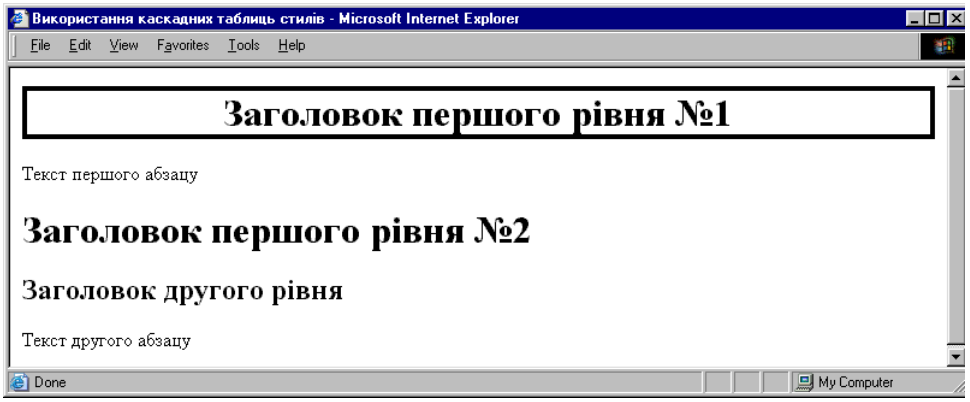


Рис. 8.3. Використання класів стилів

5. За допомогою контекстного селектора визначимо стиль для відображення заголовків другого рівня та абзаців шрифтом *Monotype Corsiva*:

```
...
h2, p {font-family: ‘Monotype Corsiva’;}
</style>
```

Переглянемо модифікований документ (рис. 8.4).

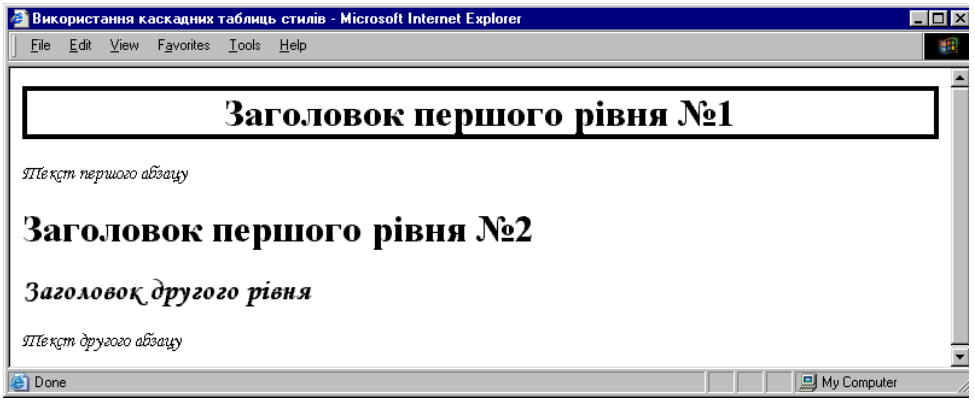


Рис. 8.4. Використання контекстного селектора

6. У першому абзаці додамо стиль для відображення тексту цього абзацу червоним кольором з нижньою границею типу *double* та верхньою границею зеленого кольору типу *dotted*:

```
<p style=«border-bottom-style: double; color: Red; border-top-color:
Green; border-top-style: dotted;»>Текст першого абзацу</p>
```

Відображення цього абзацу у вікні браузера подане на рис. 8.5.

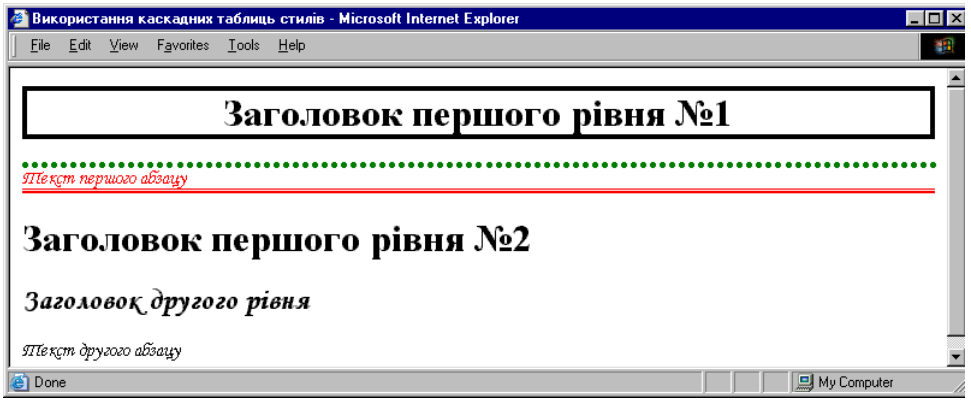


Рис. 8.5. Впровадження стилю у тег абзацу

7. Скопіюємо у папку HTML два рисунки, наприклад 1.jpg та Eifel.jpg.

8. У файлі css.html додамо HTML-код для показу на Web-сторінці цих рисунків з границею, товщиною 1 піксель. Рисунку Eifel.jpg призначимо вирівнювання «по лівому краю», а рисунку 1.jpg присвоїмо id=‘pic’:

…
<img src=«Eifel.jpg» border=«1»>
<img src=«1.jpg» border=«1» id=«pic»>
</body>

9. Додамо стиль для абсолютного позиціювання об’єкта з id=‘pic’ відносно вікна браузера з параметрами top=90 пікселів та left=450 пікселів:

...
#pic {position: absolute; top: 90px; left: 450px}
</style>

Пересвідчимося, що розміщення рисунка 1.jpg (рис. 8.6) значно відрізняється від звичайного розміщення, заданого за допомогою параметра align тегу <img>.

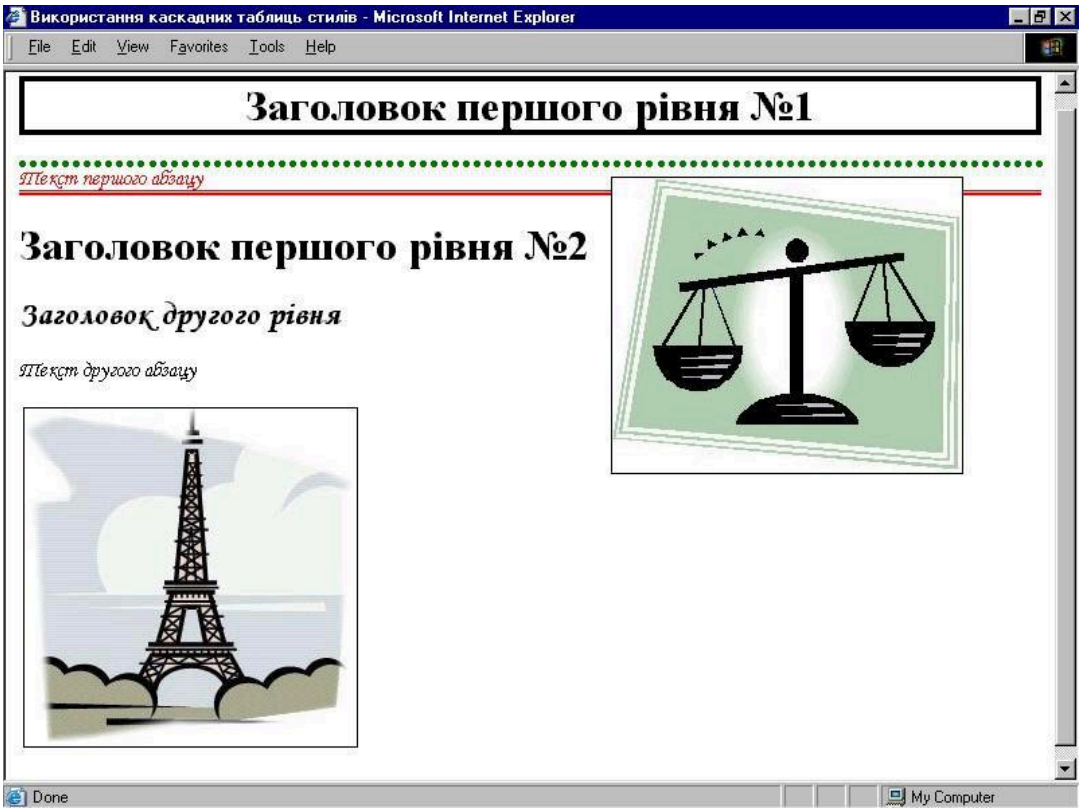


Рис. 8.6. Абсолютне позиціювання рисунка за допомогою CSS

10. Рисунка Eifel.jpg призначимо id=«eifel»:
<img src=«Eifel.jpg» border=«1» id=«eifel»>

11. Додамо стиль для того, щоб об’єкт з id=«eifel» не відображався у вікні браузера:

...
#eifel {display: none}
</style>

Пересвідчимося, що рисунок Eifel.jpg не відображається у вікні браузера (рис. 8.7).

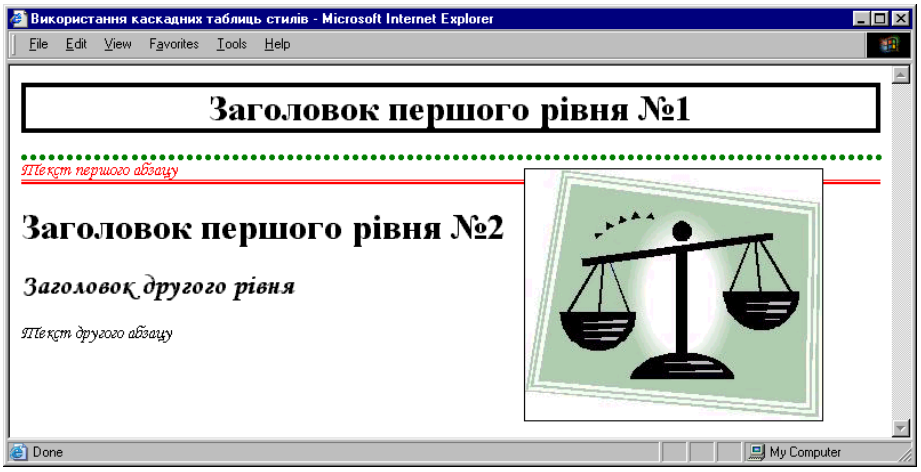


Рис. 8.7. Знищення рисунка на екрані завдяки використанню CSS

12. Модифікуємо стиль об’єкта з id=«eifel» для його відображення на екрані:
#eifel {display: none}

Пересвідчимося, що рисунок Eifel.jpg знову відображається у вікні браузера (рис. 8.6).

13. Виділимо визначені нами стилі в окремий файл та зв’яжемо його з нашим HTML-документом (css.html). Для цього:

13.1. У папці HTML створимо текстовий документ, назвемо його mystyle.css та відкриємо за допомогою текстового редактора «Блокнот».

13.2. Із файла css.html перенесемо у mystyle.css усі визначені нами таблиці стилів:
h1.m {border-width: 1; border: solid; text-align: center}
h2, p {font-family: ‘Monotype Corsiva’;}

#pic {position: absolute; top: 90px; left: 450px} #eifel {display: block}

Зазначимо, що теги <style type=«text/css»> та </style> записувати не потрібно.

13.3. За допомогою меню «Файл»->«Сохранить» збережемо файл mystyle.css;

13.4. У файлі css.html замість тегів *<style type=«text/css»> ... </style>* запишемо.

<link href=«mystyle.css» rel=«stylesheet» type=«text/css»>.

13.5. Збережемо файл css.html.

14. Проведемо оновлення нашої Web-сторінки у вікні браузера. Якщо у процесі виконання не було допущено помилок, то відображення Web-сторінки повинно залишитися без змін (рис. 8.6).

15. Розробити таблиці стилів для форматування 2–3 сторінок сайта з тематики, визначеної викладачем.

16. Оформити звіт.

Запитання для самоперевірки

1. Як встановити абсолютну позицію об’єкта відносно вікна браузера?
2. Як за допомогою стилів знищити зображення об’єкта з вікна браузера?
3. Як визначити стиль, що призведе до відображення зеле- ним кольором тексту, розміщеного у таблицях Web-сторінки?
4. Як визначити стиль, що призведе до відображення всіх заголовків шрифтом Arial?
5. Записати стиль для відображення нижньої границі тексту абзаців?
6. Записати стиль для відображення верхньої границі текс- ту абзаців?
7. Як зв’язати HTML-документ з таблицями стилів, що ви- значені в окремому файлі?
8. Навіщо використовується групування селекторів?
9. Навіщо у тегах використовується параметр id?

10. Як при визначенні правил стилів використовуються класи?

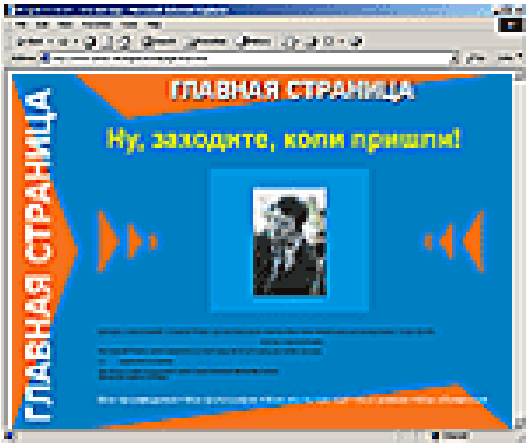
ЛАБОРАТОРНА РОБОТА 9

Тема. Моя особиста web-сторінка

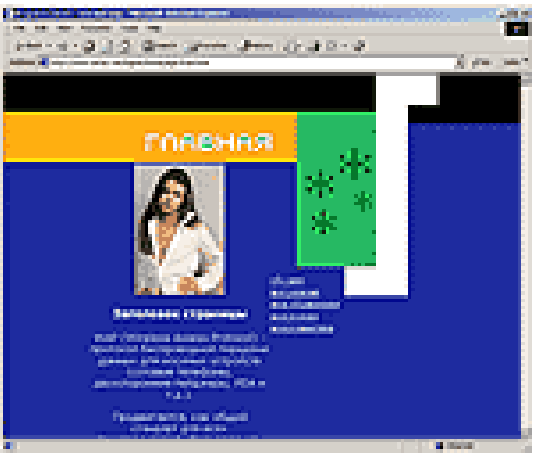
Мета: набути навичок роботи щодо створення особистої web-сторінки**.**

Хід роботи

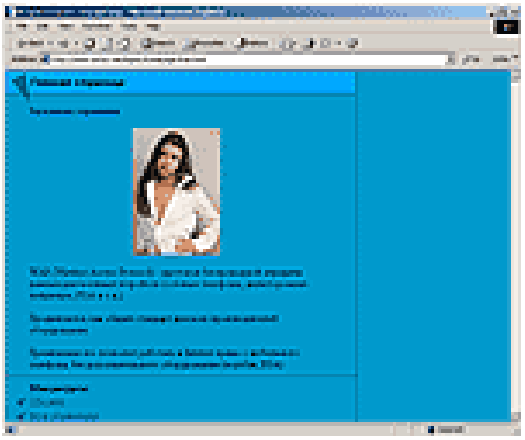
1. Створити свою особисту сторінку з інформацією про себе (по типу резюме). Використовувати вивчені теги. Самостійно створити фон сторінки, малюнки (бажано відскановані фотографії), використовувати різні типи шрифтів, абзаци, списки і стилі (викладач по варіантах задає вид таблиці, кількість і розташування фреймів).



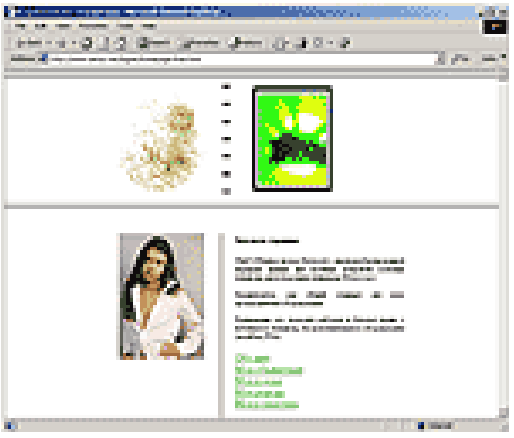
Трикутний дизайн



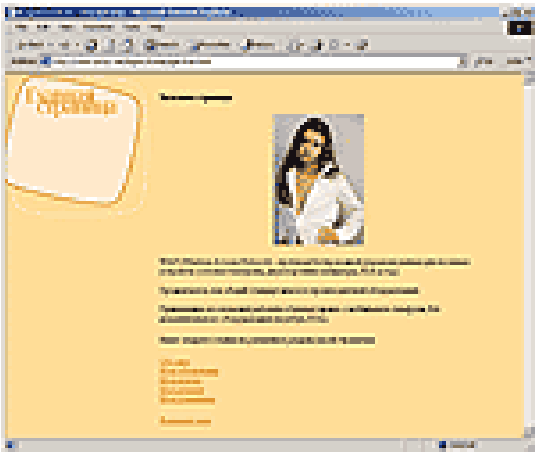
Зимовий дизайн



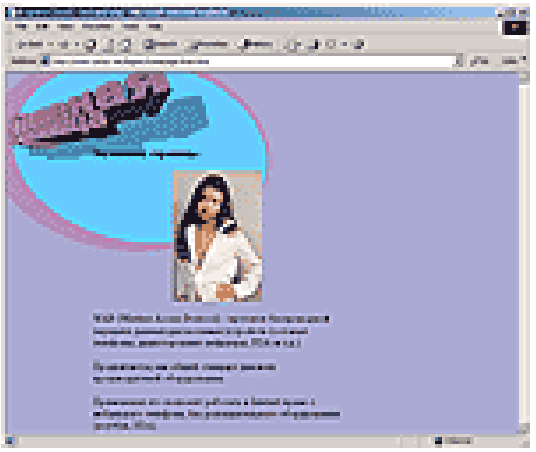
Дизайн: косі трикутники



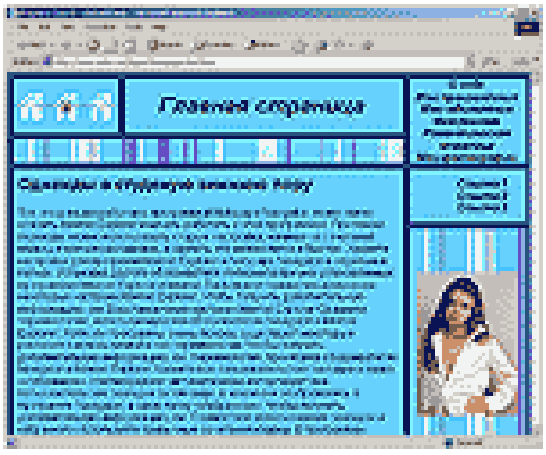
Кислотний дизайн із відбитками пальців



Дизайн: телевізор



Дизайн: тривимірна проекція



Дизайн: вертикальні смужки

Дизайн: ілюмінатори

2. Оформити і захистити звіт.

ЛАБОРАТОРНА РОБОТА 10

Тема. Створення інтерактивних Web-документів

Мета: освоєння практики створення інтерактивних Web-документів засобами JavaScript та CSS.

Хід роботи

1. У папці HTML створити файл anim.html та записати у ньому такий код («кістяк» HTML-сторінки):

```
<html>
<head>
    <meta http-equiv=«Content-Type» content=«text/html; charset=windows-1251»>
    <title>Створення інтерактивних Web-документів</title>
    <script>
    </script>
</head>
<body>
</body>
</html>
```

2. Створимо скрипт для того, щоб ця Web-сторінка завжди відкривалась у повновіконному режимі. Для цього:

2.1. Визначаємо вертикальний та горизонтальний розмір екрана:

```
...
    h=window.screen.height
    ;
    w=window.screen.width
    ;
</script>
```

2.2. Визначаємо функцію, яка буде переміщати вікно з нашою Web-сторінкою у лівий верхній кут екрана та встановить розміри вікна, рівні розмірам екрана:

```
...
```

```
function wr() {
window.moveTo(0,0);
```

```
window.resizeTo(w,h);
}
</script>
```

2.3. Модифікуємо тег <body> для звернення до функції wr() при завантаженні документа у вікно браузера:

```
<body onload=«wr()»>
```

3. Створимо на нашій Web-сторінці вертикальне меню із двох елементів (кнопок). Вибір користувачем певного елемента меню повинен приводити до показу на сторінці відповідної інфор- мації. Для цього:

3.1. Додамо на нашу Web-сторінку три таблиці. У першій таблиці розмістимо два управляючі елементи меню (кнопки), у другій таблиці запишемо зміст першого розділу меню, а у третій – зміст другого розділу. Другій таблиці присвоїмо *id=«r1»*, а третій – *id=«r2»*:

```
...
<table align=«left»>
    <tr><td><input type=«button» value=«Перший пункт меню»
style=«width: 200px;»></td></tr>
0<tr><td><input type=«button» value=«Другий пункт меню»
style=«width: 200px;»></td></tr>
</table>
<table id=«r1»>
<tr><td>Зміст першого розділу меню (Таблиця 2)</td></tr>
</table>
<table id=«r2» style=«display: none;»>
<tr><td>Зміст другого розділу меню (Таблиця 3)</td></tr>
</table>
</body>
```

3.2. Визначимо, що при завантаженні Web-сторінки відобра- жатись буде тільки зміст першого розділу меню, тобто таблиця 3 буде невидимою. Для цього модифікуємо код третьої таблиці:

<table id=«r2» style=«display: none;»>

3.3. При перегляді нашої Web-сторінки зміст другого пункту меню (таблиця 3) повинен бути невидимим (рис. 10.1).

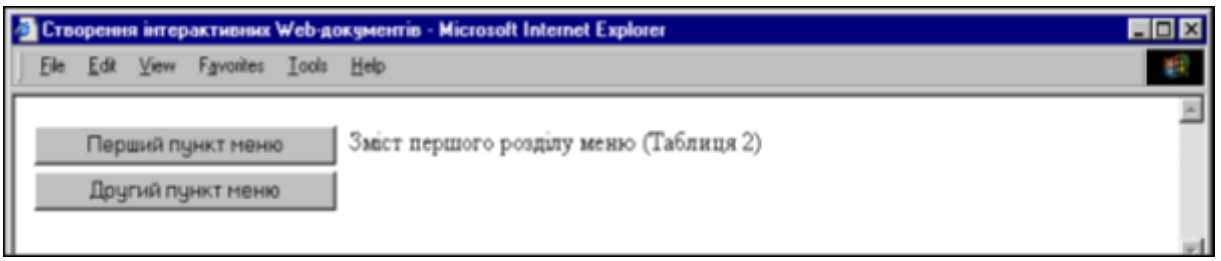


Рис. 10.1. Відображення першого розділу меню при завантаженні Web-сторінки

3.4. Додамо функцію, яка буде відображати на екрані тільки той елемент HTML-сторінки, id якого передається цій функції як параметр:

```
...
function disp(myid) {
document.getElementById(‘r1’).style.display=«none»;
document.getElementById(‘r2’).style.display=«none»;
document.getElementById(myid).style.display=«block»
;
}
</script>
```

3.5. Модифікуємо код кнопок так, щоб вибір користувачем певної кнопки приводив до відображення тільки відповідного розділу меню:

```
<input type=«button» value=«Перший пункт меню» style=«width: 200px;» onClick=«disp(‘r1’)»>
<input type=«button» value=«Другий пункт меню» style=«width: 200px;» onClick=«disp(‘r2’)»>
```

3.6. Перевіримо функціонування меню. При виборі кнопки меню повинен відображатись тільки відповідний розділ (рис. 10.2).

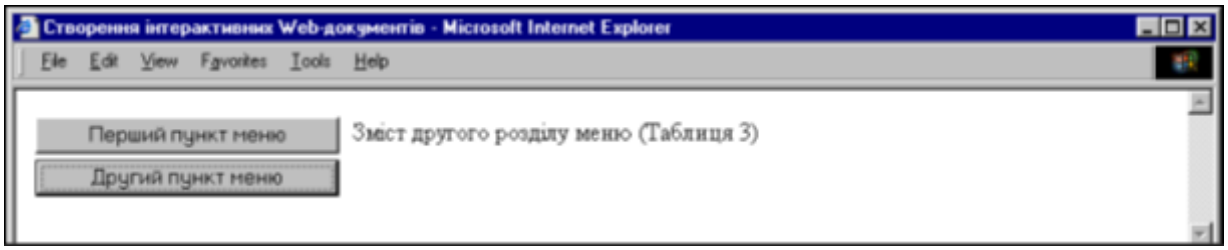


Рис. 10.2. Відображення другого розділу меню

4. Створимо скрипт для прокрутки тексту «Привіт всім!!!» у статусному рядку браузера. Для цього після визначення функції *disp* запишемо наведений нижче програмний код:

```
//визначаємо змінну pos
var pos=0;
//визначаємо функцію для прокрутки тексту
function status() {
//визначаємо текст, призначений для відображення
str=«Привіт всім!!!»;
//визначаємо фрагмент тексту, що буде
//показаний у рядку статуса
//визначення ралізується за рахунок копіювання рядка str
//з символу з номером pos по символ з номером pos+17 у рядок
str1
str1=str.substring (pos, pos+27);
//показуємо змінну str1 у рядку статуса
window.defaultStatus=str1;
//збільшуємо значення змінної pos на 1
pos++
//перевіряємо значення змінної pos
if (pos == 27) pos=0;
//рекурсивний виклик функції status з частотою 1 раз у 0,03 сек.
setTimeout(«status()»,30);
}
</script>
```

Зазначимо, що у даному випадку коментарі призначені для кращого розуміння принципів функціонування скрипта. З цієї причини записувати їх не обов’язково.

5. Модифікуємо функцію *wr* (викликається внаслідок реалі- зації події onload) для того, щоб прокрутка починалась після за- вантаження HTML-сторінки у вікно браузера:

```
function wr() {
window.moveTo(0,0);
window.resizeTo(w,h);
status();
myopen();
}
```

6. Перегляд HTML-сторінки (рис. 10.3) повинен підтвердити прокрутку тексту у статусному рядку.

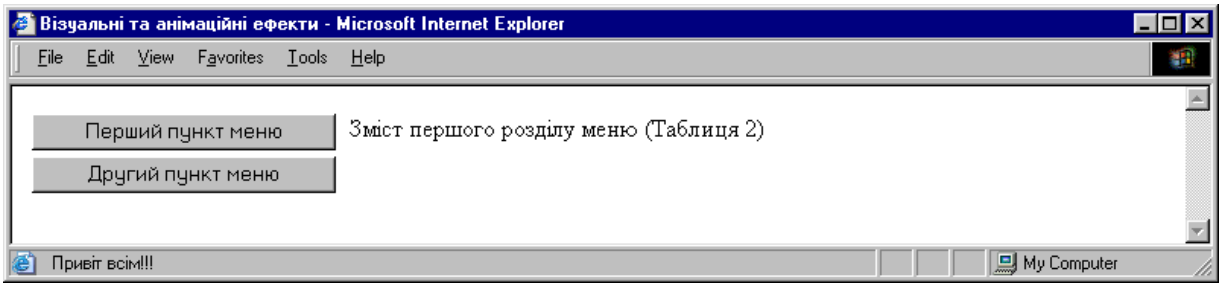


Рис. 10.3. Прокрутка тексту у статусному рядку

7. Створимо функцію *myopen*, що реалізує завантаження нового HTML-документа у нове вікно браузера. Подібні функції досить часто використовуються для показу користувачеві короткої та особливо актуальної інформації сайта. Таку інформацію можливо показувати у вікні браузера без панелі інструментів, рядка меню та смуг прокрутки. Для цього використаємо метод *open* стандартного об'єкта *window*. Методу *open* необхідно передати три параметри: ім'я файла, який буде відображено у новому вікні, ім'я та параметра вікна (параметри задаються одним рядком). У даному випадку параметри такі: панель інструментів, рядок меню та смуги прокрутки відсутні, ширина вікна 300, а висота 140 пікселів, лівий верхній кут вікна знаходиться нижче на 100 і лівіше на 200 пікселів від верхнього лівого кута екрана. Запишемо код функції *myopen*:

```
...
function myopen() {
    /* Увага! Значення змінної str необхідно записати в одному
рядку без переносута пробілів. */
str=«toolbar=0,menubar=0,Scrollbars=0,width=300,height=140,
top=100,left=200»;
window.open(«info.html»,»newa»,str);
}
</script>
```

8. У папці HTML створимо файл info.html та визначимо у ньому такий HTML-код:

```
<html>
```

```
<head>
<title>Увага!!! Актуальна інформація!!!</title>
</head>
<body>
<h1 align=«center»>Увага!!! <br>Актуальна інформація!!!</h1>
</body>
</html>
```

9. Зазначимо, що показувати користувачеві актуальну інфор- мацію доцільно при перегляді ним певної сторінки сайта. Тому виклик функції *myopen* реалізуємо у функції *wr*, яка, у свою чер- гу, викликається при завантаженні нашої HTML-сторінки (*anim.html*). Для цього модифікуємо код функції *wr* так:

```
function wr() {
window.moveTo(0,0);
window.resizeTo(w,h);
status();
myopen();
}
```

10. Під час відкриття HTML-документа *anim.html* повинно відкритись нове вікно браузера із завантаженим у ньому файлом *info.html* (рис. 10.4).

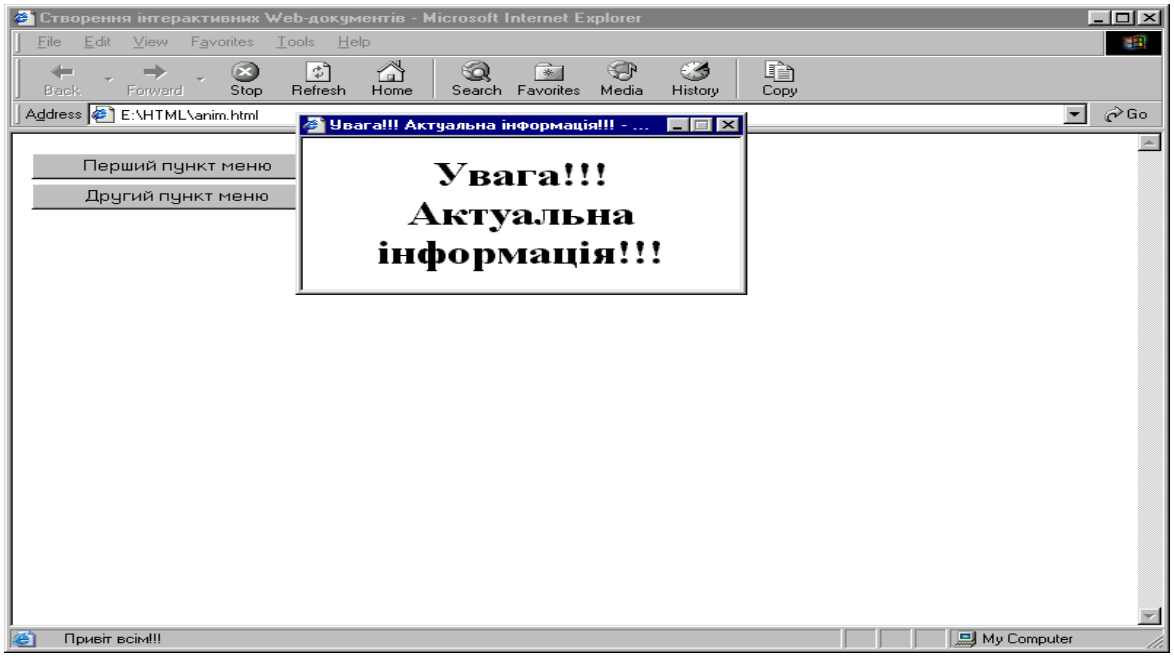


Рис. 10.4. Відкриття нового вікна браузера

11. Створити меню із 5 пунктів. Вибір пункту меню повинен привести до відображення на Web-сторінці певного рисунка.

12. Модифікувати функцію *wr* так, щоб вікно браузера займало тільки половину екрана.

13. Оформити звіт.

Запитання для самоперевірки

1. Яке призначення методу open об’єкта window?
2. Які параметри необхідно передати методу open об’єкта window?
3. Яка подія відповідає завантаженню HTML-документа у вікно браузера?
4. Яка подія відповідає вибору користувачем певного елемента Web-сторінки?
5. Яким чином можливо реалізувати рекурсивний виклик функції через певний проміжок часу?
6. Яке призначення методу moveTo об’єкта window?
7. Яке призначення методу resizeTo об’єкта window?
8. Як визначити власну функцію у коді JavaScript?
9. Як скопіювати частину символів із однієї змінної в іншу?
10. Як показати інформацію у рядку статусу вікна браузера?

ЛАБОРАТОРНА РОБОТА 11

Тема. Створення форми
Мета: навчитися створювати форми за допомогою тегів.

Хід роботи

У кожному завданні додати два посилання на свою особисту сторінку (перша – звичайна, друга – у вигляді малюнка), створену у попередній лабораторній.

Варіант 1

Створити форму для введення інформації про людину. Повинна вводитися така інформація:

Прізвище	Просте текстове введення
Ім’я	Просте текстове введення
По батькові	Просте текстове введення
Вік	Вибір з варіантів «до 7», «від 7 до 17», «від 17 до 33», «від 33 до 65» і «65 і старше»
Стать	Повинна бути «радіокнопка»
чи Пристойна людина	Повинен бути «чек-бокс»

У формі повинна бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 2

Створити форму для введення інформації про людину. Повинна вводитися така інформація:

Прізвище, ім’я, по батькові	Просте текстове введення
Рік народження	Вводиться текстом. Заборонити введення більш ніж чотирьох символів

Стать	Повинна бути «радіокнопка»
Громадянство	Вибір зі списку держав

У формі повинна бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 3

Створити форму для введення адреси. Повинна вводитися така інформація:

Країна	Вибір зі списку. За замовчуванням – Україна
Поштовий індекс	Просте текстове введення, не більш шести символів
Місто	Просте текстове введення
Вулиця, номер будинку	Просте текстове введення
Вказівка на те, що адреса служ-бова/домашній	Повинна бути «радіокнопка»

У формі повинна бути кнопка «Очистити форму» (*звичайно, вона повинна правильно працювати)*.

Варіант 4

Створити форму для введення замовлення на комп’ютер. Повинна вводитися така інформація:

Процесор	Вибір зі списку. За замовчуванням Intel Pentium II
Пам’ять	Просте текстове введення
Необхідний мережний ада-птер	Повинен бути «чек-бокс»
Необхідний модем	Повинен бути «чек-бокс»
Додаткові вимоги	Багатострокове поле введення

У формі повинна бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 5

Створити форму для замовлення товарів. Форма повинна пропонувати замовити відразу кілька товарів. Для кожного слід указувати кількість. Один раз слід указати адресу доставки. Повинна вводитися така інформація:

Найменування товару	Вибір зі списку. Зробіть кілька однакових списків, щоб можна було замовити відразу кілька різних товарів
Кількість	Уводиться текстом. Потрібно зробити окреме поле для введення кількості, напроти кожного списку товарів
Адреса доставки	Текст

У формі повинна бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 6

Створити форму для запиту пароля. Повинна вводитися така інформація:

Ім’я користувача	Уводиться текстом
Ресурс	Вибір зі списку. Передбачите кілька ресурсів Наприклад, «Перегляд бази даних», «Редагування бази даних» та ін. За замовчуванням нехай буде «Повний доступ»
Пароль	Стандартне поле для введення пароля

У формі повинна бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 7

Створити форму для замовлення на ремонт комп’ютера. Повинна вводитися така інформація:

Ім’я користувача	Уводиться текстом
Адреса	Вводиться текстом
Процесор	Вибір зі списку. За замовчуванням Intel Pentium II
Характер несправності	Багатострокове поле введення

У формі повинна бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 8

Створити форму для замовлення на ремонт автомобіля. Повинна вводитися така інформація:

Ім’я власника	Вводиться текстом
Адреса	Вводиться текстом
Модель	Вибір зі списку. За замовчуванням «ВАЗ 2101»
Характер несправності	Багатострокове поле введення

У формі повинна бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 9

Створити форму для замовлення автомобіля. Повинна вво- дитися така інформація:

Ім’я клієнта	Вводиться текстом
Адреса	Вводиться текстом
Модель	Вибір зі списку. Можливо вибрати відразу кілька моделей

Колір	Вибір зі списку. Можливо вибрати відразу кілька кольорів
Мінімальна ціна	Вибір зі списку
Максимальна ціна	Вибір зі списку

У формі повинна бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 10

Створити форму для пошуку в каталозі бібліотеки. Повинна вводитися така інформація:

Тема	Вибір зі списку. Можливо вибрати відразу кілька тем
Автор	Вводиться текстом
Назва	Вводиться текстом
Мінімальний рік видання	Вибір зі списку
Максимальний рік видання	Вибір зі списку

У формі повинна бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 11

Створити форму для гостьової книги. Форма може бути така, як Вам подобається, але обов’язково включити ім’я, URL і e-mail гостюючого й багаторядкове поле введення тексту.

У формі повинна бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 12

Створити форму для пошуку в мережі з можливістю вибору системи пошуку. Повинна вводитися така інформація:

Запит	Вводиться текстом
Система пошуку	Вибір зі списку
Кількість записів, що вертаються	Вибір зі списку. За замовчуванням, не обмежене

У формі повинна бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 13

Створити форму для реєстрації нового користувача в системі. Повинна вводитися така інформація:

Ім’я	Текст
e-mail	Текст
URL	Текст
Пароль	Поле введення пароля
Підтвердження пароля	Поле введення пароля

У формі повинна бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 14

Створити форму для введення інформації про те, які предмети хотів би вивчати студент. Предмети мають бути перераховані по одному на рядкові й перед назвою кожного повинен стояти

«check-box». Початково жоден предмет не повинен бути обраний.

Крім того, форма має містити поле для введення імені й кнопку «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 15

Створити форму для введення інформації з географічної карти. Введення має здійснюватися простим натисканням мишкою на обрану ділянку карти.

Варіант 16

Створити форму, у якій є дві незалежні серії радіокнопок. Наприклад, для введення улюбленого телеканалу. Повинна вводитися така інформація:

Прізвище, ім’я, по батькові	Просте текстове введення
Канал	Серія радіокнопок
Передача	Серія радіокнопок. Наприклад, «Новини», «Серіали», «Мультфільми» та ін.

У формі має бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 17

Створити форму, у якій є як список типу «спадаюче меню», так і простий відкритий список. В останньому повинен бути передбачений одночасний вибір декількох елементів. Заповніть списки значеннями за своїм розсудом.

У формі має бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 18

Створити форму для введення інформації про студента. Повинна вводитися така інформація:

Прізвище	Просте текстове введення
Ім’я	Просте текстове введення
По батькові	Просте текстове введення

Підлога	Повинна бути серія « радіокнопок»
Курс	Вибір зі списку

У формі має бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 19

Створити форму для введення відомості нарахування зарплати. Відомість розрахована на бригаду з 10 людей. Для кожної людини повинна вводитися така інформація:

Прізвище, ім’я, по батькові	Просте текстове введення
Оклад	Просте текстове введення. Зробіть заданий розмір рядка, що вводиться, розумним числом
чи Основне місце роботи	Чек-бокс
Відділ	Серія радіокнопок

У формі має бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 20

Створити форму для введення бухгалтерської проводки. Повинна вводитися така інформація:

Рахунок-дебет	Вибір зі спадаючого меню
Рахунок-кредит	Вибір зі спадаючого меню
Сума	Просте текстове введення. Зробіть фіксований розмір рядка, що вводиться, розумним числом
Особистий код бухгалтера	Вибір зі спадаючого меню
Пароль	Поле для введення паролів

У формі повинна бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 21

Створити форму для висновку бібліографічної інформації. Форма повинна містити опис якоїсь книги (автор, назва, видавництво, рік та ін.) і дві кнопки «така книга» і «Попередня книга».

Варіант 22

Створити форму для введення інформації про бажаний для клієнта варіант міжміського обміну. Повинна вводитися така інформація:

Прізвище, ім’я, по батькові	Просте текстове введення
Адреса	Просте текстове введення
Кількість кімнат у клієнта	Спадаюче меню
Бажані міста	Список з можливістю вибору декількох варіантів
Бажана кількість кімнат	Спадаюче меню
Поверховість	Серія радіо-кнопок. Наприклад, «Будь-яка», «Крім першого», «Крім останнього», «Крім першого й останнього» та ін.
Додаткові умови	Багаторядкове текстове поле.

У формі має бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 23

Створити форму для введення інформації про бажаний для клієнта варіант внутрішньоміського обміну квартири. Повинна вводитися така інформація:

Прізвище, ім’я, по батькові	Просте текстове введення
Адреса	Просте текстове введення
Кількість кімнат у клієнта	Спадаюче меню
Бажані райони	Список з можливістю вибору декількох варіантів
Бажана кількість кімнат	Спадаюче меню
Поверховість	Список з можливістю вибору декількох варіантів
Додаткові умови	Багаторядкове текстове поле

У формі має бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 24

Створити форму для реєстрації нового користувача в системі або просто входу в систему. Повинен бути напис, що новий користувач зобов’язаний ввести пароль двічі для підтвердження, а користувач, який уже був зареєстрований, уводить пароль один раз. Повинна вводитися така інформація:

Ім’я	Текст
e-mail	Текст
URL	Текст
Пароль	Поле введення пароля
Підтвердження пароля	Поле введення пароля

У формі має бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 25

Створити форму для голосування відразу на кілька посад. Списки кандидатів на кожну посаду повинні бути представлені спадаючими меню. Напроти кожного списку написати посаду, на яку претендують кандидати. У списках кандидатів має бути позиція «Проти всіх». Ця позиція й обрається за замовчуванням.

У формі має бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 26

Створити форму для пошуку відповідного будинку в базі даних ріелторської фірми. Повинна вводитися така інформація:

Площа	Вводиться текстом
Поверховість	Спадаюче меню
Район	Список з можливістю вибору декількох варіантів
Наявність комунікацій	Серія чек-боксів. Наприклад, «Водопровід», «Телефон», та ін.
Додаткові вимоги	Багатострокове поле введення

У формі має бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 27

Створити форму для пошуку відносних матеріалів у бібліотеці. Повинна вводитися така інформація:

Ключові слова	Уводиться текстом
Рік публікації	Вибір діапазонів зі спадаючого меню

Характер публікації	Список з можливістю вибору декількох варіантів. Наприклад, «Стаття в журналі», «Дисертація», «Монографія» та ін.
Мова	Список з можливістю вибору декількох варіантів
Доставка	Серія радіокнопок. Наприклад «Підняти в читальний зал», «Доставити на абонемент», і т.д.

У формі має бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

Варіант 28

Форма для введення інформації про те, які журнали читає клієнт. Повинні бути перераховані назви журналів. Напроти кожного, повинен бути чек-бокс *(читає чи ні)* і серія із двох радіо кнопок: «Регулярно», «Час від часу».

У формі має бути кнопка «Очистити форму» *(зрозуміло, вона повинна правильно працювати)*.

Варіант 29

Створити форму для замовлення на відновлення комп'ютера. така інформація повинна вводитися, як для наявного комп'ютера, так і для бажаного (переобладнаного). Форма повинна бути побудована у два стовпчики. Ліворуч інформація про наявне встаткування, праворуч – про бажаний. Звичайно, поле «Додаткові вимоги» повинне бути тільки в бажаного комп'ютера.

Процесор	Вибір зі списку
Пам'ять	Просте текстове введення
Мережний адаптер	Вибір зі спадаючого меню
Додаткові вимоги	Багатострокове поле введення

У формі має бути кнопка «Очистити форму» *(звичайно, вона повинна правильно працювати)*.

ЛАБОРАТОРНА РОБОТА 12

Тема. Використання таблиць стилів
Мета: навчитись використовувати таблиці стилів.

Хід роботи

1. Скопіювати у Блокнот такий текст.

```
<HTML>
<HEAD>
<TITLE>Приклади використання таблиць стилів.</TITLE>
<META HTTP-EQUIV=«Content-Type»
CONTENT=«text/html; Charset=Windows-1251»>
<META NAME=«GENERATOR» CONTENT=«Visual HTML Workshop 2.1 [SNK]»>
<META NAME=«Keywords» CONTENT=«сайт, сторінка, домашня сторінка, HTML-редактор, HTML, WEB-дизайн, Інтернет, каскадні таблиці стилів, CSS, SSI, CSS-команда,»>

<STYLE TYPE=«text/css»>
<!--
BODY
{
background:
ivory; color:
#000000;
text-align: justify;
margin-right: 20;
}
INPUT.20 {
width: 120;
height: 20;
font-family:
Arial; font-size:
9pt;
text-align: center;
```

```
background:
url(/img/wb.gif)
}
INPUT.40 {
```

```
width: 120;
height: 40;
font-family:
Arial; font-size:
9pt;
text-align: center;
background:
url(/img/wb.gif)
}
BUTTON.20 {
cursor:
hand; width:
120;
height: 20;
font-family:
Arial; font-size:
9pt; color: red;
text-align: center;
background:
url(/img/wb.gif)
}
BUTTON.40 {
cursor:
hand; width:
120;
height: 40;
font-family:
Arial; font-size:
9pt; color: red;
text-align: center;
background:
url(/img/wb.gif)
}
A {
text-decoration:none;
font-family: Arial;
```

```
font-size: 8pt; }
H4, H1 { text-align:
center; color: #004080 }
TABLE {text-align: justify;
}
IMG {
padding: 10 10;
```

```
}
.m
{
background:
url(/img/wb.gif); text-align:
center;
padding: 1 1 1 1;
border: 1px dotted
ivory; width: 140;
}
-->
</STYLE>

<style>
td {border: solid 0.1em red; background-color: khaki;}
</style>
</HEAD>
<BODY>
<center>

<!--LiveInternet counter--><script language=«JavaScript»><!--
document.write(‘<img src=«http://counter.yadro.ru/hit?r’+
escape(document.referrer)+((typeof(screen)==‘undefined’)?’’:
‘;s’+screen.width+’*’+screen.height+’*’+(screen.colorDepth?
screen.colorDepth:screen.pixelDepth))+’;’+Math.random()+
‘« width=1 height=1 alt=«»>‘)//--></script><!--/LiveInternet-->

<!--begin of Top100-->
<a href=«http://top100.rambler.ru/top100/»>
<img src=«http://counter.rambler.ru/top100.cnt?359155»
alt=«Rambler’s Top100» width=1 height=1 border=0></a>
<!--end of Top100 code-->

<TABLE WIDTH=«100%» BORDER=10 FRAME=VOID
CELLPADDING=5>
<TR><TD WIDTH=100 VALIGN=TOP>
<P></P>
<table align=center><tr><td class=«m»><a
```

href=‘http://svoisait.ru’» title=««>головна

</td></tr><tr><td class=«m»><a
href='http://svoisait.ru/css/'» title=««>DHTML
</td></tr><tr><td class=«m»><a href='css.shtml'»
title=«КАСКАДНІ ТАБЛИЦІ СТИЛІВ»>таблиці стилів
</td></tr><tr><td class=«m»><a href='css2.shtml'» title=« КАС-
КАДНІ ТАБЛИЦІ СТИЛІВ. Частина дру-
га.»>способи
застосування
</td></tr><tr><td class=«m»><a href='css3.shtml'» title=« КАС-
КАДНІ ТАБЛИЦІ СТИЛІВ. Частина тре-
тя.»>Декілька
прикладів
</td></tr><tr><td class=«m»><a href='css_class.shtml'» title=« КА-
СКАДНІ ТАБЛИЦІ СТИЛІВ. Частина четверта.»>Селектор
CLASS
</td></tr><tr><td class=«m»><a
href='css_yuri.shtml'» title=««>думки про CSS
</td></tr><tr><td class=«m»><a href='filtr.shtml'»
title=««>фільтри
</td></tr><tr><td class=«m»><a href='filtr-2.shtml'»
title=««>Загальні властивості
фільтрів
</td></tr><tr><td class=«m»><a
href='mouse.shtml'»
title=««>Обробка
подій
<input class=«20» title=«Основи мови JavaScript»
TYPE=SUBMIT value=«JavaScript»>
</td></tr><tr><td class=«m»><a
href='http://www.svoisait.ru/css/animal.shtml'» title=«Створення
мультимедійних ефектів за допомогою фільтрів.
Частина перша. Теги-контейнери DIV и SPAN. Абсолютне і від-
носне позиціонування.»>Мультеффекти
</td></tr><tr><td class=«m»><a
href='http://svoisait.ru/css/definitions_js.shtml'» title=«Основи мови
JavaScript. Частина друга. Поняття і визначення. Термінологія
мови програмування JavaScript.»>Термінологія
</td></tr><tr><td class=«m»><a
href='http://svoisait.ru/css/rules_js.shtml'» title=«Основи мови
JavaScript. Частина третя. Правила використання мови створення

сценаріїв JavaScript.»>Тег <script>

</td></tr><tr><td class=«m»><a
href=‘http://svoisait.ru/css/script.shtml’» title=«Практичне застосування JavaScript. Запит користувачу і створення змінної.»>запит користувачу
</td></tr><tr><td class=«m»><a
href=‘http://svoisait.ru/css/script_time.shtml’» title=« Практичне застосування JavaScript. Дата і час.»>дата і час
</td></tr><tr><td class=«m»><a
href=‘http://svoisait.ru/css/variable.shtml’» title=«Змінні, вони і є змінні. У мові написання сценаріїв JavaScript змінні використовуются так само, як і в інших мовах програмування!»>Змінні
</td></tr><tr><td class=«m»><a
href=‘http://svoisait.ru/css/object_js.shtml’»
title=«Посібник JavaScript. Об’єкти в мові
JavaScript.»>об’єкти

</td></tr><tr><td class=«m»><a
href=‘filtr1.shtml’»
title=«»>Допоміжні
матеріали

</td></tr><tr><td class=«m»>
</td></tr></table>
<P></P>

<script type=«text/javascript»><!--
google_ad_client = «pub-0369862403661526»;
google_ad_width = 120;
google_ad_height = 240;
google_ad_format =
«120x240_as»; google_ad_type =
«text_image»;
//2007-10-29: svoi120x240
google_ad_channel = «2020917188»;
//-->
</script>
<script type=«text/javascript»
src=«http://pagead2.googlesyndication.com/pagead/show_ads.js»
>

</script>
<script type=«text/javascript»>
var begun_auto_pad = 87823107;
var begun_block_id = 115576735;

</script>
<script
src=«http://autocontext.begun.ru/autocontext2.js»
type=«text/javascript»></script>
</TD><TD>
<H1>КАСКАДНІ ТАБЛИЦІ СТИЛІВ</H1>
<H4>ДЕКІЛЬКА ПРИКЛАДІВ</H4>

    Враховуючи те, що для деяких (у силу їх характеру або складу розуму) теорія важко перетравлювана речовина, а також те, що теорія вимагає певної «підкованості» у розглянутій області, а сайт призначений саме для новачків у ВЕБ-програмуванні, приводяться декілька практичних прикладів використання таблиць стилів.

    Будь-яке правило каскадних таблиць складається з двох частин: <I> селектора </ I> і <I> визначення </ I>. Селектором може бути будь-який тег HTML. Визначення задає, яким чином необхідно представляти цей тег. Визначення записується у фігурних дужках і, у свою чергу, теж складається з двох частин: <I> властивості </ I> і <I> значення </ I>, які відокремлюються один від одного двокрапкою. В одному правилі можна задавати декілька визначень, відокремлюючи їх один від одного крапкою з комою.

<UL>
<LI>Задання виду заголовків.
H1 {text-align:
center;
color: coral;
font-size: 34pt;
font-family:
«Times New Roman»}

Результат використання
такої таблиці
<H1 style=«text-align:
center; color: coral;
font-size: 34pt;
font-family: «Times New Roman»;>Буде виглядати так.</H1>

Тут: text-align: center; – задає вирівнювання. У даному ви- падку – по центру.
color: coral; – задаємо колір.
font-size: 34pt; – розмір
шрифта.

font-family: «Times New Roman»; – вид шрифта.
<P></P>

<LI>Абзац.
<P style=«text-align:
center; color: blue;
font: oblique 12pt/24pt «Times Cyr», serif;»> Приклад використання таблиці стилів для форматування тексту в окремому абзаці (або у всіх, це вже на Ваш розсуд). Цей абзац відформатований за допомогою таких правил:
P style=«text-align: center;
color:
blue;
 font: oblique 12pt/24pt «Times Cyr»,
serif;»
Властивість oblique в даному випадку задає висоту шрифту 12 пунктів і висоту рядків – 24 пункти.</P>
<LI> такі дві таблиці зроблені за допомогою таких
правил:
<FONT COLOR=«#0000FF»><B>border: solid 0.1em
red; background-color: teal;</B>
<B>td {border: solid 0.1em red; background-color:
khaki;}</B></FONT>
причому перший рядок діє тільки на першу таблицю, оскільки варто в тег TABLE, а друга – на всі таблиці, які є на цій сторінці, оскільки стоїть у тезі style в заголовку сторінки.
<P></P>
<TABLE style=«border: solid 0.1em red; background-color: teal;»
width=90%>
<CAPTION><B><I><FONT COLOR=«#408080»> Поштові
спонсори, з якими варто працювати.
</FONT></I></B></
CAPTION><TR>
<TD><A target=«_blank» HREF=««> WmMail </A></TD>
<TR>
<TD><ATARGET=_blank
HREF=«http://www.wepaid.com/join.php?ref=sergech2»>WePaid</A
></TD>
</TABLE>
<P><!--92700-->
</P>
</BODY>

</HTML>

2. Зберегти документ з назвою CSS1.html, відкрити його за допомогою браузера та перейти до редагування HTML-коду. Зверніть увагу на будову сторінки. У звіті опишіть стилі, які використовуються, і способи завдання.

3. Доповніть документ своїми двома стилями з виводом прі- звища і групи.

4. Збережіть у своїй папці.

5. Скопіювати у Блокнот такий текст.

```
<HTML>
<HEAD>
<TITLE>Таблиці стилів. Селектор CLASS.</TITLE>
<META HTTP-EQUIV=«Content-Type» CONTENT=«text/html;
Charset=Windows-1251»>
<META NAME=«Author» CONTENT=«SERGEI
CHERVONYASHCHY»>
<META NAME=«Keywords» CONTENT=«таблиці стилів, селек-
тор, тег, форматування, сайт, оформлення сайту, HTML, CSS, «>
<STYLE TYPE=«text/css»>
<!--
BODY {
background:
ivory; color:
#000000;
text-align: justify;
margin-right: 20;
}
INPUT.20 {
width: 120;
height: 20;
font-family:
Arial; font-size:
9pt;
text-align: center;
background:
url(/img/wb.gif)
}
```

```
INPUT.40 {
width: 120;
height: 40;
```

```
font-family:
Arial; font-size:
9pt;
text-align: center;
background:
url(/img/wb.gif)
}
BUTTON.20 {
cursor:
hand; width:
120;
height: 20;
font-family:
Arial; font-size:
9pt; color: red;
text-align: center;
background: url(/img/wb.gif)
}
BUTTON.40 {
cursor:
hand; width:
120;
height: 40;
font-family:
Arial; font-size:
9pt; color: red;
text-align: center;
background:
url(/img/wb.gif)
}
A {
text-decoration:none;
font-family: Arial;
font-size: 8pt; }
H4, H1 { text-align:
center; color: #004080 }
```

```
TABLE {text-align: justify;
}
IMG {
padding: 10 10;
}
.m
```

```
{
background:
url(/img/wb.gif); text-align:
center;
padding: 1 1 1 1;
border: 1px dotted
ivory; width: 140;
}
-->
</STYLE>

<STYLE TYPE=«text/css»>
<!--
A {
text-decoration:none;
font-family: Arial;
font-size: 10pt; }
H4 { text-align:
center; color: #FF8040
} BODY {
background:
ivory; color:
#000000;
text-align: justify;
}
P {color: blue}
H1.class1 { text-align:
center; color: #FF8040 }
H1.class2 { text-align:
center; color: red;
border-width:
1; border: solid
}
-->
</STYLE>
</HEAD>
```

<BODY>

<script type=«text/javascript»><!--
google_ad_client = «pub-0369862403661526»;

google_ad_width = 120;
google_ad_height = 240;
google_ad_format =
«120x240_as»; google_ad_type =
«text_image»;
//2007-10-29: svoi120x240
google_ad_channel = «2020917188»;
//-->
</script>
<script type=«text/javascript»
src=«http://pagead2.googlesyndication.com/pagead/show_ads.js»
>
</script>
<script type=«text/javascript»>
var begun_auto_pad = 87823107;
var begun_block_id = 115576735;
</script>
<script
src=«http://autocontext.begun.ru/autocontext2.js»
type=«text/javascript»></script>
</TD><TD>
<H1 class=class1>КАСКАДНІ ТАБЛИЦІ СТИЛІВ</H1>
<H6 ALIGN=CENTER>Зверніть увагу.</H6>
<H4>Селектор CLASS</H4>

    Перш ніж розповідати про вживання селектора CLASS, слід сказати кілька загальних слів про вживання таблиць стилів, які, власне, пояснюють необхідність даного селектора.

    Справа у тому, що хоча застосування правил стилів в окремих тегах формально не суперечить правилам HTML, насправді є абсолютно недоцільним, оскільки зводить нанівець основні переваги таблиць стилів. Згадаймо, що таблиці стилів
:<UL>
<LI><FONT COLOR=«#0000FF»>a.</FONT> дають можливість поміщати всю інформацію про зовнішній вигляд сайта в одне міс-

це, наприклад, в одному файлі. Такий файл можна зробити один раз і потім користуватися ним усе життя.
</UL>

І, як наслідок –
<UL>
<LI><FONT COLOR=«#0000FF»>б.</FONT> дають можливість оформлювати сайт як єдине ціле, робити сторінки в єдиному стилі,
<LI><FONT COLOR=«#0000FF»>в.</FONT> при бажанні або необхідності зробити якісь зміни в зовнішньому вигляді сайта, досить внести виправлення тільки у файл з правилами таблиць стилів.
</UL>

    Іншими словами: таблиці стилів полегшують працю ВЕБ-майстра, видаляючи з нього рутинні дії по прописці параметрів кожного тега і їх значень та залишаючи більше часу і сил для творчості. Правда, я не знаю, чи компенсує цей залишок ті сили і час, які підуть на вивчення CSS.

    При використанні ж правил таблиць стилів в окремих тегах для внесення змін доведеться переглянути весь документ, що вимагатиме досить великої і кропіткої роботи.

    Але у кожної медалі дві сторони. Є друга сторона і в таблиці стилів:–) (щось жартів сьогодні багато!). Справа в тому, що таблиці стилів, селектором яких використовується тег HTML, впливають на відображення ВСІХ елементів у документі, визначеним цим тегом. Тобто, якщо ми пропишемо у файлі з правилами, наприклад: P {color: blue}
<P> То все, що буде братися в теги параграфа, буде відображатися синім кольором, як цей текст. < / P >
А якщо нам потрібно (або ми дуже хочемо) виділити якийсь параграф, для особливої важливості, наприклад червоним кольором? Добре якщо один, тоді можна прописати для нього правила в тезі, що його визначає. А якщо у нас таких параграфів на сайті багато? Наприклад 20–30 % або більше? На цей випадок у HTML 4.0 введений як стандарт для всіх тегів параметр CLASS. Його значенням є посилання на клас, що задається в таблиці стилів. Ім’я класу вказується в селекторі правила після імені тега і відокремлюється від нього крапкою.

Конкретний приклад: ми хочемо, щоб заголовки першого рівня на нашому сайті були двох видів. Прописуємо в таблиці стилів таке правило:
<FONT COLOR=«#0000FF»>H1.class1 { text-align: center;
color: #FF8040 }
H1.class2 { text-align:
center;
 color: red;
border-width: 1;
border: solid }
</FONT>і отримуємо:<H1 class=class1>Цей заголовок визначається класом 1</H1><H1 class=class2>Цей - клас 2</H1>
    Яскраві приклади, чи не так? У тексті документа ці заголовки визначено такими тегами:
<H1 class=class1> Цей заголовок визначається класом 1 </ H1>

<H1 class=class2> Цей – класом 2 </ H1>
У даному прикладі я обізвав класи просто по порядку. Це не має принципового значення. Ви можете присвоювати їм будь-які імена, наприклад, називати на честь улюблених друзів:
 H1.Sveta
 H1.Vova.
Головне потім не переплутати, хто є хто, і розставити імена кла- сів у тексті документа в потрібних місцях.
Ось власне і все про селектори, він же – параметр, CLASS. Мало, але досить суттєво.
</TD></TR></TABLE>
</BODY>
</HTML>

6. Зберегти документ з назвою CSS2.html, відкрити його за допомогою браузера та перейти до редагування HTML-коду.

7. Переведіть за допомогою перекладача російський текст або замініть своїм текстом із застосуванням попередньо заданих стилів.

8. Скопіювати у Блокнот такий текст.

<HTML>
<HEAD>
<TITLE> Створення HTML-сторінок з мультимедійними ефектами за допомогою фільтрів.</TITLE>

```
<META HTTP-EQUIV=«Content-Type» CONTENT=«text/html;
Charset=Windows-1251»>
<META NAME=«Keywords» CONTENT=««<>
<META NAME=«Description» CONTENT=«Різноманітні візуальні
ефекти: поступовий прояв зображення або тексту, зміна контрас-
тності графічного зображення, «свічення» букв тексту та ін. – Усе
це досягається застосуванням фільтрів до елементів сторінки й
організацією переходів з одного візуального стану в інший.»>
<STYLE TYPE=«text/css»>
<!--
BODY {
background:
ivory; color:
#000000;
text-align: justify;
margin-right: 20;
}
INPUT.20 {
width: 120;
height: 20;
font-family:
Arial; font-size:
9pt;
text-align: center;
background:
url(/img/wb.gif)
}
INPUT.40 {
width: 120;
height: 40;
font-family:
Arial; font-size:
9pt;
text-align: center;
background:
url(/img/wb.gif)
```

```
}
BUTTON.20 {
cursor:
hand; width:
120;
height: 20;
```

```
font-family:
Arial; font-size:
9pt; color: red;
text-align: center;
background:
url(/img/wb.gif)
}
BUTTON.40 {
cursor:
hand; width:
120;
height: 40;
font-family:
Arial; font-size:
9pt; color: red;
text-align: center;
background:
url(/img/wb.gif)
}
A {
text-decoration:none;
font-family: Arial;
font-size: 8pt; }
H4, H1 { text-align:
center; color: #004080 }
TABLE {text-align: justify;
 }
IMG
{
 padding: 10 10;
}
.m
{
background:
url(/img/wb.gif); text-align:
center;
```

```
padding: 1 1 1 1;
border: 1px dotted
ivory; width: 140;
}
-->
</STYLE>
```

</HEAD>
<BODY>

</TD><TD>

    Деякі автори виділяють фільтри в окремий клас інструментів оформлення HTML-сторінок і це правильно. Але, остільки, оскільки значення властивості фільтрів вказуються, дотримуючись правил задання властивостей каскадних таблиць стилів, а також враховуючи, що цей сайт – не всеосяжний довідник, а всього лише короткий посібник для початків- ців, я дозволив собі вільність включити опис фільтрів і переходів у главу CSS. Якщо в майбутньому я наважуся розвинути цю тему далі, вона буде виділена в окремий розділ.

Отже:

<H1> Фільтри . </ H1>

Як завжди, спочатку трохи теорії (уже ніяк без неї іноді не обійтися:–(. В принципі ті , хто не любить теорію, можуть відразу переходити до прикладів :–).

<B> <I> Фільтр </ I> </ B> – це певний алгоритм, що перетворює візуальне відображення елемента у вікні браузера. Він може бути статичним і динамічним.

<B> <I> Статичний фільтр </ I> </ B> перетворює елемент і після цього він відображається.

   <B><I>Динамічний фільтр </ I> </ B> впливає на візуальне відображення елемента, змінюючи його безпосередньо на HTML-сторінці, що приводить до ефекту анімації. Динамічний фільтр ще називають

<I> Переходом </ I> з одного стану відображення в інше.

Фільтри застосовуються не до всіх елементів, а тільки до тих, які визначають прямокутний блок і не є вікнами. <A Target=_blank HREF=«filtr1.shtml»> Тут </ A > Ви знайдете перелік елементів, до яких можуть застосовуватися фільтри і до яких вони застосовуватися не можуть, а <A target=_blank HREF=«filtr2.shtml»> тут </ A> – короткий перелік фільтрів і їх властивостей.

Застосовуються фільтри шляхом завдання властивості філь-

трів, дотримуючись правил завдання властивостей каскадних

таблиць стилів. Загальний вид запису завдання властивостей такий:
<B>filter: ім’я_фільтра(параметри);</B>

Параметри, якщо вони потрібні, задаються у формі:
<B> ім’я_параметра = значення_параметра </ B>

Деяким фільтрам потрібно кілька параметрів, що задаються через кому, а деяким параметри взагалі не потрібні, але круглі дужки повинні бути присутніми обов’язково.

Якщо до елемента застосовується кілька фільтрів, вони задаються у вигляді списку з пропуском як роздільник.

Ну, а тепер пора переходити до прикладів:–) Для експериментів нам знадобиться якась картинка. Візьмемо для цього хоча б троянди, які я минулого року намагався відправити (на жаль, безуспішно) жінкам-читачам моєї розсилки на восьме березня. Дорогі жінки, ось троянди, які призначалися Вам (краще пізно, ніж ніколи:–):

```
<TABLE ALIGN=CENTER CELLPADDING=1>
<TR>
<TD><IMG SRC=«img/roza.jpg» ALT=«roza.jpg[11 кб]»
WIDTH=«75» HEIGHT=«107» BORDER=«0»
ALIGN=«right»></TD>
</TR>
</TABLE>
```


А тепер почнемо з ними експериментувати. (Я не буду приводити код, за допомогою якого це робиться. Ви можете подивитися його клікнувши правою кнопкою миші і вибравши «перегляд у вигляді HTML».)

Ми можемо зробити їх трохи розмитими:

```
<IMG SRC=«img/roza.jpg» ALT=«приклад використання фільтру
blur()» WIDTH=«75» HEIGHT=«107» BORDER=«0» style=«filter:
blur()»> <BR>
```

Можемо зробити негатив:

```
<IMG SRC=«img/roza.jpg» ALT=«приклад використання фільтру
invert()» WIDTH=«75» HEIGHT=«107» BORDER=«0»
style=«filter: invert()»> <BR>
```

Або просто чорно-біле зображення:
<IMG SRC=«img/roza.jpg» ALT=«приклад використання фільтру gray» WIDTH=«75» HEIGHT=«107» BORDER=«0» style=«filter: gray»>
<P> < / P > така композиція зроблена з одного малюнка за допомогою фільтрів:
<TABLE ALIGN=CENTER BORDER=0 COLS=2> <TR>
<TD> <IMG SRC=«img/monarch ja timotei.jpg» ALT=«приклад використання фільтрів flipH і flipV» WIDTH=«300» HEIGHT=«160» BORDER=«0» style=«filter: flipH»> < / TD >
<TD> <IMG SRC=«img/monarch ja timotei.jpg» ALT=«приклад використання фільтрів flipH і flipV» WIDTH=«300» HEIGHT=«160» BORDER=«0»> </ TD>
< / TR > <TR>
<TD> < IMG SRC = « img / monarch ja timotei.jpg « ALT = «приклад використання фільтрів flipH и flipV» WIDTH=«300» HEIGHT=«160» BORDER=«0» style=«filter: flipV flipH»></TD>
<TD><IMG SRC=«img/monarch ja timotei.jpg» ALT=«приклад використання фільтрів flipH и flipV» WIDTH=«300» HEIGHT=«160» BORDER=«0» style=«filter: flipV»>
</TD>
</TR></TABLE>
    Остільки, оскільки в одній статті все одно не вдасться описати всі можливості, які фільтри дають ВЕБ-майстру, я наведу пару прикладів використання фільтра glow з тегом SPAN:
<SPAN STYLE=«width:100 %;color:blue; font-size:28pt; text-align: center; filter:glow(Color=orange);»>
<B>«Сяючі букви «</B></SPAN>
<SPAN STYLE=«width:100 %;color:blue; font-size:24pt; text-align: center; filter:glow(Color=red, Strength=10);»>
<B>«Сяючі букви «</B></SPAN>
<SPAN STYLE=«width:100 %;color:yellow; font-size: 24pt; text-align: center;
filter:glow();»>

```
<B>« Сяючі букви «</B></SPAN><BR>
</BODY>
</HTML>
```

9. Зберегти документ з назвою CSS3.html. Скопіюйте картинки у свою папку. Відкрийте його за допомогою браузера. Опишіть, які стилі були застосовані при відображенні картинок.

10. Скопіюйте у блокнот такий текст.

```
<HTML>
<HEAD>
<TITLE> Загальні властивості деяких фільтрів.</TITLE>
<META HTTP-EQUIV=«Content-Type» CONTENT=«text/html;
Charset=Windows-1251»>
<META NAME=«Keywords» CONTENT=««>
<META NAME=«Description» CONTENT=«Різноманітні візуальні
ефекти: поступовий прояв зображення або тексту, зміна контрастності графічного зображення, «свічення» букв тексту та ін. Усе це досягається застосуванням фільтрів до елементів сторінки й організацією переходів з одного візуального стану в інший.»>
<STYLE TYPE=«text/css»>
<!--
BODY
{
background:
ivory; color:
#000000;
text-align: justify;
margin-right: 20;
}
INPUT.20 {
width: 120;
height: 20;
font-family:
Arial; font-size:
9pt;
text-align: center;
```

```
background:
url(/img/wb.gif)
}
INPUT.40 {
```

```
width: 120;
height: 40;
font-family:
Arial; font-size:
9pt;
text-align: center;
background:
url(/img/wb.gif)
}
BUTTON.20 {
cursor:
hand; width:
120;
height: 20;
font-family:
Arial; font-size:
9pt; color: red;
text-align: center;
background:
url(/img/wb.gif)
}
BUTTON.40 {
cursor:
hand; width:
120;
height: 40;
font-family:
Arial; font-size:
9pt; color: red;
text-align: center;
background:
url(/img/wb.gif)
}
A {
text-decoration:none;
font-family: Arial;
```

```
font-size: 8pt; }
H4, H1 { text-align:
center; color: #004080 }
TABLE {text-align: justify;
 }
IMG
{
padding: 10 10;
```

```
}
.m
{
background:
url(/img/wb.gif); text-align:
center;
padding: 1 1 1 1;
border: 1px dotted
ivory; width: 140;
}
-->
</STYLE>
</HEAD>
<BODY>
<center>
<H1> Загальні властивості деяких фільтрів.</H1>
    Є декілька властивостей, які притаманні всім фільтрам або якійсь їх групі.
<UL>
<LI>    Кожен фільтр має властивість <B> <I> enabled </ I> </ B>, яка дозволяє (значення <B> <I> true </ I> </ B> ) або забороняє (значення <B> <I> alse </ I> </ B>) застосування приєднаного до об’єкта | фільтра
<LI> Властивість <B> <I> strenght </ I> </ B> фільтрів <B> <I> blur </ I> </ B> , <B> <I> glow < / I> </ B> і <B> <I> wave </ I> </ B> визначає інтенсивність застосування відповідного фільтра. Воно може приймати значення в межах від 0 до 255. Ось кілька прикладів (нагадую, що для того, щоб подивитися HTML-код, яким описаний той чи інший приклад, Вам потрібно клікнути по сторінці правою кнопкою миші і вибрати з меню «подивитися у вигляді HTML « . Маленька підказка з’явиться при простому наведенні покажчика миші на об’єкт.<BR>
<UL>
<LI><IMG SRC=«img/monarch ja timotei.jpg» ALT=«приклад використання фільтра blur ( strength = 10) «WIDTH = «300 «HEIGHT =«160 «BORDER =«0 « style =« filter: blur (strength = 10) «>
```

<IMG SRC = «img / monarch ja timotei.jpg «ALT = « приклад використання фільтрів flipH blur (strength = 255) «WIDTH = «300 «HEIGHT = «160» BORDER = «0» style = « filter: flipH blur (strength = 255) «>
<LI> < SPAN title = «приклад використання фільтра glow (Color = red, Strength = 2)» STYLE = «width: 100 %; color: blue; font – size: 24pt; text – align: center;
filter: glow (Color = red, Strength = 2); «>
<B> « Сяючі літери « </ B> </ SPAN >
<LI> <SPAN title = «приклад використання фільтра glow (Color = red , Strength = 200) «STYLE = « width: 100%; color: blue; font – size: 24pt; text – align: center;
filter: glow (Color = red, Strength = 200); «>
<B> «Сяючі літери « </ B> </ SPAN >
<LI> < SPAN title = « приклад використання фільтра wave (Strength = 2)» STYLE = «width: 100 %; color: blue; font – size: 34pt; text – align: center; filter: wave (Strength = 2); «>
<B> « Хвилясті літери « </ B> </ SPAN >
<LI> < SPAN title = «приклад використання фільтра wave (Strength = 20) « STYLE = «width: 100 %; color: blue; font – size: 34pt; text – align: center; filter: wave (Strength = 20); «>
<B> «Хвилясті літери « </ B> </ SPAN >

Як видно з останнього прикладу, зі значеннями іноді потрібно працювати досить акуратно:–).

</ UL>
<LI> Властивість <B> <I> direction </ I> </ B> фільтрів <B> <I> blur </ I> </ B> і <B> <I> shadow < / I> </ B> визначає напрям, в якому розмивається об’єкт або падає тінь. Значення цієї властивості може задаватися будь-яким числом, як позитивним, так і негативним і визначає кут, що відраховується від вертикалі об’єкта за годинниковою стрілкою з кроком 45 градусів.
<LI> Властивість <B> <I> color </ I> </ B> мають фільтри, які впливають на колірну гаму елемента. Це фільтри <B> <I> chroma </ I> </ B> , <B> <I> dropShadow </ I> </ B>, <B> <I> glow </ I> </ B>, <B> <I> mask </ I> </ B>,

<B> <I> Shadow </ I> </ B>. Значення цієї властивості задається шістнадцятковим числом виду # RRGGBB або словом, наприклад red, green та ін.
 Кілька прикладів:
<UL>
<LI> <SPAN title = «приклад використання фільтра dropShadow (color = green) «STYLE = «width: 100 %; color: blue; font – size: 34pt; text – align: center; filter: dropShadow (color = green); «>
<B> «Літери з тінню» </ B> </ SPAN >
<LI> <SPAN title = «приклад використання фільтра Shadow (color = red, Strength = 35, direction = 225) «STYLE = « width: 100 %; color: blue; font - size: 34pt; text – align: center; filter: Shadow (color = red, Strength = 35, direction = 225); «>
<B> «Літери з тінню» </ B> </ SPAN >
<LI> < IMG SRC = «img / monarch ja timotei.jpg «ALT = «приклад використання фільтра blur (strength = 10) «WIDTH = «300
«HEIGHT = «160» BORDER = «0» style = «filter: blur (strength = 50, direction = 45) «>
</ UL> <LI> Фільтри <B> <I> blur </ I> </ B> і <B> <I> wave </ I> </ B> мають булеву властивість < B > <I> add </ I> </ B> включати чи не включати вихідне зображення об’єкта у відфільтрований образ. Має значення true і false.
<SPAN Title=« приклад використання фільтра blur(Strength=20)» STYLE=«width:100 %;color:blue; font-size:44pt; text-align:center; filter: blur(Strength=20);»>
<B> Розмиті літери. </ B> </ SPAN >
< SPAN title = «приклад використання фільтра blur (Strength = 20, add = false) «STYLE = «width: 100 %; color: blue; font – size: 44pt; text – align: center; filter: blur (Strength = 20, add = false); «>
<B> Розмиті букви.</B></SPAN>
</UL>
</BODY>
</HTML>

11. Зберегти документ з назвою CSS4.html. Відкрийте його за допомогою браузера. Доповніть документ двома реченнями з використанням заданих стилів.

12. Застосуйте вивчені стилі у своїх сайтах і покажіть викладачу.

13. Випишіть стилі у звіт.

14. Оформіть звіт.

Запитання для самоперевірки

1. Навіщо використовуються стилі?
2. Як можна задати стилі?
3. Як визначити стилі для H1-H6?

ЛАБОРАТОРНА РОБОТА 13

Тема. Основи програмування PHP

Мета: навчитись писати і використовувати сценарії.

Хід роботи

1. Відкрийте папку Мій комп'ютер і уважно перегляньте, які є пристрої. Завантажте Start Denwer. Знову відкрийте папку Мій комп'ютер, там повинен з'явитись новий віртуальний диск. Зайдіть у папку www.

Наприклад, S:\home\localhost\www

У директорії www (це директорія, де зберігаються html документи сервера Apache) створіть файл index.html з будь-яким текстовим змістом. Тепер запустіть, браузер і наберіть: http://localhost/index.html.

Повинен завантажитися Ваш файл. Якщо все пройшло успішно, то сервер Apache працює коректно й Ви можете далі виконувати роботу.

2. Відкрийте текстовий редактор (або Dreamweaver) і набе- ріть такий код:

```
<html>
<head><title>Вставка коду PHP</title></head>
<body>
<h1>Приклад сторінки з PHP кодом</h1>
<?
print(«<h2>PHP – фрагмент</h2>»);
?>
</body>
</html>
```

Після цього виконаєте такі дії:

− збережете даний текстовий файл у каталозі localhost\www під іменем php_start1.php (зверніть увагу, розширення у файлі.php);

– запустите web-сервер Apache і в рядку адреси браузера наберіть http://localhost/. Ви повинні побачити усередині віртуального каталогу свій файл php_start.php;

– клацніть на ньому і, якщо ви правильно набрали наведений код, у вас повинна завантажитися сторінка (рисунок 13.1);

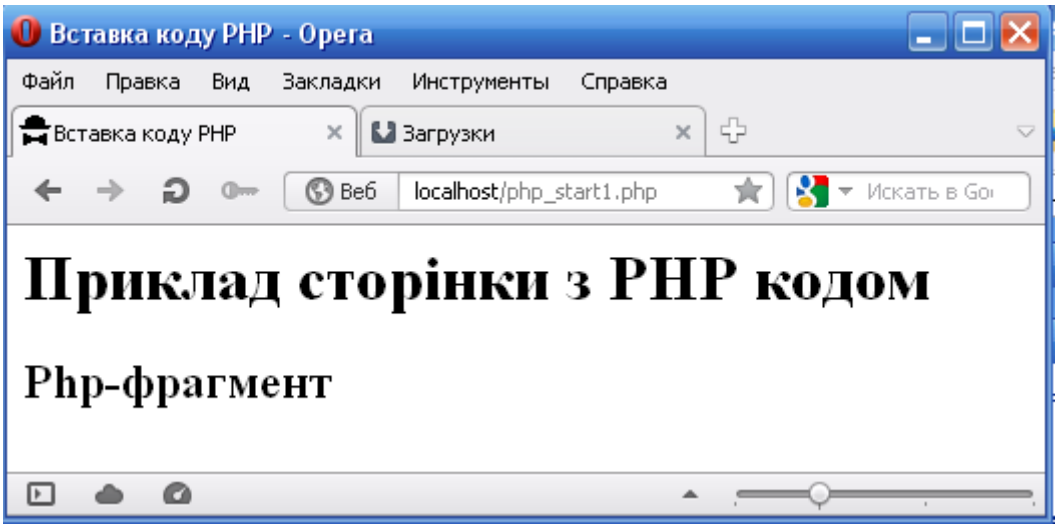


Рис. 13.1. Скріншот сторінки утримуючої Php-код

– перегляньте html-код даної сторінки (меню Вид – Перегляд HTML коду):

```
<html>
<head><title> Вставка коду PHP</title></head>
<body>
<h1>Приклад сторінки з PHP кодом</h1>
<h2> Php-фрагмент </h2>
</body>
</html>
```

Ми бачимо тільки звичайний html-код! Справа в тому, що сторінка має розширення php, тому web-сервер спочатку перед відправленням її клієнтові став переглядати вміст файла. Знайшовши фрагмент коду, який поміщений між символами <? … ?>, web-сервер відправив його інтерпретаторові PHP. Команда print(««); виводить на екран те, що перебуває між символами лапок. Причому браузер інтерпретує цей рядок як фрагмент html-коду. Саме тому строка «Php-фрагмент» виводиться як заголовок

другого рівня. І вже таку повністю згенеровану html-сторінку web-сервер передав клієнтові.

Отже, для того щоб сторінка була динамічною, потрібно, щоб виконалися дві умови:

- вона повинна мати розширення php;
- усередині неї повинен перебувати php-код, укладений між символами <? … ?>, команди повинні відділятися символом «крапка з комою».

Ми можемо створити файл, який містить тільки php-код:

```
<?
print(«<html><head><title>Вставка коду
PHP</title></head></body>»);
print(«<h1> Приклад сторінки з PHP кодом </h1>»);
print(«<h2>PHP- фрагмент</h2>»);
?>
```

Збережіть файл у каталозі S:\home\localhost\www під іменем php_start2.php. Запросивши цей файл у web-сервера, ви побачите ту саму сторінку, що й на рисунку 13.1.

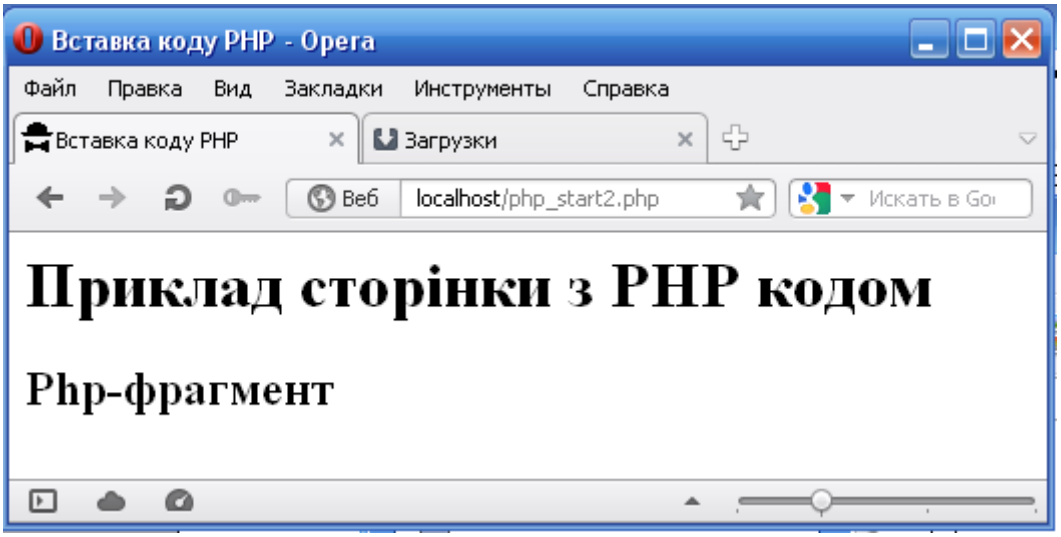


Рис. 13.2

3. PHP – це скрипт-мова, що вбудовується в HTML, який інтерпретується і виконується на сервері. Створіть файл test.php з таким змістом:

```
<html>
<head>
```

```
<title> Вчимо PHP</title>
</head>
<body>
<?php
echo «Привіт, PHP працює!<br>\n»;
phpinfo();
?>
</body>
</html>
```

4. Як і в будь-якій мові, програмування PHP уміє пристосуватися зі змінними. Їх використовувати дуже просто, досить задати її ім’я і сказати, що міститься у даній змінній. Ім’я змінної обов’язково повинне починатися зі знака долара, повідомляти тип змінної не потрібно. Значення змінної, узяте в лапки, інтерпретується як текст, числове значення без лапок – як число:

```
<?
 $a = "текстові";
 $b = "змінні";
 $c = $a." ".$b;
 print("<p>$c</p>");
 $d = 3;
 $e = 5;
 $f = $d + $e;
 echo $f;
 ?>
```

У даному прикладі ми створили дві текстові змінні й дві чис- лові. З’єднали текстові змінні (виконали конкатенацію) і викона- ли операцію додавання із числовими змінними. Результати виве- ли на екран за допомогою команди print().

Вставте даний фрагмент в html-код, збережіть його в каталозі S:\home\localhost\www під іменем php_start3.php. Відкривши в браузері даний файл, побачите два рядки «текстові змінні» і «8» (рис. 13.3).

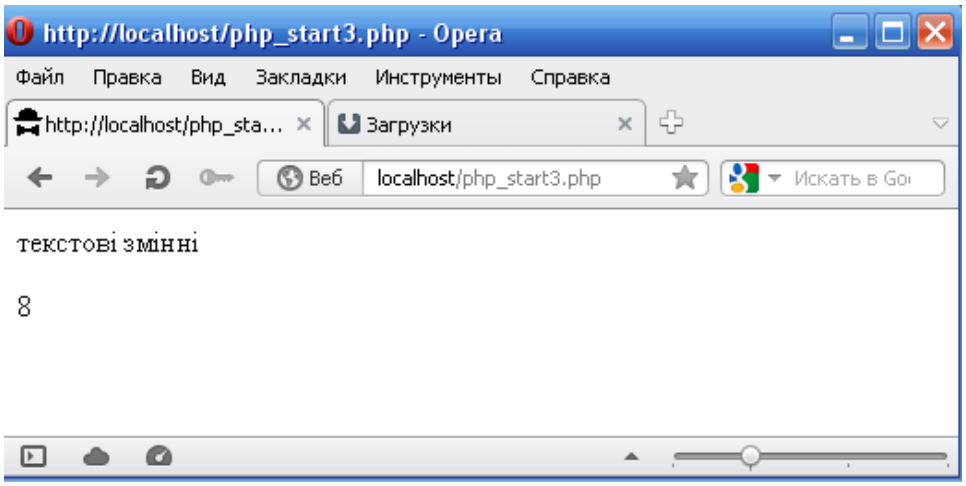


Рис. 13.3. Створення двох текстових змінних і двох числових

Зверніть увагу, команда print() вивела не рядок $c, а значення змінною с. Саме для легкості виявлення змінної усередині будь-якої конструкції її ім'я повинно обов'язково починатися зі знака долара.

PHP підтримує більше ніж два типи змінних, але для наших проектів числових і строкових буде досить.

Увага! PHP – мова, яка розрізняє рядкові й прописні букви, $user_name і $User_name – це різні змінні.

5. А як зробити так, щоб людина самостійно вводила значення змінних? Тут нам на допомогу приходять форми.

Давайте згадаємо, що кожний об'єкт форми має ім'я (параметр name=««). По суті, коли людина друкує текст у поле введення або ставить хрестик у поле типу checkbox, він задає значення змінної. Залишилося лише навчитися їх передавати php-коду!

Створіть новий html-файл, що містить такий код:

```
<html>
 <head>
<title>уведення значень у форму</title>
</head>
<body>
<form action="age.php" method="get">
<p>ваше ім'я: <input name="user_name" size="20" type="text"></p>
<p>рік народження: <input name="user_yare" size="20" type="text"></p>
<input  name="b1" type="submit" value="відправити"><input  name="b2" type="reset" value="очистити">
</form>
</body>
</html>
```

Тут створюємо форму, що включає чотири об’єкти: два текстові поля й дві кнопки. Кожний об’єкт має своє унікальне ім’я. Сама форма має два параметри. Параметр action, що повідомляє, якому файлу будуть передаватися дані після натискання кнопки submit, це – файл age.php.

Параметр method має значення get, змінні з форми будуть передаватися у відкритому вигляді, приєднуючись до адреси. Припустимо, ми введемо ім’я користувача Nata, а рік його народження 1980, то нажавши кнопку «відправити», дані будуть передані файлу age.php, і рядок адреси прийме вигляд (рис. 13.3):
http://localhost/age.php?user_name=Nata&user_yare=1980&b1=%E2%B3%E4%EF%F0%E0%E2%E8%F2%E8

і тут потрібні деякі коментарі:

усі передані дані розташовуються за символом «?»;

усі дані зібрані у вигляді: ім’я змінної = значення змінної;

змінна b1 має значення «відправити» текст, що містить кирилицю. Для таких змінних браузер автоматично виконує Url-кодування.

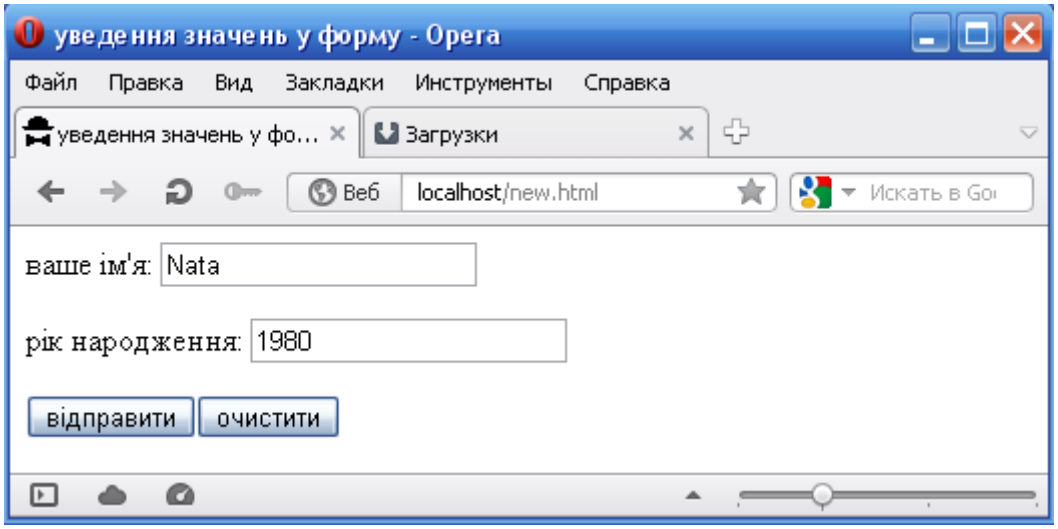


Рис. 13.4. Форма, що включає чотири об’єкти

Створіть файл age.php і наберіть у ньому код.

```
<html>
<head>
<title>обчислення віку</title>
</head>
<body>
<?
  $name =  $_GET['user_name'];
  $yare = $_GET['user_yare'];
?>
<p> Ласкаво просимо <? echo ($name) ?></p>
<?
  $y =  date("Y");
  $user_age =  $y - $yare;
  print  ("<p>вам  $user_age років</p>");
?>
</body>
</html>
```

Файл age.php буде відкритий у той момент, коли користувач натисне кнопку «відправити». Він одержить значення двох змінних user_name і user_yare, які буде використовувати в php-коді (рис. 13.5).

Тому що змінні, отримані методом get, спочатку обов'язково необхідно витягти з отриманого масиву даних. Саме це робить рядок:

$user_name = $_GET[‘user_name’];

Якщо дані передаються методом post, то відповідно потрібно витягати змінні з відповідного масиву.

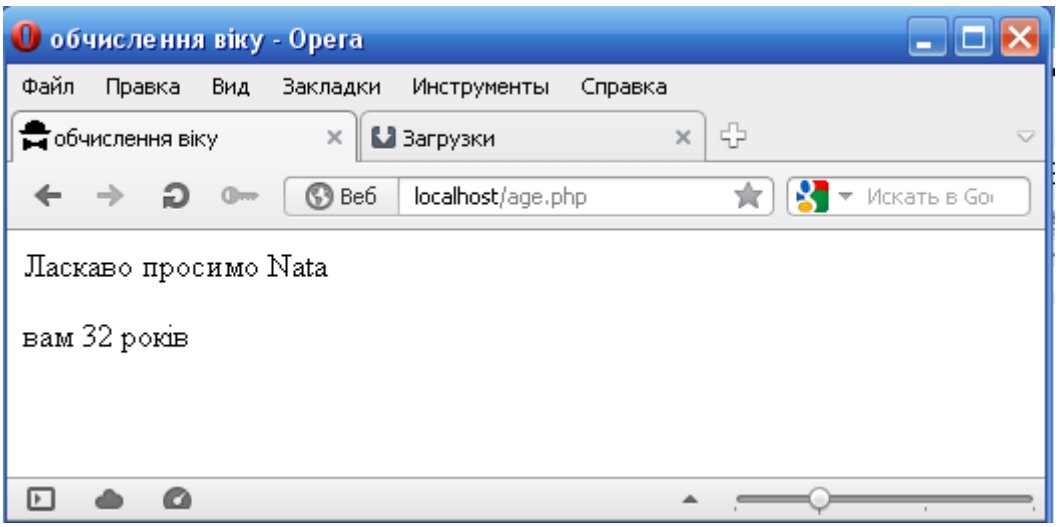


Рис. 13.5. Файл age.php

У наведеному фрагменті використовується кілька нових функцій:

echo(); – функція, відповідальна за виведення значень змінних, багато в чому аналогічна функції print().

date(); – функція, що повертає поточну дату у вигляді рядка. Функція має велику кількість параметрів, що показують, у якому вигляді повинен бути представлений результуючий рядок. У нашому лістингу функція date() має єдиний параметр «Y», завдяки ньому в змінній $y буде поточний рік.

Інші параметри функції date():

G – година, 24-часовий формат без провідних нулів; тобто від «0» до «23»;

i – хвилини; тобто від «00» до «59»;

j – день (число) місяця без провідних нулів; тобто від «1» до «31»;

m – місяць; тобто від «01» до «12»;

H – година, 24-часовий формат; тобто від «00» до «23»;

n – місяць без провідних нулів; тобто від «1» до «12»;

s – секунди; тобто від «00» до «59»;

Y – рік, 4 цифри; наприклад, «1999»;

y – рік, 2 цифри; наприклад, «99»;

z – день року; тобто від «0» до «365».

Приклад використання функції: date()

$today = date(«j, n, Y»); змінна $today прийме значення: 10, 3, 2001 (число, місяць, рік).

$today = date(«H:i:s»); змінна $today прийме значення: 17:16:54 (годинник, хвилини, секунди).

Незважаючи на те, що змінна $yare – строкового типу, ми можемо не перетворювати рядок у число, а одразу відняти поточний рік від року народження, уведеного користувачем. PHP це зробить перетворення самостійно.

Додавання коментарів

Гарний стиль програмування вимагає вставки коментарів до свого коду. Ви можете додати коментарі в PHP двома способами:

- Для коментаря одного рядка вам потрібно на початку до- дати символи //.
- Для коментування багаторядкового блоку необхідно заключити блок символами /* ... */.

Остерігайтеся вкладених коментарів. Такі конструкції: /* /* ... */ */ викличуть проблеми.

6. У цьому прикладі показано, як в PHP легко обробляти дані з html–форм.

Створіть простий html-файл:

```
<html>
<head>
<title>Інформаційний запит</title>
<body>
<center>
Прагнете більше знати про наші товари?
<p>
<table width=400><TR><TD align=right>
<form action="email.php" method="POST">
Ваше ім'я:<br>
<input type="text" name="name" size="20" maxlength="30">
<p>
Ваш email:<br>
<input type="text" name="email" size="20" maxlength="30">
<p>
Мене цікавлять:
<select name="preference">
<option value="Яблука">Яблука
<option value="Апельсини">Апельсини
</select>
<p>
<input type="submit" value="Відправити запит!">
</form>
</td></tr></table></center>
</body>
</html>
```

Назвіть цей файл request.html. У ньому ви вказали, що дані форми будуть оброблятися файлом email.php. Приведемо його зміст:

```
<?php
/* Цей скрипт отримує змінні із request.html */
echo"<center>";
echo"Привіт,  ".$_POST['name'];
echo "<br><br>";
echo "Дякую за Ваш інтерес.<br><br>";
echo "Вас цікавлять".$_POST ['preference'].",
Інформацію про них ми надішлем вам на email: ".$_POST['email'];
echo "</center>";
?>
```

Тепер, якщо користувач викличе request.html і набере у формі «Володимир», email: vladymyr@yahoo.com і вкаже, що його цікавлять «Яблука», то у відповідь з'явиться email.php, який виведе приблизно наступне:

Привіт, Володимир.
Дякую за Ваш
інтерес.
Вас цікавлять Яблука. Інформацію про них ми надішлемо вам на email: vladymyr@yahoo.com

7. Тепер ми повинні дотримати обіцянки й вислати email. Для цього в PHP є функція MAIL.

Синтаксис: void mail(string to, string subject, string message, string add_headers);

to – email адреса одержувача.

subject – тема листа.

message –текст повідомлення.

add_headers – інші параметри заголовка листа (необов'язковий параметр).

Допишемо в кінець файла email.php такий код:

```
<?php
mail($email, «Запит на інформацію», «$name\n
Спасибі за ваш інтерес!\n
Вас цікавлять $preference\n
Ми їх поширюємо безкоштовно. Звернетеся в найближчу філію нашої компанії і отримаєте ящик цього продукту.\n
«);
mail(«administration@me.com»,
```

«Був запит на інформацію.»,

«$name цікавили $preference\n email-
адреса: $email. \n»);
?>

Тепер користувач буде отримувати листа з більш докладною інформацією про наші товари. Такий лист отримає і адміністратор сайта.

8. Мережа Internet побудована на язикових контактах. Користувачі мають справу з інформацією і рядками. Рядки є базовими типами даних PHP. Тому можна вести пошук підстрок, зіставляти рядка й символи і безліч інших операцій.

```
<?php $strarray=explode(":","Понеділок:Вівторок:Середа:Четвер:П'ятниця:Субота:Неділя");?>
<select name="dayofweek" size="1">
<?php for($i=0;$i<7;$i++){ ?>
<option><?php echo($strarray[$i]);?></option>
<?php } ?>
</select>
```

9. Робота з масивами

Спочатку визначаємо два масиви. Один для українських імен місяців, інші для назв днів тижня. При цьому не забуваємо про нульове значення масивів і залишаємо його порожнім. Потім зчитуємо функцією PHP номер поточного місяця, день тижня і далі працюємо з масивом.

У перших витягаємо з відповідної комірки масиву день тижня. А з другого масиву вибираємо ім’я місяця. Залишилося тільки вивести результат на екран.

```
<?php
 // ------- визначаємо масив для місяців ----------
 $q[]="";
 $q[]="січня";
$q[]="лютого";
$q[]="березня";
$q[]="квітня";
$q[]="травня";
 $q[]="червня";
$q[]="липня";
$q[]="серпня";
$q[]="вересня";
$q[]="жовтня";
$q[]="листопада";
 $q[]="грудня";
 // ------- визначаємо масив для днів тижня ----------
 $e[]="неділя";
$e[]="понеділок";
$e[]="вівторок";
$e[]="середа";
$e[]="четвер";
 $e[]="п'ятниця";
$e[]="субота";
$m=date('m'); // ---- зчитуємо місяць
 if ($m=="01") $m=1;
if ($m=="02") $m=2;
 if ($m=="03") $m=3;
 if ($m=="04") $m=4;
if ($m=="05") $m=5;
 if ($m=="06") $m=6;
 if ($m=="07") $m=7;
 if ($m=="08") $m=8;
if ($m=="09") $m=9;
$we=date('w'); // ---- зчитуємо день тижня
 $chislo=date('d'); // ---- зчитуємо число
 $den_nedeli = $e[$we]; // ---- витягаємо з масиву відповідне значення дня тижня
 $mesyac = $q[$m]; // ---- витягаємо з масиву відповідне значення місяця
 echo "Сьогодні ".$chislo." ".$mesyac.", ".$den_nedeli;
 ?>
```

Практичне завдання

1. Створіть два файли. Помістіть їх у каталог, запустіть сер- вер і відкрийте файл html. Введіть у поля ваше ім’я і рік вашого народження, натисніть кнопку відправити, у вас повинен відкри- тися файл, який одержить дані з першого файла, використає їх і видасть результат.

2. Самостійно створіть форму, у якій уводиться ім’я студен- та і його рік народження. Дані з форми повинні передаватися у php-файл, який визначає, на якому курсі вчиться людина (припу- скаючи, що студент вступив до університету в 16 років).

Питання для самоконтролю

1. Які типи змінних підтримує мова PHP?
2. У чому відмінність php-сторінки і html-сторінки?
3. Як передати змінну в php-сторінку?
4. Які параметри існують у функції data()?
5. Що повертає web-сервер при запиті php-сторінки?

ЛАБОРАТОРНА РОБОТА 14

Тема. Використання констант у сценаріях PHP

Мета: розібратися, як створюються константи та відображаються їх значення.

Хід роботи

Створіть PHP-сторінку найпростішими засобами. Почніть зі створення простої PHP-сторінки:

```
<html>
<title>Workflow Registration</title>
<body>
  <p>You entered:</p>
  <p><?php echo "Some Data"; ?></p>
</body>
</html>
```

У нашому випадку є тільки одна команда, echo, це команда серверу вивести текст в лапках. Тобто, якщо ви збережете цю сторінку і захочете переглянути її своїм браузером, браузер відобразить такий текст:

```
<html>
<title>Workflow Registration</title>
<body>
  <p>You entered:</p>
  <p>Some Data</p>
</body>
</html>
```

Збережіть текст PHP-сторінки у файл з іменем registration_ action.php .

Відкрийте браузер з адресою http://localhost/registration_ action.php. Результат роботи вашого браузера буде подібний тому, що ви можете бачити нижче на рис. 14.1.

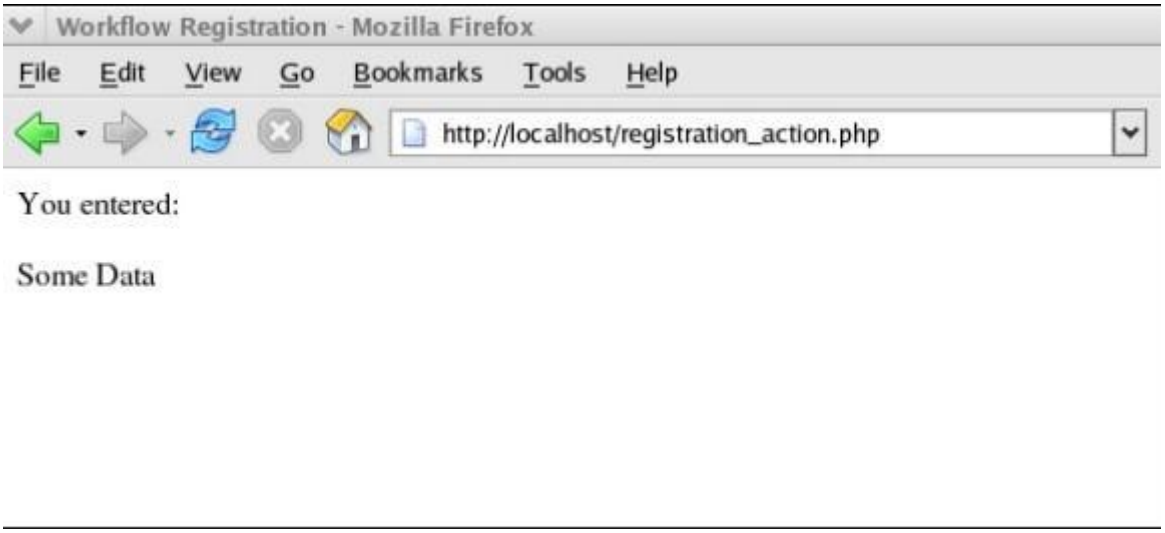


Рис. 14.1. Результат роботи команди echo

Таким чином, ви написали сторінку з використанням PHP.

Змінна служить контейнером й іменем для даних. З того моменту, як ви привласнили змінній деяке значення, PHP завжди, зустрівши цю змінну, замінить її на приписане значення. Розглянемо приклад, для цього внесіть такі зміни у свою сторінку:

```
<html>
<title>Workflow Registration</title>
<body>
   <p>You entered:</p>

<?php
   $username = "tyler";
   $password = "mypassword";

   echo "<p>Username = " . $username . "</p>";
   echo "<p>Password = " . $password . "</p>";
?>

</body>
</html>
```

Збережіть файл і поновіть сторінку вашого браузера. Результат виконання на рис. 14.2.

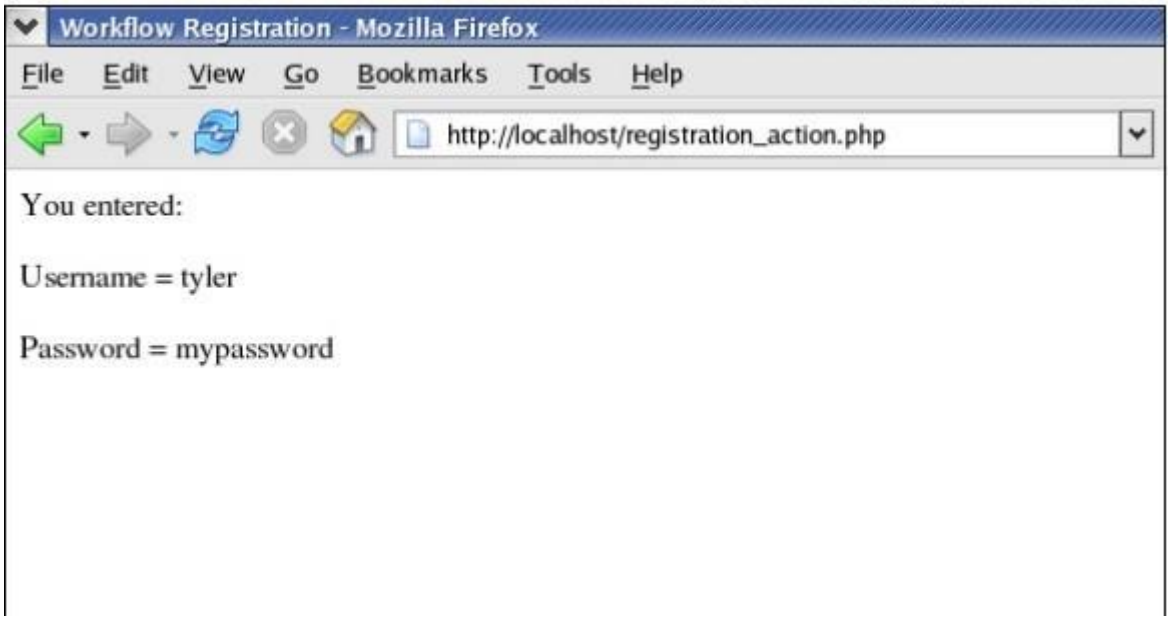


Рис. 14.2. Сторінка браузера після відновлення

Для об'єднання або конкатенації двох текстових елементів використовується крапка. Ви можете об'єднати в такий спосіб будь-яке число рядків або текстових елементів.

Константи. Ви можете вільно змінювати значення змінної, але іноді буває корисно задати змінну, значення яке не може бути змінене. Такі об'єкти називаються константами. Наприклад, можете визначити константу для зберігання заголовка, який буде виводитися на кожній сторінці:

```
<?php
define(«PAGE_TITLE», «Workflow Registration»);
?>

<html>
<title><?php echo PAGE_TITLE ?></title>
<body>
<p>You entered:</p>
...
```

Цей приклад може здатися тривіальним, але надалі ви переконаєтеся, що таке визначення корисне, коли йдеться про декілька сторінок.

Зверніть увагу, що в цьому операторі задається пара: ім'я константи і її значення. Якщо ви спробуєте змінити значення

константи після того, як воно було визначено, то отримаєте повідомлення про помилку.

При посиланні на константу в елементі title ми не використовували знак долара, як це робиться перед іменами змінних. Ви можете привласнювати константам довільні імена, але, згідно із прийнятими правилами, імена констант складаються із прописних букв.

Дотепер використовували команду **echo** для відображення інформації, але це досить громіздка конструкція. У тих випадках, коли вивести потрібно один елемент, існує більш простий спосіб.

Для виводу даних у PHP існує оператор виводу і ви можете передати дані за допомогою конструкції <?= ?> :

```
<?php
define(«PAGE_TITLE», «Workflow Registration»);
?>
<html>
<title><?= PAGE_TITLE ?></title>
<body>
<p>You entered:</p>
...
```

Зверніть увагу, що за оператором виводу не стоїть крапка з комою.

Створення й використання форм у PHP

Мова PHP споконвічно розроблялася як мова для Web-програмування. Звичайно, ви можете запустити PHP з командного рядка, але фактично випадки використання PHP поза Web-додатків дуже рідкі. Як правило, PHP використовується разом з HTML-формами.

Ви створюєте форму з використанням HTML, потім користувач вводить дані в цю форму й ініціює її передачу, браузер пересилає дані серверу у формі масиву.

Створення форм у HTML

Почнемо зі створення сторінки реєстрації для вашого додатка. На цій сторінці користувачі будуть уводити свої дані, а ви будете затверджувати їх, тобто виконувати деякі перевірки, перш ніж переслати дані в базу. Для початку просто створимо форму для введення. Створимо новий файл registration.php і помістимо в нього такий текст:

```
<html>
<head><title>Workflow System</title></head>
<body>
<h1>Register for an Account:</h1>
<form action="registration_action.php" method="GET">

Username: <input type="text" name="name" /><br />
Email: <input type="text" name="email" /><br />
Password: <input type="password" name="pword" /><br />
<input type="submit" value="GO" />
</form>

</body>
</html>
```

Таким чином, ми маємо просту форму, яка втримується усередині HTML-тегу form, у ній є текстове поле для введення пароля і клавіша для відсилання даних. Збережіть цей текст у файл із іменем registration.php і помістіть файл у кореневу папку з документами для вашого сервера, потім відкрийте браузер з адресою http://localhost/registration.php. Результат роботи вашого браузера буде схожий на те, що можна бачити нижче, на рис. 14.3.

Рис. 14.3. Форма для реєстрації

Зверніть увагу на те, що в полі для введення пароля не відображаються символи, які ви набираєте, замість цього з’являються зірочки. Що ж відбувається, коли користувач натискає клавішу GO?

Передача даних з форми

Розглянемо докладніше формат елемента form нашої форми<form action=«registration_action.php» method=«GET»>

У цьому елементі визначено два параметри. Перший, action, повідомляє браузер, куди посилати інформацію. У нашому випадку посилання веде до того файлу, який ми створили раніше, registration_action.php. Другий параметр, method, повідомляє браузер, як передавати дані.

Подивимося, як це працює. Введіть які-небудь дані й натисніть кнопку GO. Ви побачите щось схоже на рис. 14.4.

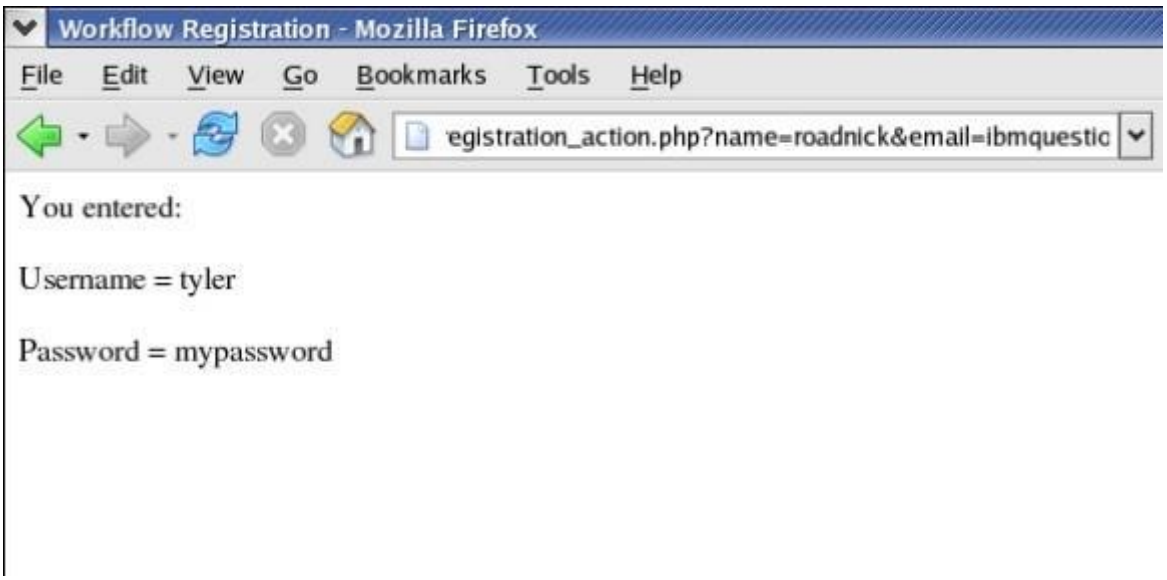


Рис. 14.4. Висновок даних, уведених у форму

У тому випадку, якщо ви не ввели дані, але нажали кнопку, форма буде працювати також, оскільки в нас ще не налагоджене приймання введених даних. Зверніть увагу на URL, який з’явився у полі адреси браузера.http://localhost/registration_action.php?name=roadnick&email=ibmquestions%40nicholaschase.com&pword=supersecretpassword

Зверніть увагу, що кожному елементу форми в URL відповідає пара ім’я=значення і ці пари розділені амперсандами. Такий

вид URL обумовлений тим, що ми використовували як метод пе- редачі GET. Далі ми розглянемо використання методу POST.

Приймання даних з форми

Розглянемо, яким чином дані, введені в HTML-форму, можна зробити доступними для обробки PHP-сторінки. Внесіть такі зміни у файл registration_action.php:...

```
<body>
  <p>You entered:</p>

<?php
  $username = $_GET['name'];
  $password = $_GET['pword'];

  echo "<p>Username = " . $username . "</p>";
  echo "<p>Password = " . $password . "</p>";
?>

</body>
</html>
```

Значення змінних будуть тепер вибиратися за іменем з масиву $_GET. Трохи нижче поговоримо про масиви докладніше, а зараз, спробуйте обновити віконце браузера, ви побачите, що введені нами дані з'явилися на своїх місцях, як показано на рис. 14.5.

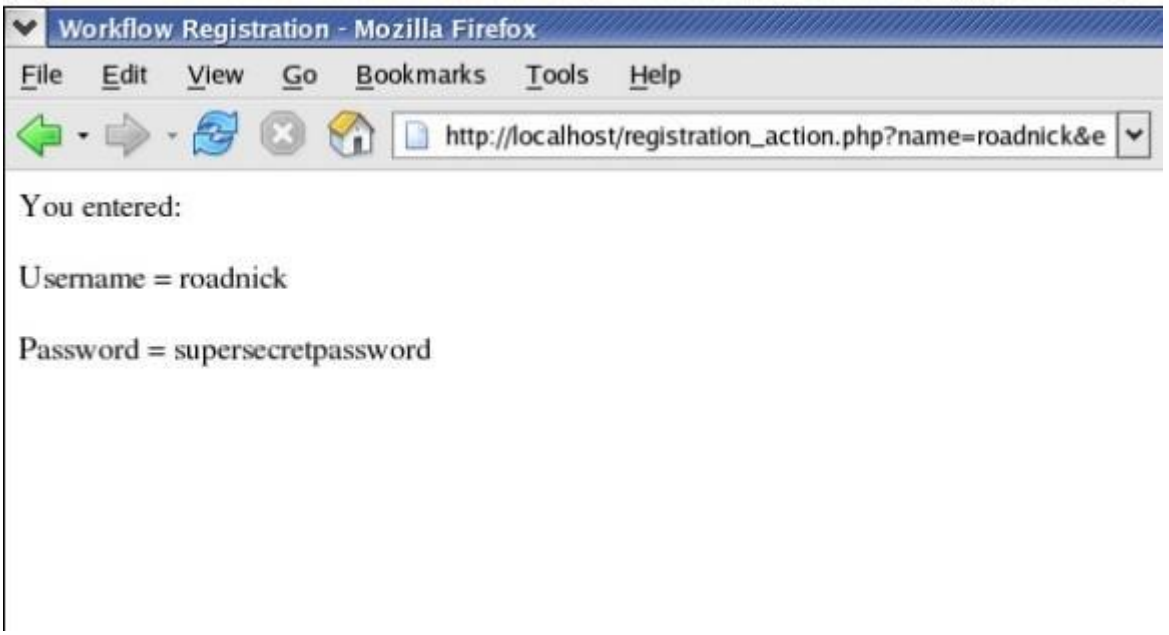


Рис. 14.5. Введені дані відображаються у браузері

Ви можете витягти будь-яку частину переданої інформації з імені, але є і інші можливості роботи з масивами.

Масиви

У PHP можна організовувати масиви даних, тобто списки значень, це дозволяє легко переміщати одночасно цілі групи даних. Наприклад, створимо масив із трьох елементів і передамо його для виводу на сторінку:

```
$formnames = array(«name», «email»,
«pword»); echo «0=«.$formnames[0].»<br />«;
echo «1=«.$formnames[1].»<br
/>«; echo
«2=«.$formnames[2].»<br />«;
```

Функція **array()** повертає змінну, яка є масивом. (Про цю функцію будемо говорити пізніше, поки досить розуміти, що функцію можна викликати і що вона повертає значення того або іншого типу, яке ви приписуєте змінній).

Результат роботи цього скрипта буде таким:

```
0=name<br />
1=email<br  />
2=pword<br
/>
```

Зверніть увагу, що перше значення у масиві має індекс, який дорівнює 0, але не 1. Помітьте також, що виділяємо потрібне значення, додавши індекс у квадратних дужках до імені масиву. Подібним чином ми одержували доступ до значень змінної форми. Змінна $_GET є іменем масиву спеціального типу, асоціативного масиву, до елементів якого звертаються не по індексах, а по ключах.

У той момент, коли ви відсилаєте свою форму, ви, по суті, створюєте асоціативний масив:

```
$_GET = array(«name» => «roadnick»,
«email» => «ibmquestions@nicholaschase.com»,
«pword» => «supersecretpassword»);
```

Саме цей факт дозволяє згодом виділити окремі значення,

наприклад, $_GET[«name»], імена змінних у цьому випадку відіграють роль ключів.

Одержання інформації про масив

Асоціативні масиви корисні при передачі даних між формами, але при їхньому використанні можуть виникнути труднощі, далеко не завжди відома структура масиву. Така ситуація може бути, наприклад, якщо асоціативний масив був отриманий у результаті виконання запиту до бази даних.

У PHP є дві функції, які полегшують використання таких масивів:

```
<body>
  <p>You entered:</p>

<?php
  $form_names = array_keys($_GET);
  $form_values = array_values($_GET);

  echo "<p>". $form_names[0] . " = " . $form_values[0] . "</p>";
  echo "<p>". $form_names[1] . " = " . $form_values[1] . "</p>";
  echo "<p>". $form_names[2] . " = " . $form_values[2] . "</p>";
?>

</body>
</html>
```

Функції array_keys() і array_values() повертають звичайні масиви, до елементів до яких можна звертатися за допомогою числових індексів, як показано нижче на рисунку 14.6.

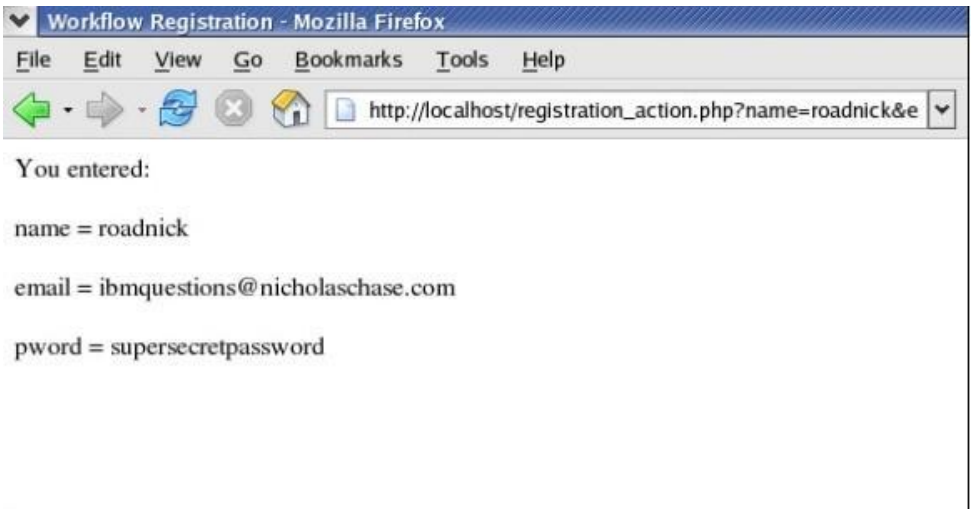


Рис. 14.6. Витяг даних з масиву з використанням числових індексів

Цей спосіб роботи з масивами також має незручності. Наприклад, ви можете не знати, скільки всього елементів у асоційованому масиві. У PHP існують інші способи обігу з асоційованими масивами. Наступним кроком ми розглянемо два такі способи.

Використання циклу for-next

Завдання послідовної обробки деякого числа однотипних значень виникає досить часто. Для її виконання застосовується така конструкція, як цикл. Наприклад, один із прикладів циклу:

```
for ($i = 0; $i < 10; $i++) {
echo $i. « «;
}
```

Результат його роботи буде таким: 0 1 2 3 4 5 6 7 8 9.

Розглянемо синтаксис вираження for. Спочатку PHP присвоює значення 0 змінної $i. Цикл триває доти, доки значення змінної $i залишається менше 10, на кожному кроці циклу значення змінної $i збільшується на одиницю.

Оскільки в нас є можливість довідатися, скільки всього значень містить масив $_GET, ми легко можемо організувати цикл, який відобразить усі значення, передані нам з форми:

```
<body>
  <p>You entered:</p>

<?php
  $form_names = array_keys($_GET);
  $form_values = array_values($_GET);

  for ($i = 0; $i < sizeof($_GET); $i++) {
    echo "<p>".$form_names[$i]." = " . $form_values[$i] . "</p>";
  }
?>

</body>
</html>
```

Функція sizeof() повертає число елементів масиву, у нашому випадку масиву $_GET. Ця функція зупиняє виконання циклу на

останньому елементі масиву, результат роботи циклу показаний на рис. 14.7.

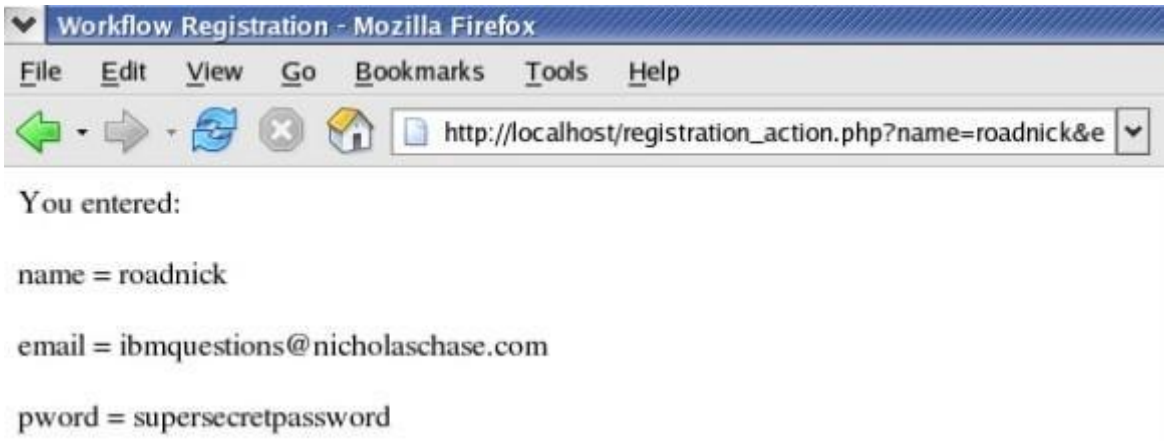


Рис. 14.7. Використання функції sizeof для зупинки циклу

Для роботи з асоційованим масивом $_GET можете використовувати й інший тип циклу, цикл із ключовим словом foreach.

Використання циклу foreach

Асоційовані масиви настільки часто використовуються у PHP, що для роботи з ними була створена спеціальна конструкція, яка дозволяє одержувати доступ до даних, минаючи ключі. Замість цього можете використовувати цикл foreach, він дозволяє маніпулювати з даними масиву прямо. Давайте розглянемо приклад коду:...

```
<?php
foreach ($_GET as $value) {
echo «<p>« . $value . «</p>«;
}
?>
```

На першому кроці циклу буде відображено перше значення з масиву $_GET, це значення буде приписано змінній $value, потім це значення виводиться командою echo. Потім цикл повторюється і змінній $value приписується друге значення і так далі, цикл працює доти, поки в масиві $_GET ще залишаються неопрацьовані елементи. Результат роботи цього циклу буде таким:

```
<p>roadnick</p>
```

ibmquestions@nicholaschase.com
supersecretpassword

Ця конструкція дозволяє витягати не тільки значення елементів асоційованого масиву, але й ключі:...

```
<?php
foreach ($_GET as $key=>$value) {
echo «<p>«.$key.» = « . $value . «</p>«;
}
?<
```

...

Такий варіант коду приведе нас до результату, який ми вже бачили вище:

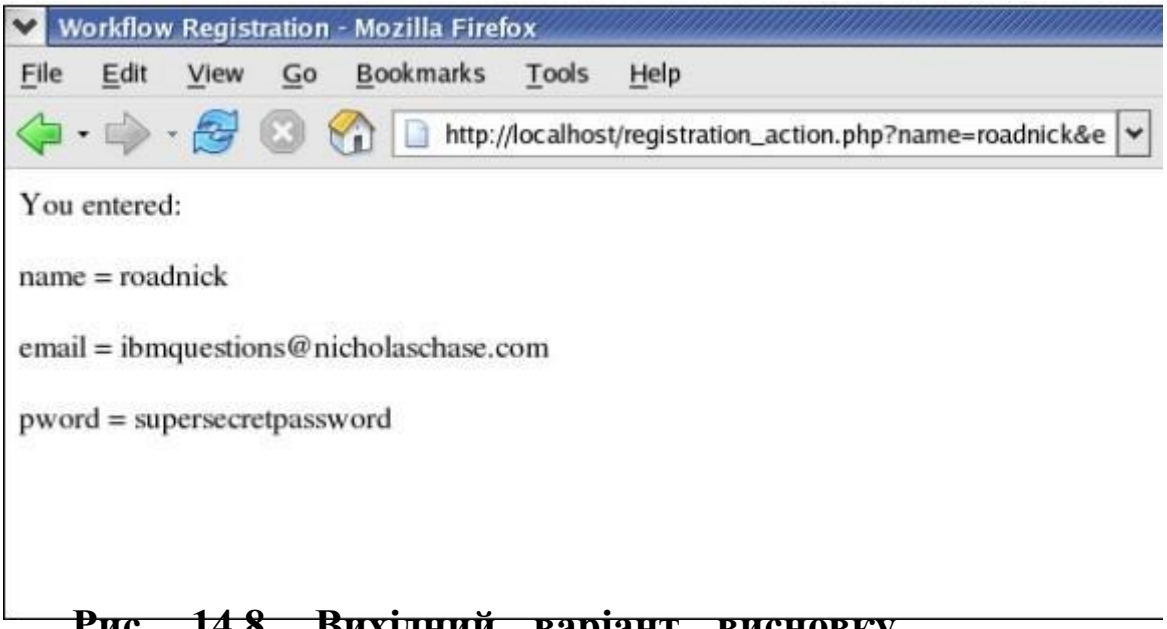


Рис. 14.8. Вихідний варіант висновку

Повторювані значення у формах

Іноді виникає ситуація, коли у формі повинно бути введено кілька значень для змінної з тим самим іменем. Прикладом може служити поле пароля pword. Оскільки користувачі не можуть бачити символи пароля, які вони набирають, то ви можете попросити їх ввести пароль двічі, для того щоб переконатися, що вони не зробили помилку при введенні: ...

```
Username: <input type="text" name="name" /><br />
Email: <input type="text" name="email" /><br />
Password: <input type="password" name="pword[]" /><br />
Password (again): <input type="password" name="pword[]" /><br />
<input type="submit" value="GO" />
...
```

Зверніть увагу, що ім'я поля pword трохи змінилося. Оскільки це поле повинно містити кілька значень, то воно саме стало масивом. Таким чином, масив переданих даних для цієї форми буде містити як один зі своїх елементів інший масив. Коли на- тиснете кнопку відправлення форми, то в полі адреси виникне наступний URL:http://localhost/registration_action.php?name=roadnick&email=ibmquestions%40nicholas chase.com&pword[]=supersecretpassword&pword[]=supersecretpassword

При цьому в процесі обробки буде створений масив:

$passwords = array(«supersecretpassword», «supersecretpassword»);
$_POST = array(«name»=>«roadnick»,
«email»=>«ibmquestions@nicholaschase.com»,
«pword»=>$passwords);

У цьому варіанті для перегляду значення пароля ви повинні звертатися до відповідної змінної як до масиву із числовим індексом:...

```
foreach ($_GET as $key=>$value) {
echo «<p>«.$key.» = « . $value . «</p>«;
}

$passwords = $_GET[«pword»];
echo «First password =
«.$passwords[0]; echo «<br />«;
echo «Second password = «.$passwords[1];
...
```

Якщо відішлете форму (не забудьте обновити сторінку), то можете помітити деякі зміни, як показано нижче на рис. 14.9.

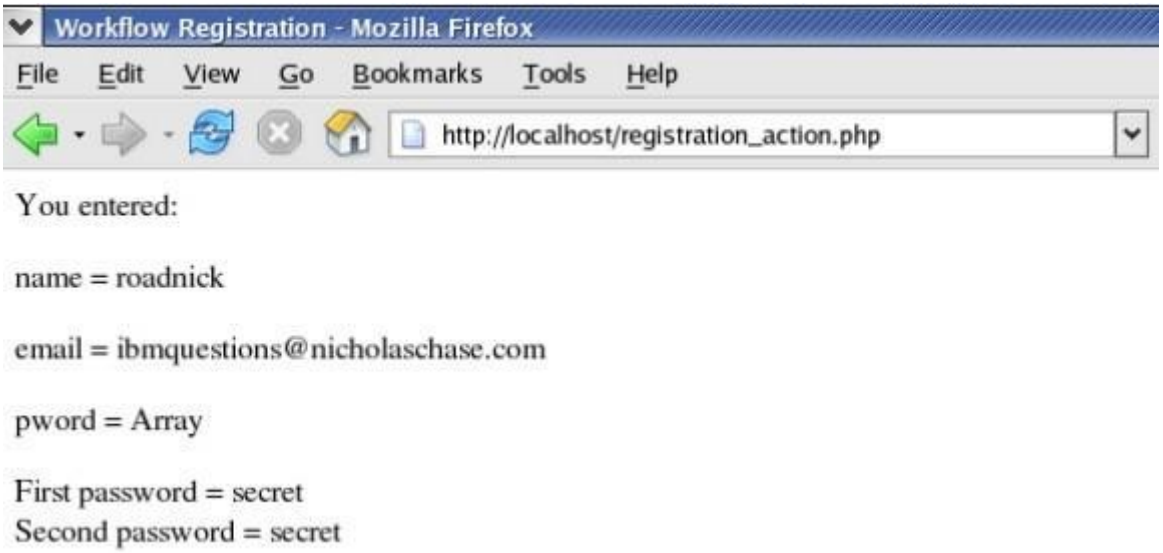


Рис. 14.9. Новий варіант виводу

Зверніть увагу, у полі, де раніше відображався пароль, з'явилося слово Array.

Відмінності методів GET і POST

Дотепер для передачі даних з форми використовували метод GET, головною незручністю й обмеженням цього методу є той факт, що передані дані містяться прямо на адресу URL. У деяких випадках це цілком припустиме, в інших – немає. Наприклад, потрібно передати з форми великий блок даних з поля типу textarea, звичайно, такий тип використовується для розміщення коментарів. Як правило, Web-сервер обмежує число символів, які можна помістити в Url-адресу, у такий спосіб передати дані за допомогою методу GET просто неможливо.

Згідно із прийнятими стандартами технології Web-програмування метод GET не слід використовувати для передачі даних у скрипти, які можуть мати «побічні ефекти», а фактично будь-який скрипт, у якому відбувається обробка даних, має цю властивість. Дотепер тільки відображали введені дані, обробки не відбувалося, але, якщо прагнемо додати запис у базу даних, то це вже обробка, яка може мати побічний ефект.

Той факт, що метод GET передає дані в Url-адресі відкритим, тобто доступним для перегляду способом, не тільки впливає на безпеку системи, але має й інші незручності. Наприклад,

корис-

тувач може зробити закладку на цю адресу, що приведе до того, що операція буде виконуватися багато разів; пошукові машини, які індексують Url-адреси, можуть перехопити дані, що направляються у вашу базу, і так далі.

Тому в більшості випадків має сенс користуватися методом POST, який передає дані в тілі запиту, а не в заголовку, як метод GET.

Використання методу POST

Зовні використання методу POST мало відрізняється від використання методу GET. Спочатку внесемо зміни у файл registration.php:...

<h1>Register for an Account:</h1>

<form action=«registration_action.php» method=«POST»>

Username: <input type=«text» name=«name» />

...

При відсиланні форми побачите, що зміст Url-адреси змінився: http://localhost/registration_action.php

Зміни потрібні й у файлі registration_action.php для приймання даних, замість масиву $_GET буде використовуватися масив $_POST:

...

```
<body>
  <p>You entered:</p>

<?php

  foreach ($_POST as $key=>$value) {
    echo "<p>".$key." = " . $value . "</p>";
  }

  $passwords = $_POST["pword"];
  echo "First password = ".$passwords[0];
  echo "<br />";
  echo "Second password = ".$passwords[1];
?>

</body>
</html>
```

Подальша робота з масивом $_POST відбувається так само, як і з масивом $_GET.

Контроль даних: умовний оператор if-then.

Вище попросили користувача ввести пароль двічі. Тепер подивимося, як можна перевірити, чи не було зроблено помилки при введенні пароля. Для цього використовуємо умовний оператор if-then:...

```
$passwords = $_POST["pword"];
echo "First password = ".$passwords[0];
echo "<br />";
echo "Second password = ".$passwords[1];

if ($passwords[0] == $passwords[1]) {
   echo "<p>Passwords match.  Thank you.</p>";
} else {
   echo "<p>Passwords don't match.  Please try again.</p>";
}
```

У заголовку умовного оператора в дужках поміщене деяке вираження (у нашому прикладі це $passwords[0] == $passwords[1]), яке може бути істинне (TRUE) або хибне (FALSE). Якщо воно іс- тинне, то PHP виконує той оператор, який іде за умовним вира- женням, це може бути блок операторів у фігурних дужках. Якщо воно хибне, то виконується блок, який випливає за ключовим словом else, а якщо його нема, то скрипт переходить до виконан- ня наступного оператора.

Зверніть увагу на те, що ми написали $passwords[0] == $passwords[1], а не $passwords[0] = $passwords[1], тобто використовували подвійний знак рівності. Це не випадково. Простий знак рівності працює як оператор присвоювання, тобто значення $passwords[0] у цьому випадку дорівнюватиме $passwords [1]. Нам потрібно зовсім не це, ми прагнемо зрівняти значення змінних і тому використовуємо оператор порівняння – подвійна рівність.

На рис. 14.10 показана сторінка, яку побачить користувач у тому випадку, якщо він зробив помилку при введенні пароля.

Рис. 14.10. Попередження про помилку при введенні пароля

Для формування більш складних логічних виражень можна застосовувати логічні оператори, наприклад, оператор И (&&) і оператор АБО (||). Розглянемо такий вираз:

```
if (($today == «Monday») && ($status == «Not a holiday»))
{ echo «GO TO WORK!!!»;
}
```

Значення логічного виразу в дужках буде істина (TRUE) у тому випадку, якщо сьогодні понеділок (Monday) і цей день не є святковим.

ЛАБОРАТОРНА РОБОТА 15

Тема. База даних Mysql

Мета: навчитись створювати бази даних, заносити дані, створювати запити, знищувати дані.

Хід роботи

Усі сучасні web-сервіси використовують бази даних, у них зберігають списки зареєстрованих користувачів. У пакет Denver входить база даних Mysql.

Припустимо, нам потрібно зберігати інформацію про учнів класу. Для цих цілей необхідна база даних, у якій усі дані будуть структуровані, і можна отримувати різну інформацію через запити. Якщо ви знайомі з базами даних, наприклад з MS Access, що входять в MS Office, то ви знаєте, що всі дані в базі даних зберігаються у зв'язаних таблицях. Кожна таблиця містить набір полів, кожне може зберігати інформацію одного типу. Якби створювали базу даних у MS Access, то таблиця повинна мати такі поля:

№	назва	тип поля
1.	ідентифікатор	лічильник
2.	прізвище ім'я	текстове
3.	e-mail	текстове
4.	дата народження	дата/час
5.	інформація про учня	поле MEMO
6.	рік вступу в школу	числовий

Тут необхідні невеликі пояснення:

- у таблиці повинне бути поле, значення якого унікальне для кожного запису. Найпростіший спосіб створення такого поля – це довірити цю роботу самій базі даних, поле типу «лічильник» буде формуватися автоматично і буде приймати значення 1, 2, 3 і т. д.;
- поле «MEMO» містить текстову інформацію великого обсягу на відміну від просто «текстового» поля, розмір якого об-

межено 255 символами;

поле «рік вступу в школу» зробити типом дата/час не зручно, тому що доведеться заповнювати не тільки рік, але й місяць, і день вступу.

База даних Mysql також оперує таблицями, полями, типами полів. Для цього в командному рядку Windows (кнопка Пуск – Виконати **cmd**) наберіть команду для запуску бази даних. Зверніть увагу, запуск Mysql походить із віртуального диска, у наведеному прикладі це диск Z.

```
Z:\>cd usr
Z:\usr>cd local
Z:\usr\local>cd mysql-5.5
Z:\usr\local\mysql-5.5>cd bin
Z:\usr\local\mysql-5.5\bin>mysql -u root
-p Enter password:
Welcome to the Mysql monitor. Commands end with ; or \g.
Your Mysql connection id is 1
Server version: 5.0.45-community-nt Mysql Community Edition
(GPL) Type ‘help;’ or ‘\h’ for help. Type ‘\c’ to clear the buffer.
mysql>
```

Параметри -u root -p, повідомляють, що запуск походить від імені суперкористувача root з порожнім паролем.

Уся робота з базою даних відбувається не через звичний графічний інтерфейс, а через командний рядок (рис 15.1).

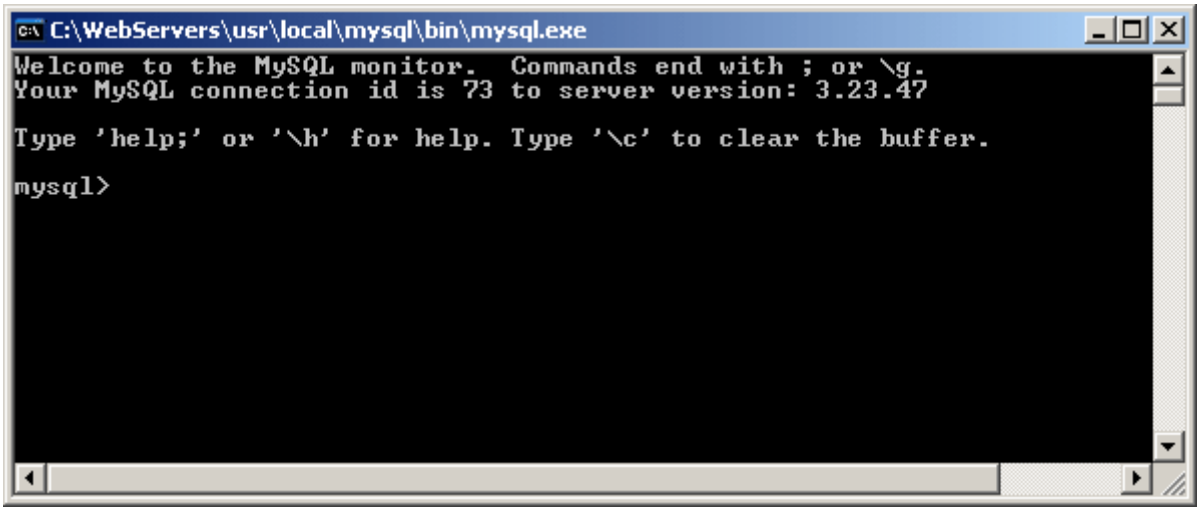


Рис. 15.1. Консоль для роботи з базою даних Mysql

Спосіб роботи з Mysql – це використання утиліти Phpmyadmin, яка входить в Denver і доступна за адресою http://localhost/phpmyadmin. Уся робота відбувається через web-інтерфейс. Подібні утиліти є на багатьох віддалених серверах, на яких ви будете розміщувати ваші сайти.

Команди Mysql

Усі команди друкуються на консолі. Запам'ятайте прості правила:

- Рядок у консолі починається із символів mysql>.
- Команди можуть бути досить довгими, і ви можете нати- скати на клавішу Enter, переносячи команду на наступний ря- док. Для завершення і виконання команди необхідно поставити символ «крапка з комою».
- У випадку, якщо ви ввели команду неправильно, на екра- ні з'явиться повідомлення про помилку з коментарем Mysql.
- Для того щоб не набирати всю команду заново, ви може- те нажати на клавіатурі кнопку зі стрілкою нагору, викликати по- передню команду, а після внести в неї виправлення.
- Відповідь консолі «Query OK (0.03 sec)» повідомляє, що команда успішно виконана.

Команди:
show databases; – команда, що показує вже існуючі бази даних.
Відповідь консолі представлено на рисунку 15.2.

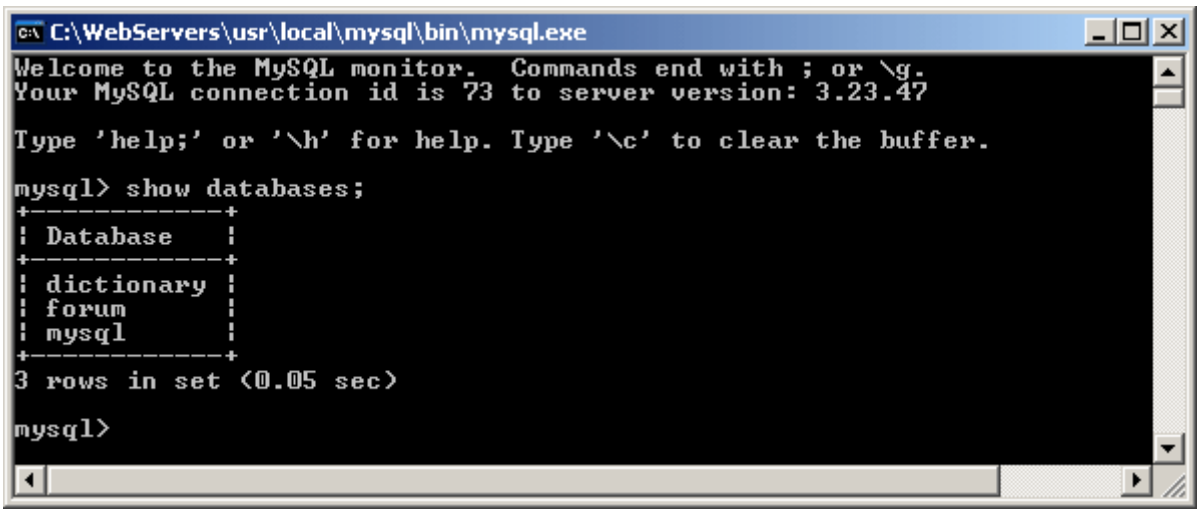


Рис. 15.2. Вивід баз даних Mysql

Ви бачите, що в цей момент у користувача створено три бази даних.

create database name; – створити базу даних name. Замість «name» ви повинні ввести ім'я вашої нової бази даних. При цьому створюється порожня база, що не містить ніяких таблиць. При цьо- му Denver створить у каталозі C:\Webservers\usr\local\mysql\data порожній каталог з іменем вашої бази даних. Ви можете використ- товувати цей каталог з усім умістом для переносу своїх баз із комп'ютера на комп'ютер.

use name; – перейти до бази даних «name». Ту саму команду можна записати коротше: **\u name**

drop database name; – вилучити базу даних name. При цьому буде вилучений каталог, у якому зберігалася база даних з усім умістом. Ніяких додаткових питань при видаленні Mysql задавати не буде, так що будьте уважні.

Створення нової таблиці даних

Після того, як база даних створена (і ви зайшли в неї за допомогою команди \u), необхідно створити таблицю, описуючи всі поля, які будуть у ній:

```
mysql> create table student (id_student int not null
auto_increment, name varchar(50),
birthday
date, info
text,
e_mail
varchar(50), yare
int,
primary key (id_student));
Query OK, 0 rows affected (0.06 sec)
```

У даному прикладі створена таблиця student, що містить шість полів. У першому рядку ми задали ім'я таблиці. Далі йде опис поля id_student, воно повинне бути цілим (int), не може бути порожнім (not null) і автоматично збільшуватись на одиницю для кожного нового запису (auto_increment). Це – аналог поля типу

«лічильник» у MS Access. Поле name містить текст довжиною в 50 символів, поле birthday має тип дата (у ньому зберігається

інформація виду 2006-08-24). Поле info може містити фрагмент тексту великого обсягу. Останній рядок задає ключове поле (id_author), через яке дана таблиця може зв'язуватися з іншими таблицями.

Назва бази даних і всі поля обов'язково пишеться латиницею, без пробілів.

На віддалених серверах, як правило, дозволяється використовувати тільки одну базу даних (додаткові бази за додаткові гроші). А кількість таблиць у базі не обмежено, тому в одній базі зберігають усі таблиці. Щоб працювати з ними було зручно, усім таблицям для одного сервісу використовувати однотипні назви, наприклад: school_students, school_teachers і т.д.

Після того, як таблиця створена, ви можете переглянути її структуру за допомогою команди:

mysql> describe student;

ви одержите наступне повідомлення:

mysql> describe student;

+_____+_____+____+___+_____+_____+

| Field | Type | Null | Key | Default | Extra |

+_____+_____+____+___+_____+_____+

| id_student | int(11) | | PRI | NULL | auto_increment |

| name | varchar(50) | YES | | NULL | |

| birthday | date | YES | | NULL | |

| info | text | YES | | NULL | |

| e_mail | varchar(50) | YES | | NULL | |

| yare | int(11) | YES | | NULL | |

+_____+_____+____+___+_____+_____

+ 6 rows in set (0.00 sec) mysql>

Для додавання даних у таблиці існує універсальна мова SQL.

Sql-запити. Абревіатура SQL розшифровується як Structure Query Language – структурована мова запитів. За допомогою цієї мови ви можете витягти інформацію з будь-якої сучасної бази даних.

Основні групи запитів:

Додавання даних у таблицю

mysql> INSERT INTO student VALUES(null, ‘Іванов’, ‘1993-02–20’, «, «, ‘2004’);

Дана команда додає в таблицю user запис про учня. Повинні бути перераховані всі поля у тій послідовності, в якій вони створювалися за допомогою команди create table. Тому що перше поле id_student формується автоматично, ми не повинні самі додавати значення, це зробить параметр null. Поля «інформація про учня» і «e-mail» залишені порожніми, але вони однаково повинні бути зазначені в запиті. У даному запиті використовували російські букви, але необхідно пам’ятати, що Mysql підтримує кодування koi-8, тому при виводі даних у браузері з такими символами бу- дуть проблеми. Якщо команда набрана правильно, ви одержите відповідь:

Query OK, 1 row affected (0.33 sec)

Команди SQL нечутливі до регістру, тому ключові слова набирати заголовними буквами, а змінювані параметри, імена – малими літерами. Ця звичка вам пригодиться при написанні програмного коду.

Вибір даних

mysql> SELECT * FROM student;

Команда вибирає всі значення з таблиці student. Якщо необхідно вибрати тільки частину значень, можна запит побудувати трохи інакше:

mysql> SELECT * FROM student WHERE yare > 2001;

У такому випадку будуть виведені на екран тільки ті учні, які поступили в школу після 2001 року.

За допомогою запиту на вибірку даних ви можете здійснювати сортування даних:

mysql> SELECT * FROM student ORDER BY name;

Такий запит буде коректно сортувати за зростанням числової інформації, а також англомовного тексту. У випадку, якщо сортувати поля з кирилицею, даний запит необхідно видозмінити:

mysql> SELECT * FROM student ORDER BY binary(lower(name));

Видалення даних

mysql> DELETE FROM student WHERE id_ student = 5;

Даний запит видаляє запис із таблиці для учня, у якого ідентифікатор рівний п’яти.

Наступний запит вилучить усі записи з таблиці:

mysql> DELETE FROM student WHERE id_ student >= 1;

Зміна даних

mysql> update student set student =‘Петров’, birthday=‘1993-06-15’ where id_student=4;

Запит на зміну даних у запису з ідентифікатором id_student. рівним чотирьом. Можна змінити одне або кілька полів у даного запису. Порядок змінюваних полів у запиті значення немає.

Практичні завдання

1. У режимі командного рядка створіть базу даних «users», яка повинна містити інформацію про зареєстрованих користува- чів сайта. Обов’язкові поля: ім’я клієнта, логін, пароль, e-mail, додаткова інформація. Ви можете доповнити список полів на свій розсуд. У режимі командного рядка заповніть не менше п’яти за- писів у створеній вами таблиці.
2. Напишіть sql-запит для виводу всіх користувачів, у яких порожній пароль.
3. (*) Запустіть Denver, запустіть утиліту Phpmyadmin, http://localhost/phpmyadmin. Створіть базу даних, створіть у ній таблицю.

Питання для самоконтролю

1. Як запустити консоль Mysql?
2. Як довідатися, які бази даних доступні вам на вашому комп’ютері?
3. Що означає символ зірочки в запиті на вибірку даних?

4. Як за допомогою одного запиту вилучити всі дані з таблиці?

5. Про які Sql-запити ви довідалися на цьому занятті?

6. Що відбувається на жорсткому диску комп’ютера при створенні нової бази даних? Нової таблиці?

ЛАБОРАТОРНА РОБОТА 16

Тема. Налагодження бази даних Денвер у phpmyadmin. Mysql в Denwer

Мета: навчитись створювати бази даних та робити запити за допомогою Денвер.

Хід роботи

Phpmyadmin – це інтерфейс, що дозволяє працювати з базою даних. Mysql бази даних – основний інструмент для створення динамічних сайтів.

Принцип роботи полягає у наступному: створюється Html-каркас сайта і в певні місця каркаса (наприклад у область основного вмісту) за допомогою Php-скриптів з бази даних виводиться інформація, яка і формує контент сайта.

У даній лабораторній роботі ви навчитесь створювати бази даних на локальному комп'ютері за допомогою Денвера.

phpmyadmin – це утиліта з відкритим кодом, написана на PHP, що і забезпечує повноцінне адміністрування баз даних Mysql через браузер.

Налагодження **Mysql в Denwer** проводитися за адресою http://localhost/Tools/phpmyadmin/. Щоб це посилання відкрилося, нам спочатку потрібно запустити Денвер за допомогою ярлика, який перебуває на Робочому столі. У **Denwer налагодження бази даних** стоїть на першому місці. Тому починаємо саме з нього, а інші робляться за необхідності. Головна сторінка **phpmyadmin** у Денвері зображена на рис. 16.1.

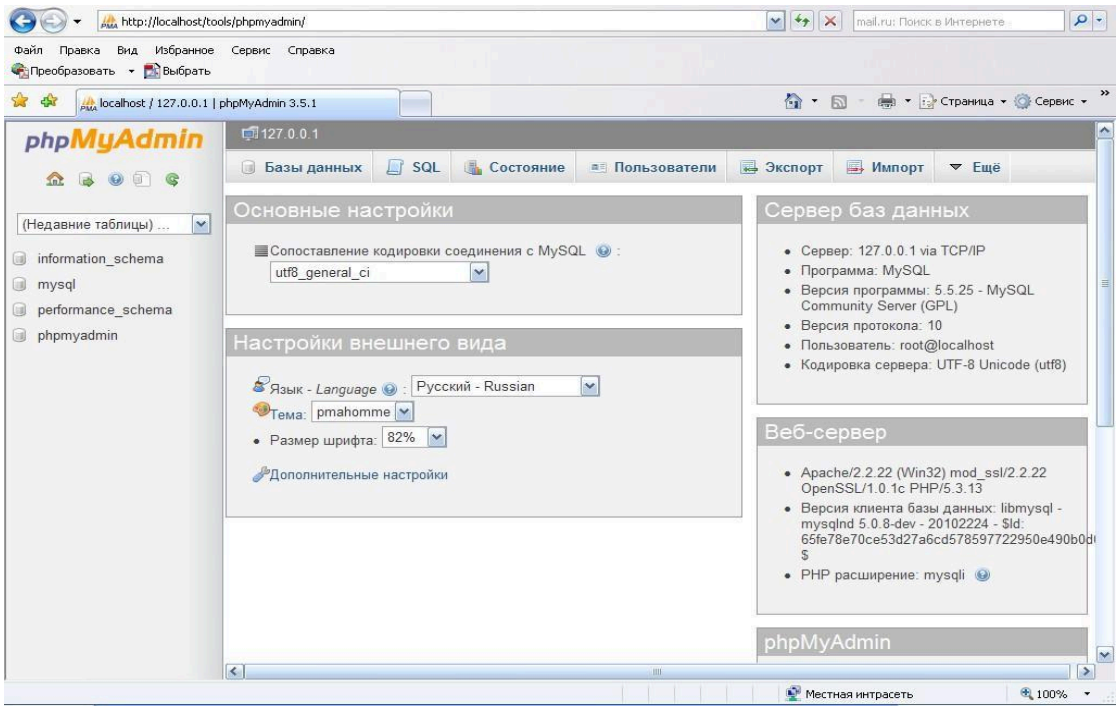


Рис. 16.1. Головна сторінка phpmyadmin у Денвері

З лівої сторони розташований список наявних Mysql баз даних у Денвері. Звичайно, можна використовувати одну базу даних для декількох сайтів, додавши кожному із сайтів свої префікси. Але якщо Хостинг провайдер не обмежує число створених баз даних, звичайно, краще створити кожному сайту окрему базу даних. А Денвер phpmyadmin дозволяє створювати необмежену кіль- кість баз даних. Для налагодження Mysql у Денвері, на головній сторінці phpmyadmin знадобляться всього лише два блоки. У цих двох блоках настроюються найголовніші налагодження (рис. 16.2).

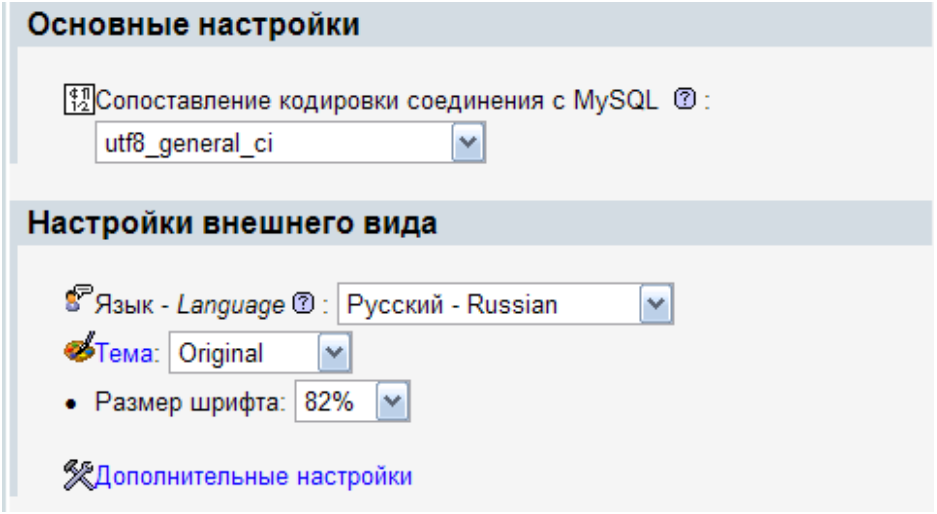


Рис. 16.2. Основні налагодження

Починаємо налагодження в Denwer бази даних. У першому блоці є список, що випадає, набір символів, який зіставляє з'єднання *з Mysql в* Denwer. Якщо в цьому списку у вас обране не «utf8_general_ci» (рис. 16.3), то виберіть його.

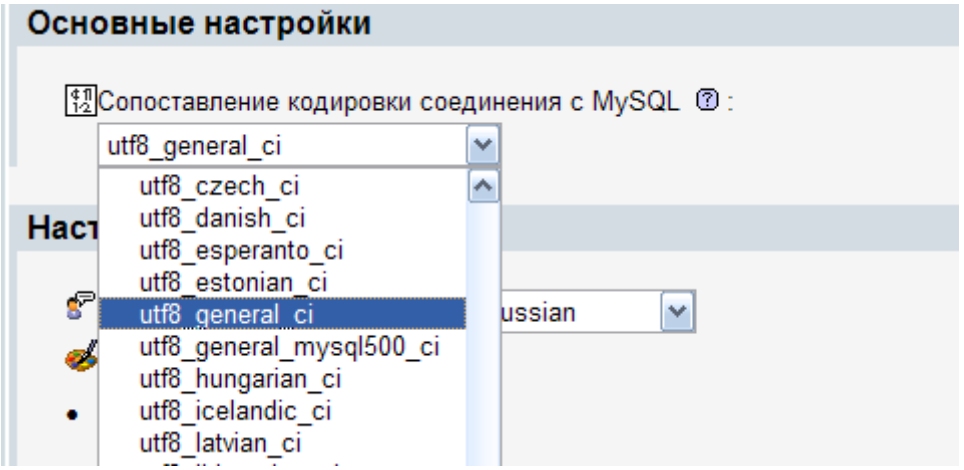


Рис. 16.3. Вибір кодування

«utf8_general_ci» – це основний набір символів кодування «UTF-8». «UTF-8» – кращий варіант кодування, на якому варто створити сайт. Він підтримує і кирилицю, і навіть китайські ієрог- ліфи. Наступне налагодження не дуже важливе, але може знадо- битися. Це вибір мови інтерфейсу phpmyadmin в Denwer (рис. 16.4).

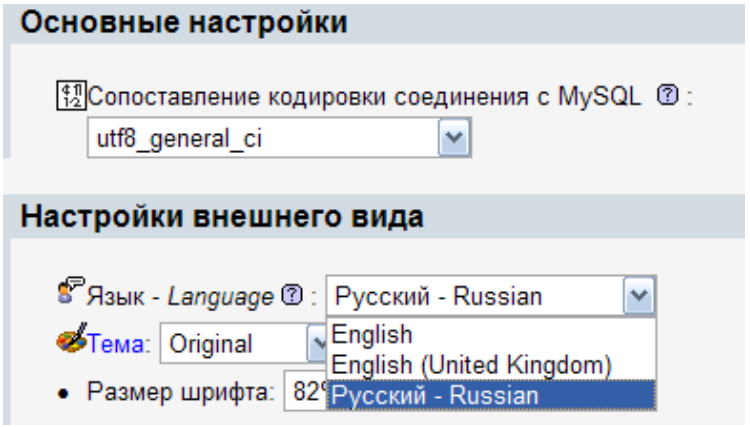


Рис. 16.4. Вибір мови інтерфейсу

Вибираємо потрібну мову інтерфейсу phpmyadmin. Денвер в основному настроєн на російську мову і phpmyadmin теж багатомовна утиліта. Але раптом, якщо там не виявиться та мова, на якій вам було б зручно працювати, то знайти його не важко. А тепер поговоримо про меню phpmyadmin (рис. 16.5).

Рис. 16.5. Вікно меню

Є кілька пунктів у меню phpmyadmin, з яких вам знадобляться лише три. Перше – це користувачі.

Переходячи по цьому меню, ми попадаємо в сторінку облікових записів Mysql в Denwer (рис. 16.6). Для бази даних Денверу вистачить одного облікового запису, тому що локальний сервер (localhost) перебуває на нашому комп'ютері. Крім нас, на локальний сервер ніхто не має доступу. А для хостинга краще створити окремий обліковий запис. Після таблиці користувачів є посилання «Додати нового користувача». Натискаючи на нього, переходимо на сторінку додавання нового користувача Mysql в Denwer (рис. 16.7).

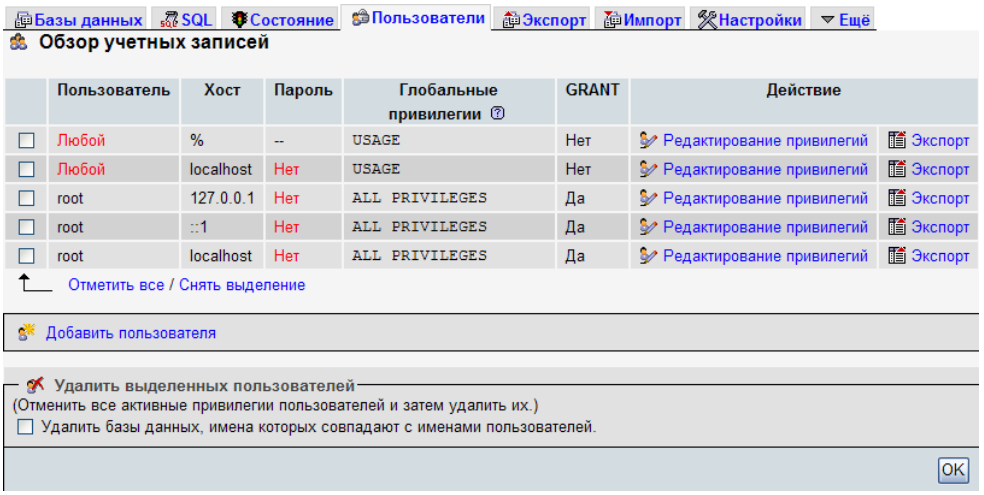


Рис. 16.6. Сторінка облікових записів

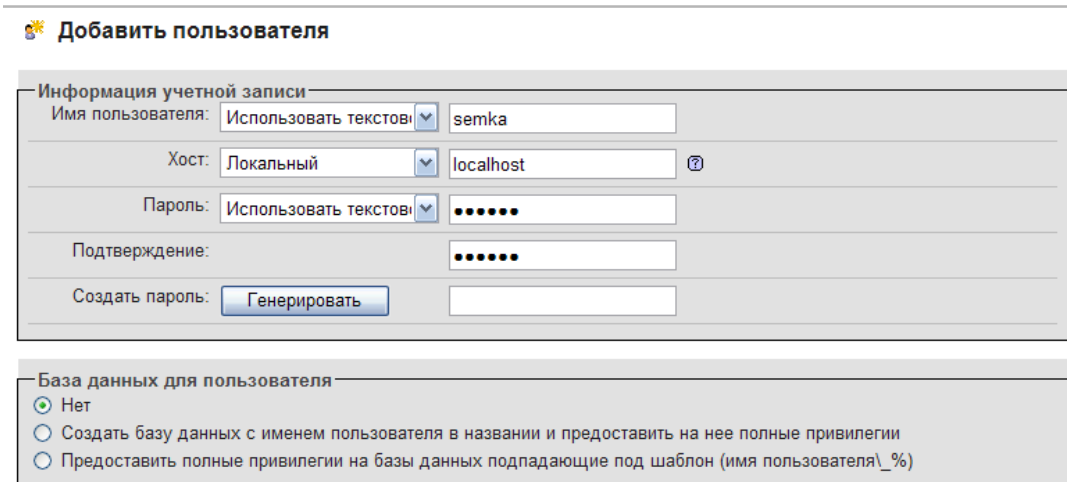


Рис. 16.7. Додавання нового користувача

У першому полі вводимо ім’я користувача. У другому полі «Хост» з меню, що впадає, вибираємо «локальний», у результаті автоматично вводитися «localhost». У наступних полях вводимо па- роль як звичайно. У блоці «База даних для користувача» залиша- ємо всі як є і переходимо в блок «Глобальні привілеї» (рис. 16.8).

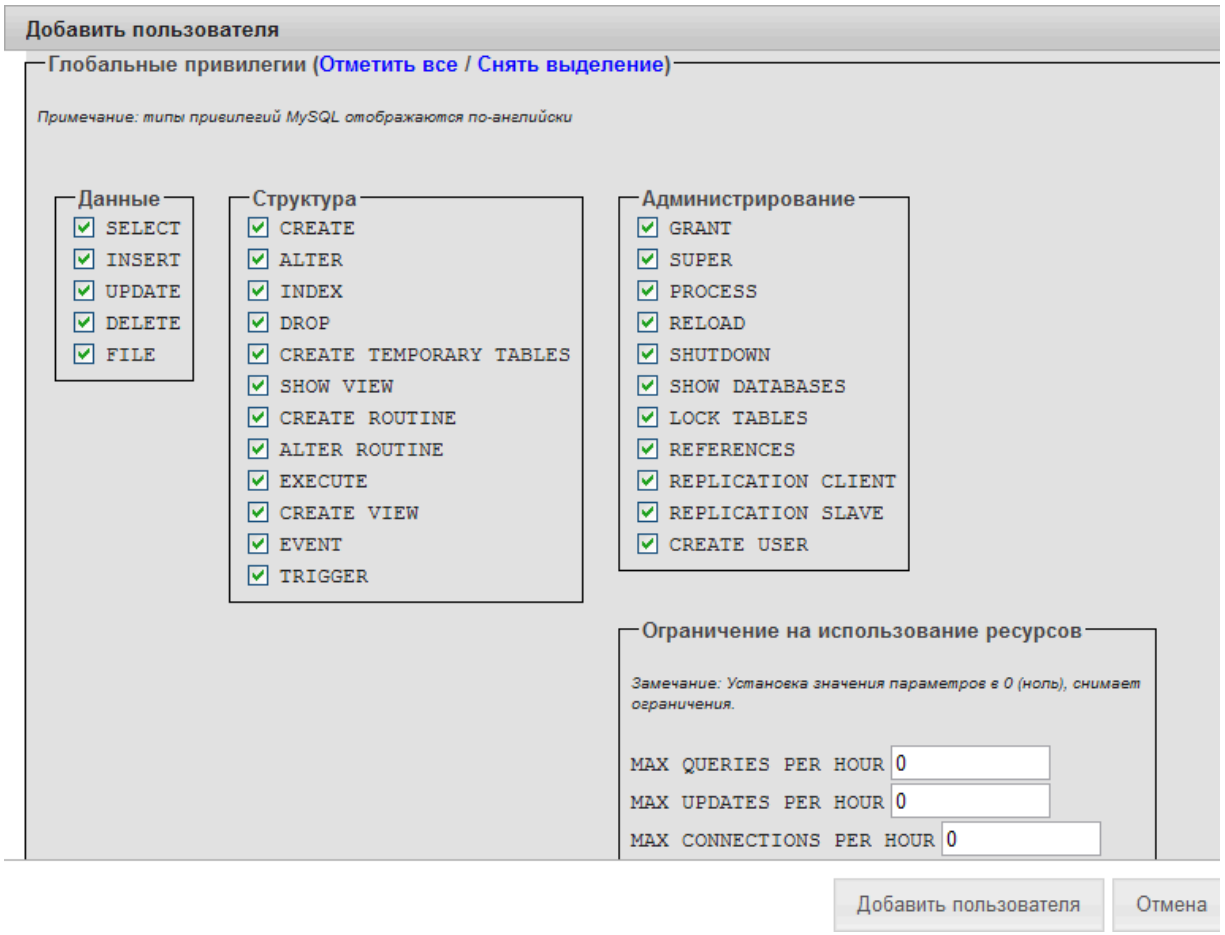


Рис. 16.8. Глобальні привілеї

У цьому блоці вибираємо, які привілеї будемо давати цьому користувачеві. Тому що ми – власник цього локального сервера, натискаємо на посилання «Відзначити всі» і наприкінці сторінки натискаємо на «ОК». Після успішного додавання нового користувача знову переходимо на сторінку «Користувачів», щоб переконатися про додавання нового користувача до бази даних Denwer.

Наш новий користувач стоїть в останньому рядку таблиці користувачів. Тому що в нас уже є обліковий запис, з яким будемо працювати надалі, можна створювати нашу першу базу даних

у Денвер (рис. 16.9).

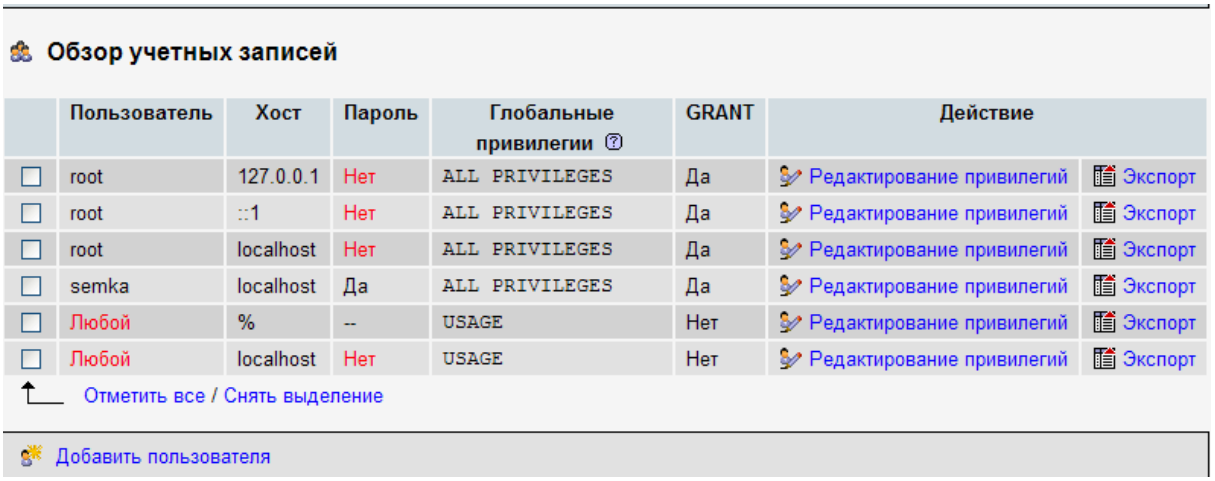


Рис. 16.9. Обліковий запис

Переходимо на головну сторінку phpmyadmin Денвера, щоб створити нову базу даних Mysql у Денвері (рис. 16.10). У першому блоці, який називається «Mysql localhost», у поле «Нова база даних» вводимо назву нової бази даних.

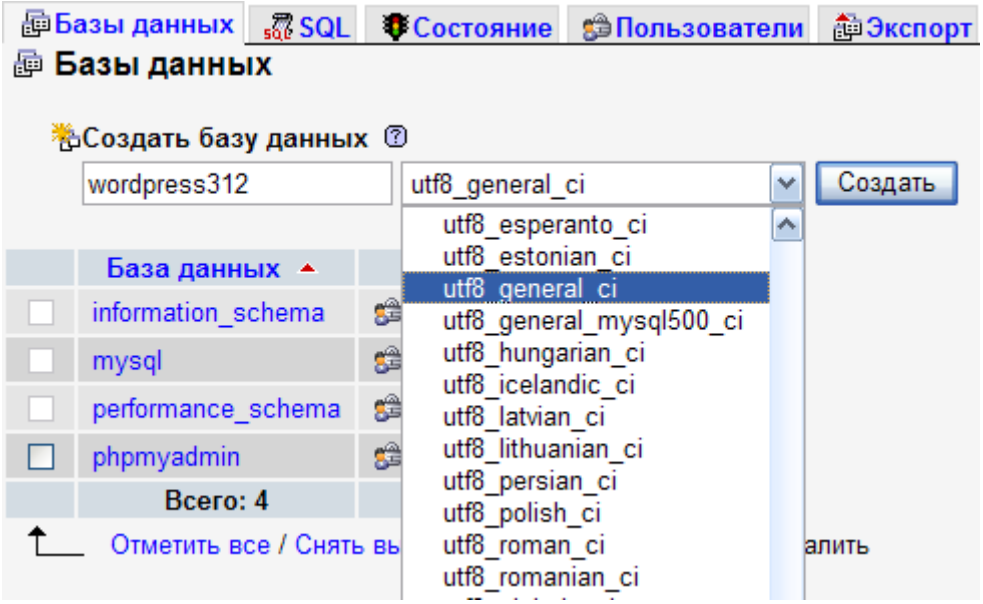


Рис. 16.10. Створення нової бази даних

Краще ввести зрозумілу назву, щоб було б зрозуміло за назвою, якому сайту належить та або інша база даних. Особливо, коли набереться така велика кількість баз даних. Вводимо в це поле «wordpress312». Далі на меню, що випадає, вибираємо кодування майбутньої бази даних і натискаємо на кнопку «створити». Після створення нової бази даних Mysql у Денвері ви одразу попадаєте усередину цієї бази даних (рис. 16.11).

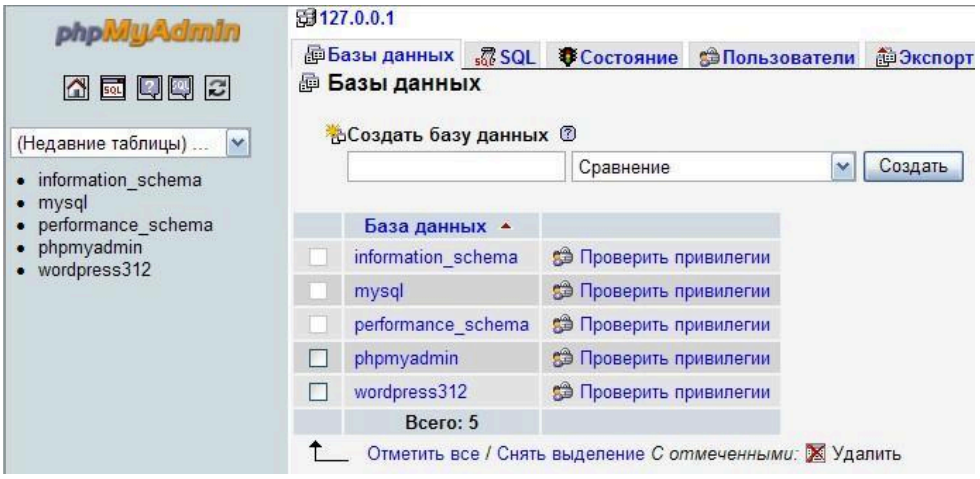


Рис. 16.11. Вміст БД

Тут поки порожньо. Немає жодної таблиці. І тут поки робити нема чого. Зверху, в що випадає меню і під ним бачите назву бази даних, це означає, що ви перебуваєте саме усередині цієї бази даних.

Тепер поговоримо про функції phpmyadmin Денвера, які знадобляться надалі. Залишилися два пункти меню, які будете використовувати. Це «Експорт» і «Імпорт». Перейдемо на сторінку «Експорт» (рис. 16.12).

У блоці «Експорт» виділяємо назву бази даних, яку прагнемо експортувати. Можемо виділити і кілька баз даних, і всі одразу. Якщо вам необхідно експортувати деякі таблиці певної бази даних, то спочатку потрібно ввійти усередину бази даних, а потім перейти по меню «Експорт». Тоді можете виділити таблиці бази даних. Після виділення назви бази даних, які прагнемо експортувати, переходимо в кінець сторінки.

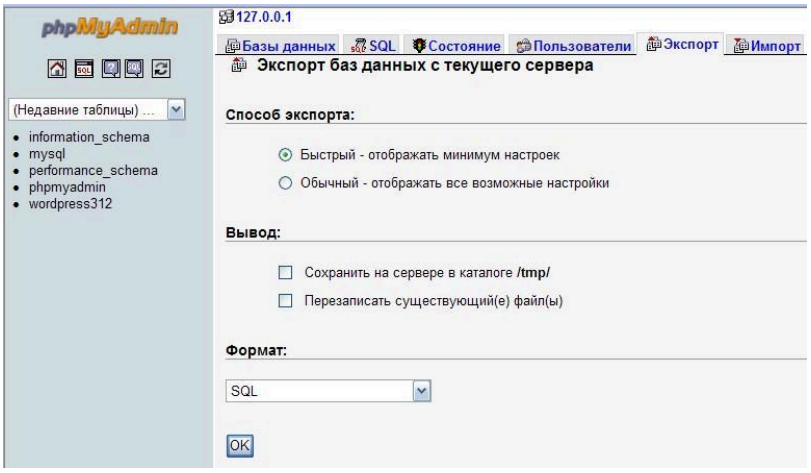


Рис. 16.12. Експорт БД

Рис. 16.13. Зберігання файла

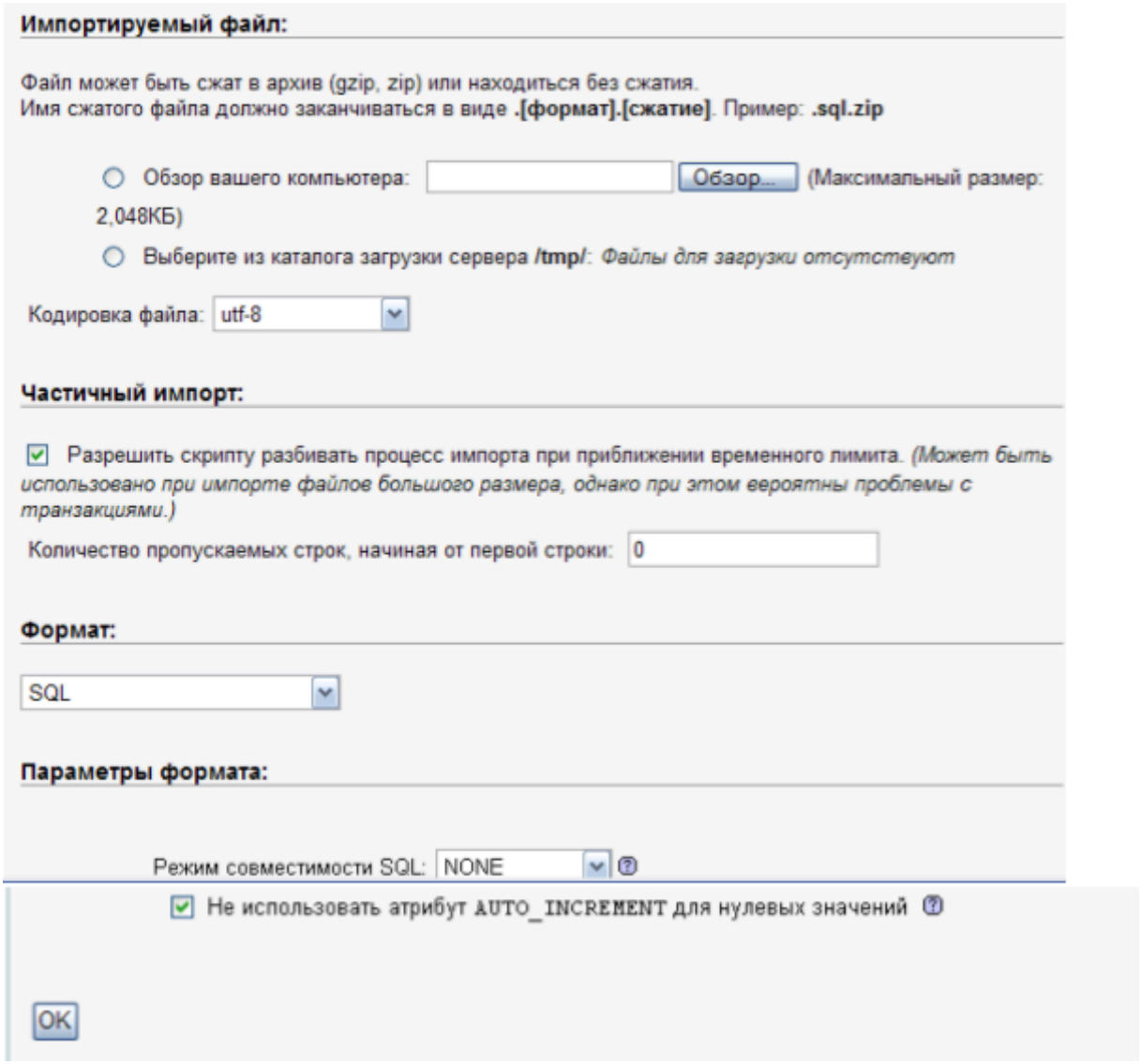


Рис. 16.14. Імпорт файла

Інші параметри краще залишити недоторканими. А в блоці «Зберегти як файл» вибираємо «gzip» (рис. 16.13). Звичайно, можна вибирати будь-який варіант, але «gzip» максимально архівує базу даних. Так швидше і експортувати й імпортувати. Настав час

натискати на кнопку «ОК», зберегти архівну копію бази даних у зазначенім місці вашого комп'ютера.

Перейдемо на сторінку «Імпорт» (рис. 16.14). Тут усе легко. Вибираємо копію бази даних натисканням на кнопку «Огляд», вибираємо кодування файлу і натискаємо на кнопку «ОК».

Отже, ми розглянули основні налаштівання Денвера. Phpmyadmin повністю готовий до роботи. У нас є користувач бази даних Mysql і нова база даних «wordpress312», яку можна використовувати.

ЛАБОРАТОРНА РОБОТА 17

Тема. Доступ до даних через Web-інтерфейс

Мета: навчитись підключатись до бази даних, створювати форму для введення і виведення даних.

Хід роботи

У даній лабораторній роботі ви навчитесь підключати базу даних Mysql до сайта, а також довідаєтеся, як можна управляти даними через web-інтерфейс (тобто через браузер).

На даний момент у вас повинна бути створена база даних «school», що включає таблицю «user». Таблиця повинна містити п’ять полів:

id_user (лічильник);
user_name (текст);
user_login (текст);
user_password
(текст); user_e-mail
(текст);
user_info (текст великого обсягу).

У ході заняття ви створите PHP-файл, який з назвою user.php. Він повинен мати таку структуру:

код для підключення до бази даних (лістинг 1);

код для видалення запису в базі даних (чому код для видалення перебуває до виводу обговориться пізніше) (лістинг 5);

форма для введення значень у базу даних (лістинг 2);

код для додавання запису в базі даних (лістинг 3);

код для висновку значень на екран (лістинг 4).

Підключення бази даних

Для того щоб додавати інформацію у базу даних, необхідно

спочатку сторінки (наприклад, після тегу < body >) додати такі рядки php-коду:

Лістинг 1

```
<?
$db = mysql_connect("localhost", "root");
mysql_select_db("school", $db);
?>
```

Перший рядок за допомогою функції mysql _ connect () створює об'єктну змінну $db. Функція має два параметри ім'я «хоста», де розташовується база даних, а другий – ім'я користувача, що має право на з'єднання з базою (root – це користувач створюваний системою, що й має максимальні права). У другому рядку відбувається підключення до конкретної бази даних.

Наведені рядки будуть коректно працювати на локальному комп'ютері, якщо у вас установлений Denver, але при спробі перенести ваш сервіс на віддалений сервер відбудеться помилка, адже там база буде розташовуватися в іншому місці, права користувача також набудовує адміністратор і він вам не дасть прав адміністратора, отже, ці два рядки потрібно міняти. Причому міняти у всіх php-файлах даних, що використовують підключення до бази. Має сенс винести ці чотири рядки в окремий файл, назвати його connect.php, а в основному php-файлі зробити посилання:
<? include («connect.php»); ?>

Це не помилка, для файла, що зберігає підключення до бази даних, краще використовувати розширення php. Тому що, якщо ви використовуєте розширення .inc, браузер буде сприймати його як текстовий файл і якщо зловмисник набере в рядку адреси connect.inc, він одержить ім'я бази даних і паролі до неї. Зміст php-файла буде приховано, сервер у відповідь на запит згенерує порожню html-сторінку.

Створення форми для введення даних

Для того щоб ви могли заповнювати дані, необхідно створити форму (лістинг 2).

Лістинг 2

```
<H2>Персональна інформація:</H2>
<form action="user.php" method="get">
<p>ім'я: <input name="name" size="50" type="text"></p>
<p>логін: <input name="login" size="50" type="text"></p>
<p>пароль: <input name="password" size="50" type="password"></p>
<p>e-mail: <input name="e_mail" size="50" type="text"></p>
<p>інформація: <textarea name="info" rows="4" cols="40"></textarea></p>
<p><input name="add" type="submit" value="додати"><input name="b2"
type="reset" value="очистити"></p>
</form>
```

Як видно з коду, дані вводяться у п’ять полів і при натисканні на кнопку «додати» змінні форми передаються у той же самий файл user.php.

Додавання даних у таблицю

Лістинг 3

```
<?
if (isset($_GET['add'])){
$name = $_GET['name'];
$login = $_GET['login'];
$password = $_GET['password'];
$e_mail = $_GET['name'];
$info = $_GET['info'];
$sql_add = "INSERT INTO users VALUES (null, '$name', '$login', '$password',
'$e_mail', '$info')";
mysql_query($sql_add);
print("<p>Спасибі, ви зареєстровані в базі даних</p>");
}
 ?>
```

Змінна «add» буде передана у файл user.php разом з іншими змінними і якщо це відбудеться, виконується чотири дії.

З форми будуть витягнуті всі змінні (у наведеному фрагменті вони передавалися методом GET).

Створюється строкова змінна $sql_add, що містить sql-запит. У ньому замість значень полів використовуються імена витягнутих змінних, у яких зберігається ім’я користувача, логін тощо. Кожна змінна взята в апострофи.

За допомогою функції mysql_query() виконується sql-запит. Аргументом цієї функції є рядок із запитом.

Друкується повідомлення про додавання нового користувача.

Виведення даних з таблиці на екран

Припустимо, що ви прагнете вивести на екран список зареєстрованих користувачів. При цьому не потрібно виводити персональну інформацію, логін, пароль, досить тільки імена і електронні адреси. Для виводу даних можна використовувати такий код (лістинг 4).

Лістинг 4

```
<h2>Зареєстровані користувачі:</h2>
<?
$sql_select = "SELECT id_user, user_name, user_email FROM user ORDER BY
binary (lower(user_name))";
$result = mysql_query($sql_select);
 $num = mysql_num_rows($result);
print("<P>Усього користувачів: $num.</p>");
 while($row = mysql_fetch_array($result)){
print("<p>$row[user_name] - ");
 print("<a href=\"user.php?act=delete&id_user= $row[id_user] \">вилучити</a>,
");
 print("<a href=\"mailto:$row[user_email]\">написати лист</a></p>");
 }
?>
```

Наведений фрагмент такий складніший. Для більш легкого сприйняття його поділили на три частини.

У першій частині створюється текстова змінна $sql_select, у якій формується sql-запит на вибірку даних із сортуванням по полю «user_name». Зверніть увагу, в sql-запиті перераховані не всі, а тільки три поля, які нас цікавлять у вибірці. Функція mysql_query() виконує цей запит, і результат затягає в змінну $result. Можна уявити собі, що в цій змінній зберігається не одне значення, а таблиця значень.

У другій частині за допомогою функцій mysql _ num _ rows () рахується кількість записів, витягнутих з таблиці. Аргументом

функції є змінна $result. Отриманий результат виводиться на екран.

У третій частині обрані дані виводяться на екран. Для цього змінна $result розбивається на рядки. Кожний рядок містить дані про один запис таблиці. Доти поки така розбивка можлива (використовується цикл while) кожна строка за допомогою функції mysql_fetch_array() розбивається на окремі значення і кожне значення заноситься у масив $row[] (рисунок 17.1).

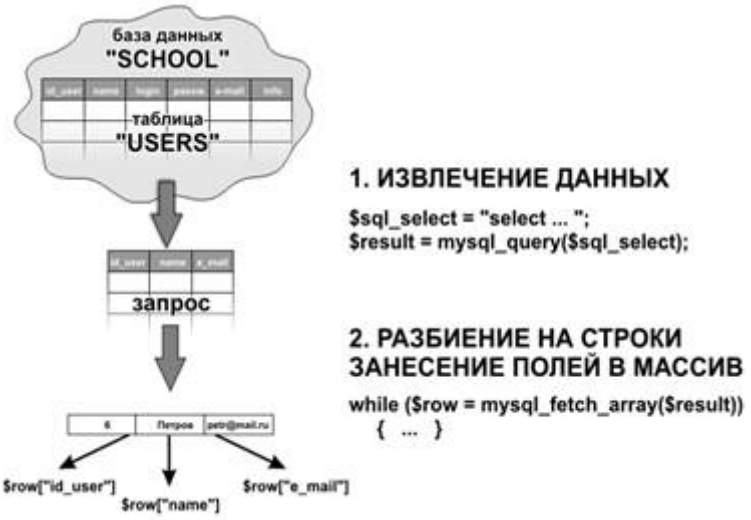


Рис. 17.1. Схема витягу даних з таблиці

Тепер ви можете виводити значення на екран. Зверніть увагу на два моменти:

1. Для створення посилання необхідно використовувати тег <a href=«/…»>…</ a>.

Але у символі лапки усередині функції print () своя робота – обмеження текстового рядка, який необхідно виводити на екран. Тому в наведеному лістингу перед лапками в тегу < a > використовується символ «зворотний слеш». Він маскує лапки html-тегу усередині текстового рядка.

2. Дії щодо видалення даних з таблиці знаходяться у тому самому файлі user.php. Тому разом з адресою сторінки в тегу <a> передаються дві змінні act (тип дії) і id_user (ідентифікатор користувача, що знищується). Значення ідентифікатора береться з масиву $row[id_user].

Видалення даних з таблиці

Код щодо видалення даних з таблиці (лістинг 5) необхідно розташувати до коду виводу користувачів на екран. Адже, коли людина натисне посилання «вилучити», відбудеться перезавантаження файла user. php і дані повинні бути вилучені з таблиці раніше, ніж буде виконаний SQL-запит на вибірку даних.

Лістинг 5

```
<?
if (isset($_GET[‘act’])) {
$id_user = $_GET[‘id_user’];
$sql_delete = «DELETE FROM user WHERE id_user= $id_user»;
mysql_query($sql_delete);
}
?>
```

Програмний код виконується у тому випадку, коли у файл передається змінна $act. У наведеному коді формується текстовий рядок з sql-запитом. Далі цей запит виконується за допомогою вже відомої функції mysql_query() .

Отриманий програмний код недосконалий. Перелічимо головні недоліки:

Ви можете реєструвати того самого користувача багаторазово, дані будуть заноситися у таблицю, змінюючи тільки поле id _ user .

Ви можете реєструвати порожніх користувачів, адже в програмному коді немає перевірки.

Відсутня можливість змінювати введені дані.

Усі ці недоліки можна досить легко усунути, і це буде одним із практичних завдань.

Практичні завдання

Створіть php-файл, який може додавати, видаляти/змінювати значення в базі даних «Клієнти».

Внесіть зміни в php-код, щоб програма перевіряла, чи введене ім’я користувача, його логін і тільки в цьому випадку вносила дані в таблицю.

(*) Створіть модифікований php-файл, у якому виправлені перераховані вище недоліки.

Запитання для самоконтролю

1. Чому зручніше підключення до бази даних виводити в зовнішній файл і підключати його за допомогою функції include()?
2. Для чого при виводі даних використовується цикл While?
3. Чому код на видалення даних потрібно вставляти на початок сторінки?
4. Які нові функції ви вивчили, виконуючи лабораторну роботу?

ЛАБОРАТОРНА РОБОТА 18

Тема. Технологія XML

Мета: набуття практичних навичок про синтаксис, основні конструкції та застосування мови XML.

Хід роботи

1. У блокноті створіть XML-файл з ім’ям (ex01.xml).

```
<?xml version=«1.0» encoding=«WINDOWS-1251»?>
<tutorial>
<title>«Замітки про XSL»</title>
<author>Сиротенко Сергій Петрович</author>
</tutorial>
```

Відкрийте цей файл у браузері. У звіті занотуйте інформацію, яку побачили. Для знищення зайвої інформації до XML-файла додамо шаблон перетворення – XSL-файл.

Перепишіть Xml-файл у наступному вигляді (ex01-1.xml).

```
<?xml version=«1.0» encoding=«WINDOWS-1251»?>
<?xml-stylesheet type=‘text/xsl’ href=‘ex01-1.xsl’?>
<tutorial>
<title>«Замітки про XSL»</title>
<author>Сиротенко Сергій Петрович</author>
</tutorial>
```

І створіть XSL-файл ex01-1.xsl.

```
<xsl:stylesheet version=«1.0»
xmlns:xsl=«http://www.w3.org/TR/Wd-xsl»>
<xsl:template match=«/»>
<p><strong><xsl:value-of select=«//title»/></strong></p>
<p><xsl:value-of select=«//author»/></p>
</xsl:template>
</xsl:stylesheet>
```

Якщо відкрити файл ex01-1.xsl у браузері. Занотуйте у звіті відображену інформацію на екрані. Зробіть висновки.

Результат, який ви отримаєте на екрані браузера, наведений нижче.

«Замітки про XSL»
Сиротенко Сергій Петрович

2. Легко також побачити, що порядок виводу рядків визначається тільки змістом шаблону перетворення – XSL-файла. За необхідності шаблон можна легко поміняти, абсолютно не міняючи основний XML-файл.

Перепишіть XML-файл. Інформаційну частину змінювати не треба, а шаблон вкажіть інший ex01-2.xml.

```
<?xml version=«1.0» encoding=«WINDOWS-1251»?>
<?xml-stylesheet type='text/xsl' href='ex01-2.xsl'?>
<tutorial>
<title>«Замітки про XSL»</title>
<author>Сиротенко Сергій Петрович</author>
</tutorial>
```

Створіть XSL-файл ex01-2.xsl.

```
<xsl:stylesheet version=«1.0»
xmlns:xsl=«http://www.w3.org/TR/Wd-xsl»>
<xsl:template match=«/»>
<p><strong><xsl:value-of select=«//author»/></strong></p>
<p><xsl:value-of select=«//title»/></p>
</xsl:template>
</xsl:stylesheet>
```

Якщо відкрити файл ex01-2.xsl у браузері, то результат буде таким:

Сиротенко Сергій Петрович
«Замітки про XSL»

4. Інформація у XML-сторінці з'являється, як правило, у ре- зультаті запиту до бази даних. Фактично XML-сторінка – це і є тимчасовий буфер для результатів запитів. Тільки замість нестан- дартного і трудомісткого програмування використаємо стандарт- ний механізм XSL.

Більшість сучасних СУБД можуть форматувати результати запиту до бази даних у вигляді XML-файла.

Введіть програмний код.

```
<?xml version=«1.0» encoding=«WINDOWS-1251»?>
<tutorial>
<title>«Замітки про XSL»</title>
<author>Сиротенко Сергій Петрович</author>
</tutorial>
```

Перший рядок інформує браузер про те, що файл має формат XML. Атрибут version є обов'язковим. Атрибут encoding не є обов'язковим, але якщо у тексті є російські букви, то необхідно вставити атрибут WINDOWS-1251, а якщо ні, то XML-файл просто не буде оброблятися.

Наступні рядки – це тіло XML-файла. Вони складаються з елементів, які в сукупності утворюють деревоподібну структуру. Елементи ідентифікуються тегами і можуть бути вкладені один в одного.

Елементи можуть мати атрибути, значення яких теж можуть оброблятися відповідно до шаблону.

Змініть Xml-файл, додавши теги елемента верхнього рівня. Запишіть у блокноті наступну програму.

```
<?xml version=«1.0» encoding=«WINDOWS-1251»?>
<knowledgedatabase>
<tutorial>
<title>«Замітки про XSL»</title>
<author>Сиротенко Сергій Петрович</author>
</tutorial>
<tutorial>
<title>«Введення в CSP»</title>
<author>Сиротенко Сергій Петрович</author>
</tutorial>
</knowledgedatabase>
```

Зазначимо, що імена тегів чутливі до регістру символів. Перейдемо тепер до шаблону перетворення – до XSL-файла. Завдання XSL-файла – перетворити інформацію XML-файла в іншу, яка буде відповідати формату HTML і може бути зображена на екрані браузера з урахуванням форматування, вибору шрифтів тощо.

Для того щоб браузер виконав необхідне перетворення, по- трібно в XML-файлі вказати посилання на XSL-файл:

```
<?xml version=«1.0» encoding=«WINDOWS-1251»?>
<?xml-stylesheet type=‘text/xsl’ href=‘ex01-1.xsl’?>
```

Введіть текст XSL-файла:

```
<xsl:stylesheet version=«1.0»
xmlns:xsl=«http://www.w3.org/TR/Wd-xsl»>
<xsl:template match=«/»>
<p><strong><xsl:value-of select=«//title»«/></strong></p>
<p><xsl:value-of select=«//author»/></p>
</xsl:template>
</xsl:stylesheet>
```

Перший рядок файла містить тег елемента xsl:stylesheet. Атрибути елемента – номера версії і посилання на простір імен. Ці атрибути елемента xsl:stylesheet є обов’язковими. Для XSL-файлів посилання на простір імен є стандартним.

XSL-файл містить елемент верхнього рівня xsl:stylesheet, а далі йдуть правила перетворення.

Синтаксис коментаря такий - <!-- Текст коментаря -->.

5. Створіть файл, який виведе значення атрибута елемента. Введіть такий XML-файл ex02-1.xml

```
<?xml version=«1.0» encoding=«WINDOWS-1251»?>
<?xml-stylesheet type=‘text/xsl’ href=‘ex02-1.xsl’?>
<tutorial>
<dog caption=«Собака: « name=«Кулька»>
<doginfo weight=«18 кг» color=«рудий із чорними підпалинами»/>
</dog>
</tutorial>
```

У цьому файлі інформація зберігається не в змісті елемен- тів, а у вигляді значень атрибутів. Файл ex02-1.xsl має вигляд

```
<xsl:stylesheet version=«1.0»
xmlns:xsl=«http://www.w3.org/TR/Wd-xsl»>
<xsl:template match=«/»>
<P><B><xsl:value-of select=«//dog/@caption»/></B>
```

<xsl:value-of select=«//dog/@name»/>.
<xsl:value-of select=«//doginfo/@weight»/>, <xsl:value-of select=«//doginfo/@color»/>.</P>
</xsl:template>
</xsl:stylesheet>

Зверніть увагу на синтаксис посилання на атрибут елемента – //dog/@name. Ім'я елемента і ім'я атрибута розділені парою символів «/@». В іншому синтаксис той самий, що і для посилання на зміст елемента. Результат має такий вигляд:

Собака: Кулька. 18 кг, рудий з чорними підпалинами.

Зверніть увагу на такий момент. У XSL-файлі немає елемента tutorial. Насправді можна було використовувати повний шлях. Перепишіть XML-файл (ex02-2.xml).

<?xml version=«1.0» encoding=«WINDOWS-1251»?>
<?xml-stylesheet type='text/xsl' href='ex02-2.xsl'?>
<tutorial>
<enimals>
<dog caption=«Собака: « name=«Кулька»>
<doginfo weight=«18 кг» color=«рудий із чорними підпалинами»/>
</dog>
</enimals>
</tutorial>

Файл ex02-2.xsl має вигляд:

<xsl:stylesheet version=«1.0»
xmlns:xsl=«http://www.w3.org/TR/Wd-xsl»>
<xsl:template match=«/»>
<P><B><xsl:value-of select=«//enimals/dog/@caption»/></B>
<xsl:value-of select=«//enimals/dog/@name»/>.
<xsl:value-of select=«//enimals/dog/doginfo/@weight»/>,
<xsl: value-of select=«//doginfo/@color»/>.</P>
</xsl:template>
</xsl:stylesheet>

Результат виводу на екран.

Собака: Кулька. 18 кг, рудий із чорними підпалинами.

Зробіть висновки результатів запиту.

6. Введіть у блокнот такий XML-файл - ex03.xml.

```
<?xml version=«1.0» encoding=«WINDOWS-1251»?>
<tutorial>
<enimals>
<dogs>
<dog>
<dogname>Кулька</dogname>
<dogweight caption=«кг»>18</dogweight>
<dogcolor>рудий із чорними підпалинами</dogcolor>
</dog>
<dog>
<dogname>Тузик</dogname>
<dogweight caption=«кг»>10</dogweight>
<dogcolor>білий із чорними плямами</dogcolor>
</dog>
<dog>
<dogname>Бобик</dogname>
<dogweight caption=«кг»>2</dogweight>
<dogcolor> біло-сірий</dogcolor>
</dog>
<dog>
<dogname>Трезор</dogname>
<dogweight caption=«кг»>25</dogweight>
<dogcolor>чорний</dogcolor>
</dog>
</dogs>
</enimals>
</tutorial>
```

Перший крок – це додавання шаблону перетворення. Модифікуйте файл, додавши в нього посилання на шаблон. У результаті отримаєте файл ex03-1.xml.

Шаблон перетворення ex03-1.xsl.

```
<?xml version=«1.0» encoding=«WINDOWS-1251» ?>
<xsl:stylesheet version=«1.0»
xmlns:xsl=«http://www.w3.org/TR/Wd-xsl»>
<xsl:template match=«/»>
```

<table border=«1»>
<tr bgcolor=«#CCCCCC»>
<td align=«center»>Кличка</td>
<td align=«center»>Вага</td>
<td align=«center»>Колір</td>
</tr>
<xsl:for-each select=«tutorial/enimals/dogs/dog»>
<tr bgcolor=«#F5F5F5»>
<td><xsl:value-of select=«dogname»/></td>
<td align=«right»><xsl:value-of select=«dogweight»/><xsl:value-of select=«dogweight/@caption»/></td>
<td><xsl:value-of select=«dogcolor»/></td>
</tr>
</xsl:for-each>
</table>
</xsl:template>
</xsl:stylesheet>

Перший рядок у XSL-файлі нормально сприймає російські літери. Наступні два рядки шаблону є вже звичними. Такі шість рядків – це рядок, що містить заголовки стовпців таблиці. Десятий рядок:

<xsl:for-each select=«tutorial/enimals/dogs/dog»>

Цей елемент шаблону дозволяє вибрати і переглянути всі групи інформації, повний шлях до яких задається списком тегів «tutorial/enimals/dogs/dog». Зверніть увагу, шлях задається повністю, жоден з тегів вилучити не можна. Далі в комірках таблиці міститься інформація про собак. Змініть інформацію про кличку у файлі ex03-2.xml:

<dogname>
<dognick>Кулька</dognick>
</dogname>

Якщо у відповідному XSL-файлі поставити посилання <xsl:value-of select=«dognick»/>, то у відповідному стовпці ніякої клички не побачите. Посилання повинно бути повним

-

<xsl:value-of select=«dogname/dognick»/>. Правильний результат.

Кличка	Вага	Колір
Кулька	18 кг	рудий із чорними підпалинами
Тузик	10 кг	білий із чорними плямами
Бобик	2 кг	біло-сірий
Трезор	25 кг	чорний

7. Сортування даних у таблиці

У попередніх прикладах порядок рядків у таблиці повністю відповідав групам тегів у Xml-файлі. Цей порядок можна змінювати. Додамо в тег <xsl:for-each select=«tutorial/enimals/ dogs/ dog»> атрибут order-by:

<xsl:for-each select=«tutorial/enimals/dogs/dog» order-by=«dogname»>

Таблиця прийме вигляд (ex03-3.xml, ex03-3.xsl).

Кличка	Вага	Колір
Бобик	2 кг	біло-сірий
Трезор	25 кг	чорний
Тузик	10 кг	білий із чорними плямами
Кулька	18 кг	рудий із чорними підпалинами

Відсортуйте таблицю по стовпцю «Вага». Спочатку спробуйте зробити за аналогією з попереднім прикладом – атрибут order-by=«dogname» змініть на order-by=«dogweight». Результат наведений нижче (ex03-4.xml, ex03-4.xsl).

Кличка	Вага	Колір
Тузик	10 кг	білий із чорними плямами
Кулька	18 кг	рудий із чорними підпалинами
Бобик	2 кг	біло-сірий
Трезор	25 кг	чорний

Таблиця дійсно відсортована по стовпцю «Вага», але це не числове, а строкове сортування. Для того, щоб браузер сприйняв значення як числа, йому необхідно про це сказати – замість order-by=«dogweight» необхідно написати order-by=«number (dogweight)». Змініть дані й отримайте правильний результат

(ex03-5.xml, ex03-5.xsl).

Кличка	Вага	Колір
Бобик	2 кг	біло-сірий
Тузик	10 кг	білий із чорними плямами
Кулька	18 кг	рудий із чорними підпалинами
Трезор	25 кг	чорний

Зробіть сортування по декількох стовпцях. Різні елементи в атрибуті order-by повинні розділятися символом «;» – order-by=«number(dogweight); dogname» (ex03-6.xml, ex03-6.xsl). Таблиця наведена нижче.

Кличка	Вага	Колір
Трезор	10 кг	чорний
Тузик	10 кг	білий із чорними
плямами Бобик	18 кг	біло-сірий
Кулька	18 кг	рудий із чорними підпалинами

8. Елемент XSL:IF – фільтр

Розглянемо тепер способи фільтрації рядків таблиці. Перший приклад використовує старий синтаксис. У ньому умова фільтрації вказується безпосередньо в атрибуті select (ex04-1.xml, ex04-1.xsl).

Нижче наведений рядок, у який внесли необхідні зміни.

xsl:for-each
select=«tutorial/enimals/dogs/dog[dogweightgt10]
«order-by=«number(dogweight);
dogname;»> І таблиця результатів має
вигляд:

Кличка	Вага	Колір
Кулька	18 кг	рудий із чорними підпалинами
Трезор	25 кг	чорний

У таблиці залишилися тільки ті собаки, чия вага перевищує 10 кг, причому першим записано Кульку, чия вага менша.

9. Більш гнучкі можливості надає новий синтаксис (ex04-2.xml, ex04-2.xsl), розгляньте його. Зверніть увагу, у новому син-

таксисі атрибут order-by в елементі xsl:for-each не підтримується, замість нього вставте два елементи xsl:sort.

<xsl:sort order=«ascending» select=«number(dogweight)»/>

<xsl:sort order=«ascending» select=«dogname»/>

Крім того, умову фільтра винесено в окремий елемент xsl:if.

<xsl:if test=«dogweight>10»>

Не забувайте вказувати кінцевий тег елемента xsl:if.

```
<xsl:if test=«dogweight&amp;gt;10»>
<tr bgcolor=«#F5F5F5»>
<td><xsl:value-of select=«dogname»/></td>
<td align=«right»><xsl:value-of select=«dogweight»/><xsl:value-of
select=«dogweight/@caption»/></td>
<td><xsl:value-of select=«dogcolor»/></td>

</tr>
</xsl:if>
```

У цьому прикладі таблиця результатів повністю аналогічна попередній.

Кличка	Вага	Колір
Кулька	18 кг	рудий із чорними підпалинами
Трезор	25 кг	чорний

10. Введіть такий приклад (ex04-3.xml, ex04-3.xsl). У цьому прикладі використовується функція position(), що визначає порядковий номер фрагмента у вихідному Xml-Файлі.

Відповідний елемент xsl:if.

```
<xsl:if
    test=«position()<3»>
```

Результат:

Кличка	Вага	Колір
Кулька	18 кг	рудий із чорними підпалинами
Тузик	10 кг	білий із чорними плямами

Застосуйте використання більш цікавих функцій – start-with (string,startsubstring) і contains(string,anysubstring). Функція start-with(string,startsubstring) перевіряє, чи починається рядок string з підрядка startsubstring. Приклад – ex04-4.xml, ex04-4.xsl).

Синтаксис елемента xsl:if.

```
<xsl:if test=«starts-with($vardogname,$varstartwith)»>
```

У цьому елементі використовуються змінні. Значення змінних були ініційовані раніше

```
<xsl:variable name=«varstartwith»>T</xsl:variable>
```

<xsl:for-each select=«tutorial/enimals/dogs/dog»>
<xsl:variable name=«vardogname»><xsl:value-of select= «dogname»
/></ xsl:variable>

Змінна varstartwith являє собою підрядок, з якої повинні починатися необхідні нам клички. Вона не міняється, тому ініціюється перед циклом. Змінна vardogname містить кличку собаки, вона міняється на кожному кроці циклу і, відповідно, ініціюється в тілі циклу.

Результат:

Кличка	Вага	Колір
Тузик	10 кг	білий із чорними плямами
Трезор	25 кг	чорний

11. Функція contains(string,anysubstring) перевіряє, чи містить рядок string подрядок anysubstring. Приклад – ex04-5.xml, ex04-5.xsl.

Синтаксис елемента xsl:if.

<xsl:if test=«contains($vardogname,$varstartwith)»>

Цей приклад повністю аналогічний попередньому.

Результат:

Кличка	Вага	Колір
Бобик	2 кг	біло-сірий
Трезор	25 кг	чорний

12. Два елементи xsl:if, вкладені один в одного, дають нам ефект оператора AND (ex04-6.xml, ex04-6.xsl).

Відповідний фрагмент Xsl-Файла.

```
<xsl:if test=«dogweight&gt;10»>
<xsl:if test=«dogweight&lt;20»>
...
</xsl:if>
</xsl:if>
```

Результат:

Кличка	Вага	Колір
Кулька	18 кг	рудий із чорними підпалинами

13. Використання оператора OR. Для цього потрібно включити два цикли, у кожному з яких формується своя вибірка (ex04-

7.xml, ex04-7.xsl).

Відповідний фрагмент XSL-файла.

<xsl:for-each select=«tutorial/enimals/dogs/dog»>
<xsl:sort order=«ascending» select=«number(dogweight)»/>
<xsl:if test=«dogweight<10»>
<tr bgcolor=«#F5F5F5»>
<td><xsl:value-of select=«dogname»/></td>
<td align=«right»><xsl:value-of select=«dogweight»/><xsl:value-of select=«dogweight/@caption»/></td>
<td><xsl:value-of select=«dogcolor»/></td>
</tr>
</xsl:if>
</xsl:for-each>
<xsl:for-each select=«tutorial/enimals/dogs/dog»>
<xsl:sort order=«ascending» select=«number(dogweight)»/>
<xsl:if test=«dogweight>15»>
<tr bgcolor=«#F5F5F5»>
<td><xsl:value-of select=«dogname»/></td>
<td align=«right»><xsl:value-of select=«dogweight»/><xsl:value-of select=«dogweight/@caption»/></td>
<td><xsl:value-of select=«dogcolor»/></td>
</tr>
</xsl:if>
</xsl:for-each>

Результат:

Кличка	Вага	Колір
Бобик	2 кг	біло-сірий
Кулька	18 кг	рудий із чорними підпалинами
Трезор	25 кг	чорний

14. Приклад використання елемента xsl:if для поліпшення зовнішнього вигляду таблиці. Заодно використаємо функцію position(). Будемо використовувати цю функцію для того, щоб чергувати колір парних і непарних рядків таблиці (ex04-8.xml, ex04-8.xsl).

Фрагмент XSL-файла, який відповідає за необхідне чер- гування.

<tr>
<xsl:if test=«position() mod 2 = 0»>

<xsl:attribute name=«bgcolor»>#CCCCCC</xsl:attribute>
</xsl:if>

15. Оператор mod дає залишок від ділення на 2. А елемент xsl:attribute дозволяє динамічно підставляти у файл результатів різні атрибути. Це досить потужний елемент.

Таблиця результатів:

Кличка	Вага	Колір
Кулька	18 кг	рудий із чорними підпалинами
Тузик	10 кг	білий із чорними плямами
Бобик	2 кг	біло-сірий
Трезор	25 кг	чорний

16. Динамічне формування атрибутів на прикладі параметрів посилання в тегу <a>. Припустимо тепер, що в кожному рядку таблиці потрібно зробити посилання на деяку сторінку і передати на цю сторінку два параметри – кличку і вагу собаки. Завдання легко вирішується за допомогою елемента xsl:attribute.

Побудуємо спеціальний приклад, обмежимося тільки відповідним фрагментом XSL-файла.

```
<td>
<!-- Create reference to display details. Parameters – Dog Name and
Dog Weight -->
<a target=«_blank»>
<xsl:attribute
name=«href»>Displaydetails.html?dogname=<xsl:value-o
f select=«dogname»/>&amp;amp;dogweight=<xsl:value-of
select=«dogweight»/></xsl:attribute>
<xsl:attribute name=«title»>To view some more details about
<xsl:value-of select=«dogname»/> click to dog name</xsl:attribute>
<xsl:value-of select=«dogname»/>
</a>
</td>
```

У цьому прикладі в клітинці таблиці розміщується посилання на сторінку з докладними описами. Посилання вказується в атрибуті href тегу <a>. Оскільки на сторінку передаються два параметри, значення яких беруться з XML-файла, цей атрибут фор-

мується динамічно. Зверніть також увагу на символ & (амперсанд), що розділяє передані параметри, записується в XSL-файлі у вигляді &. У другому атрибуті потрібна спливаюча підказка (атрибут title), яка з'являється при наведенні курсору миші на посилання. Текст цієї підказки теж змінюється динамічно. Нарешті, статичний атрибут target розміщений безпосередньо в тегу <a>.

17. Об'єднаємо тепер знання XML з можливостями, які надає Javascript. Припустимо, що нам потрібно мати можливість динамічно змінювати сортування стовпців таблиці при клацанні на заголовку того або іншого стовпця.

Як XML-файл візьмемо звичний файл зі списком собак – ex05-1.xml. Створіть три XSL-файли, у кожному з яких буде свій елемент xsl:sort, що задає сортування рядків – ex05-1a.xsl, ex05-1b.xsl, ex05-1c.xsl.

Текст елемента xsl:sort для кожного файла:

```
<xsl:sort order=«ascending» select=«dogname»/>
<xsl:sort order=«ascending» select=«number(dogweight)»
data- type=«number»/>
<xsl:sort order=«ascending» select=«dogcolor»/>
```

Об'єднаємо все це разом. Нижче повністю приведений текст файла ex05-1.htm.

```
<html>
<head>
<script
language=«Javascript»> var
source;
var style;
```

Функція ініціалізації необхідних об'єктів. У цій функції виводиться первісний варіант на екран.

```
function init() {
```

Створюємо об'єкт для файла – джерела даних.

```
source = new Activexobject(«Microsoft.XMLDOM»);
source.async = false;
```

Створюємо об'єкт для файла із шаблоном перетворення (для файлу стилю).

```
style = new
Activexobject(«Microsoft.XMLDOM»); style.async
= false;
```

Завантажуємо записи у файл – джерело даних. Записи беремо з існуючого XML-файла.
source.load(«ex05-1.xml»);

Завантажуємо файл стилю. Первісне сортування – по кольору. style.load(«ex05-1a.xsl»);

Тепер нам потрібно вивести інформацію на екран. Уважно проаналізуйте синтаксис і запам’ятайте його.

```
document.all.item(«xslresult»).innerhtml
= source.transformnode(style);
return true;
}
```

Сортуємо записи по кличці.

```
function orderbynick() {
style.load(«ex05-1a.xsl»);
document.all.item(«xslresult»).innerhtml
= source.transformnode(style);
return true;
}
```

Сортуємо записи за вагою.

```
function orderbyweight() {
style.load(«ex05-1b.xsl»);
document.all.item(«xslresult»).innerhtml
= source.transformnode(style);
return true;
}
```

Сортуємо записи за кольором.

```
function orderbycolor() {
style.load(«ex05-1c.xsl»);
document.all.item(«xslresult»).innerhtml
= source.transformnode(style);
return true;
}
</script>
</head>
```

При завантаженні сторінки створимо всі необхідні об’єкти і виведемо первісний варіант на екран.

<body onload=«init()»>

<!-- Тут буде розміщатися остаточний варіант Html-вмісту -->

</body>
</html>

При клацанні мишею на заголовку стовпця рядки сортуються відповідно до значення в обраному стовпці.

18. Збережіть звіт і виконані завдання у папці студента.

Контрольні завдання (тести)

Тест 1

1. Яка міжнародна організація займається розробкою стандартів для web-середовища?

1. ISO – International Standards Organization.
2. ECMA – European Computer Manufacturers Association.
3. W3 Consortium.

2. Яка мова була основою для HTML (Hyper Text Markup Language)?

1. SGML.
2. XML.
3. DHTML.

3. Які завдання мов розмітки документів і, зокрема, HTML?

1. Створення сценарію.
2. Розмітка тексту.
3. Форматування тексту.

4. Якаметазастосування каскадних таблиць стилів (Cascading Style Sheets, CSS)?

1. Створення сценарію.
2. Розмітка тексту.
3. Форматування тексту.

5. Який стандарт HTML створений на основі XML?

1. SGML.
2. XHTML.
3. DHTML.

6. Які завдання клієнтських мов, зокрема, Javascript?

1. Створення сценарію.
2. Розмітка тексту.
3. Форматування тексту.

7. Які з перерахованих мов використовуються для напи- сання сценаріїв?

1. Basic.
2. Javascript.
3. Delphi.
4. Vbscript.

8. Чи допускає стандартxhtml-докумен та використання непарних дескрипторів?

1. Так.
2. Ні.

9. Яка з мов розмітки є чутливою до регістра символів?

1. HTML.
2. XHTML.

10. Яка основна функція браузера?

1. Інтерпретація і представлення html-документа.
2. Форматування html-документа.
3. Створення сценарію.

Тест 2

1. Яка мета вживання мови HTML?

1. Створення сценарію.
2. Розмітка тексту.
3. Форматування тексту.

2. Які конструкції мови HTML, що управляють?

1. Оператори.
2. Дескриптори (теги).
3. Процедури.

3. Яким чином можна уточнити дію дескриптора?

1. За допомогою атрибутів.
2. За допомогою приміток.
3. За допомогою вкладеного дескриптора.

4. Якими дескрипторами html-документа обмежена область заголовка?

1. <body>.</body>.

2. <html>.</html>.
3. <head>.</head>.

5. Якими дескрипторами html-документа обмежена область «тіла» документа?

1. <body>.</body>.

2. <html>.</html>.
3. <head>.</head>.

6. Якими дескрипторами html-документа обмежений заголовок документа, що відображається у браузері?

1. <body>.</body>.

2. <title>.</title>.
3. <head>.</head>.

7. У якій частині html-документа розташовані упроваджені стильові описи і скрипти?

1. <body>.</body>.

2. <html>.</html>.
3. <head>.</head>.

8. Який із запропонованих варіантів вкладення дескри- пторів є правильним?

1. <A>.

2. <A>.

9. Який з перерахованих програмних продуктів використовується для візуальної компоновки web-сторінки (є wysiwyg-редактором)?

1. Notepad.

2. Dreamweaver.
3. Homesite.

Який з перерахованих програмних продуктів є спеціалі-

зованим (не WYSIWYG) html-редактором?

1. Notepad.

2. Dreamweaver.
3. HomeSite.

Тест 3

1. Яке призначення каскадних таблиць стилів (CSS)?

1. Опис параметрів оформлення документа.
2. Опис структури документа.
3. Створення сценарію.

2. Вкажіть три способи вживання таблиць стилів у документах HTML.

1. Оголошення, скріплення і вбудовування.

2. Вбудовування, впровадження і скріплення.
3. Додавання, впровадження і вставка.

3. У якій частині html-документа розташовується опис формату при вбудовуванні стилів у документ?

1. В елементах.

2. Усередині блоку <style>.
3. У зовнішньому файлі з розширенням **css.**

4. У якій частині html-документа розташовується опис формату при впровадженні стилів у документ?

1. В елементах.

2. Усередині блоку <style>.
3. У зовнішньому файлі з розширенням css.

5. Де розташовується опис формату при використанні скріплення?

1. В елементах.

2. Усередині блоку <style>.
3. У зовнішньому файлі з розширенням css.

6. З якою метою при створенні стилів застосовуються класи?

1. Для створення декількох наборів стилів для одного елементу HTML.

2. Для створення одного стилю для декількох елементів HTML.

7. З якою метою при створенні стилів застосовується групування?

1. Для створення декількох наборів стилів для одного елементу HTML.
2. Для створення одного стилю для декількох елементів HTML.

8. Що є точкою відліку при абсолютному позиціюванні об'єктів?

1. Верхній лівий кут вікна браузера.

2. Зовнішній блок.

9. Які фільтри міняють вигляд об'єкта, залишаючи при цьому картину нерухомою?

1. Статичні.

2. Динамічні.

10. Які фільтри дозволяють спостерігати плавний пере- хід від одного стану об'єкта до іншого?

1. Статичні.

2. Динамічні.

Тест 4

1. Яке призначення Об'єктної моделі браузера (BOM)?

1. Розмітка документа.
2. Надання сценарію Javascript доступу до елементів браузера.
3. Створення сценарію.

2. Який об'єкт є старшим в ієрархії об'єктів браузера?

1. Window.
2. Screen.
3. Document.

3. Який об'єкт відтворює html-документ у браузері?

1. Window.

2. Screen.
3. Document.

4. Який дочірній об’єкт window дозволяє отримати інфор- мацію про тип і версію браузера?

1. Location.

2. Navigator.
3. History.

5. Який дочірній об’єкт window дозволяє отримати інформацію про адресу ресурсу, завантаженого в браузер?

1. Location.

2. Navigator.
3. History.

6. Яка з приведених конструкцій дозволяє звернутися до заданого (і-му) елементу масиву графічних зображень, розташованих на сторінці?

1. document.images[i].

2. image(i).
3. document.image[i].jpg.

7. Яка властивість об’єкта window дозволяє визначити дату останньої модифікації сторінки?

1. status.

2. lastmodified.
3. name.

8. Яка властивість об’єкта window дозволяє вивести інформацію в статусний рядок?

1. status.

2. lastmodified.
3. name.

9. Який об’єкт дозволяє аналізувати властивості браузера?

1. Window.

2. Navigator.
3. History.

10. Який метод об'єкта window дозволяє відкрити нове вікно і вказати його параметри?

1. Open.
2. Focus.
3. Moveto.

Тест 5

1. Таблиця каскадних стилів – це:

а) файл з описом структури веб-сторінки;

б) певна структура з описом властивостей елементів веб-сторінки;

в) спеціальна структура, в якій задається колір шрифту для елементів веб-сторінки;

г) набір правил оформлення та форматування, які застосовують до різних елементів веб-сторінки;

д) файл із призначенням кольору для веб-сторінки.

2. Зовнішня таблиця стилів – це:

а) файл із розширенням .html з описом властивостей елементів веб-сторінки;

б) файл із розширенням .txt з описом властивостей елементів веб-сторінки;

в) файл із розширенням .css із описом властивостей елементів веб-сторінки;

г) опис властивостей елементів веб-сторінки в окремому файлі, який можна використовувати для багатьох веб-сторінок;

д) веб-сторінка з різними настройками функціональних розділів.

3. Внутрішня таблиця стилів:

а) розміщується безпосередньо в розділі <HEAD> усередині блоку, що охоплений тегами <STYLE>...</STYLE>;

б) розміщується безпосередньо в розділі <HEAD>;

в) розміщується у будь-якому місці HTML-документа у фігурних дужках;

г) це файл із розширенням .css із описом властивостей еле- ментів веб-сторінки;

д) це файл з описом структури веб-сторінки,

4. Вбудований стиль – це:

а) перелік параметрів форматування елемента веб-сторінки, заданий у його тегу за допомогою атрибута STYLE;

б) опис властивостей фрагмента тексту чи об’єкта, розміще- ний у фігурних дужках у будь-якому місці HTML-документа;

в) опис властивостей елемента веб-сторінки в її тілі;

г) файл із розширенням .html з описом властивостей елемен- тів веб-сторінки;

д) файл з описом структури веб-сторінки.

5. Сценарій – це:

а) список гіперпосилань веб-сайту;

б) програма, написана спеціальною мовою програмування і вбудована в HTML-документ;

в) послідовність відображення веб-сторінок під час перегля- ду сайта;

г) програма, яку використовують для динамічного оновлен- ня веб-сторінки;

д) програма будь-якою мовою програмування, розміщена на веб-сторінці.

6. JavaScript – це:

а) певна структура, що описує властивості елементів веб- сторінки;

б) файл із розширенням .html з описом властивостей елемен- тів веб-сторінки;

в) мова програмування, яка дає змогу вбудовувати викону- вані модулі в документи, написані мовою HTML;

г) програма, що дає змогу імітувати рух зображень на веб- сторінці;

д) засіб для створення інтерактивних веб-сторінок.

7. Об’єктна модель – це:

а) структура, що дозволяє описати властивості елементів веб-сторінки;

б) подання об’єктів, властивостей і подій у вигляді, зручно- му для роботи з ними в кодах HTML та у сценаріях;

в) спосіб взаємодії між HTML-кодом веб-сторінки та користувачем;

г) засіб для роботи зі структурою документа;

д) основа для створення динамічно керованих веб-сторінок.

8. Об’єкт Window:

а) це основний контейнер, у якому розміщується все те, чим можна керувати за допомогою браузера;

б) керує інформацією, що міститься у видимій на екрані частині веб-сторінки;

в) надає всю інформацію про HTML-документ за допомогою колекцій і властивостей;

г) це об’єкт найвищого ієрархічного рівня, в який завантажуються документи;

д) надає можливість для керування вмістом та розмірами вікна під час відтворення веб-сторінки у браузері.

9. Об’єкт Document:

а) керує інформацією, що міститься у видимій на екрані частині веб-сторінки;

б) надає всю інформацію про HTML-документ за допомо- гою колекцій і властивостей;

в) надає всю інформацію про HTML-документ, а також методи та події для роботи з ним;

г) дає змогу динамічно формувати вміст сторінки у процесі її завантаження;

д) це основний контейнер, у якому розміщується все те, чим можна керувати за допомогою браузера.

10. Тег <FORM>...</FORM>:

а) інформує про використання JavaScript або іншої мови веб-сценаріїв у тілі HTML-документа;

б) визначає атрибути кнопок, розташованих на веб-сторінці;

в) форматує певну частину тексту;

г) містить теги полів для введення даних, перемикачів, прапорців, кнопок;

д) містить атрибути, що надають можливість надсилати дані на сервер для подальшого опрацювання.

11. Тегом кнопки надсилання даних, введених у форму, є:

а) <INPUT TYPE=«reset» ...>;

б) <INPUT TYPE=«submit»
...>; в) <INPUT TYPE=«goto»
...>; г) <INPUT TYPE=«select»
...>; д) <INPUT
TYPE=«radio»...>.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Глинський Я. М., Інтернет. Сервіси, HTML і Web-дизайн: навч. посіб. / Я. М. Глинський, В. А. Ряжська. – [2-ге вид., доп.]. – Львів: Деол; СПД Глинський, 2003. – 192 с.

2. Глушаков С. В. Программирование Web-страниц / Глушаков С. В., Жакин И. А., Хачиров Т. С. – Харьков: Фолио, 2005. – 390 с.

3. Корпоративные порталы на основе XML и Web-служб / под. ред. д.т.н., проф. А. Д. Иванникова. – Москва: КУДИЦ-ОБРАЗ, 2004. – 368 с.

4. Дунаев В. В. Основы WEB-дизайна. Самоучитель / Дунаев В. В. – СПб.: «БХВ-Петербург», 2006. – 512 с.

5. Когзолл Дж. PHP 5. Полное руководство / Дж. Когзолл. – Москва: Вильямс, 2006. – 752 с.

6. Колисниченко Д. Н. Самоучитель PHP 5 / Колисниченко Д. Н. – [изд. 3-е.]. – СПб.: Наука и техника, 2006. – 576 с.

7. PHP 5 и MySQL. Библия пользователя / [Конверс, Тим и др.]; пер. с англ. – Москва: Изд-й дом «Вильямс», 2006. – 1216 с.

8. Крамер Э. HTML: наглядный курс Web-дизайна / Э. Кра- мер. – Москва: Изд. дом «Вильямс», 2001. – 304 с.

9. Кузнецов М. В. PHP 5. Практика создания WEB-сайтов / Кузнецов М. В., Симдянов И. В., Голышев С. В. – СПб.: «БХВ-Петербург», 2006. – 960 с.

10. Леонтьев Б. Web-дизайн. Тонкости, хитрости и секреты / Б. Леонтьев. – Москва : СОЛОН-Пресс, 2003. – 640 с.

11. PHP 5 для начинающих / [Мерсер Д. У., Кент А., Новиц- кий С. та ін]; пер. с англ. – Москва: Вильямс, 2006. – 848 с.

12. Мэрдок К. Л. Java Script: наглядный курс создания динамических Web-страниц / Мэрдок К. Л. – Москва: Изд. дом «Вильямс», 2001. – 288 с.

13. Нидерст Дж. Web-мастеринг для профессионалов. Настольный справочник / Дж. Нидерст. – СПб.: Питер, 2001. – 576 с.

14. Питтс Н. XML за рекордное время / Н. Питтс. – Москва: Мир, 2000. – 444 с.

15. Старыгин А. XML: разработка Web-приложений / А. Ста- рыгин. – СПб.: БХВ-Петербург, 2003. – 592 с.

16. Хольцшлаг М. Языки HTML и CSS: для создания Web-сайтов: Официальный учебный курс / М. Хольцшлаг. – Москва: «издательство ТРИУМФ», 2006. –304 с.

17. Храмцов П. Б. Основы WEB-технологий: [учебное пособие] / П. Б. Храмцов, С. А. Брик, А. М. Русак, А. Н. Сурин. – [2-ге изд., испр.]. – Москва: Интернет-университет информац. технолог., БИНОМ, Лаборатория знаний, 2007. – 374 с.

18. Будилов В. А. PHP-5. Экспресс-курс / Будилов В. А. – СПб.: БХП-Петербург, 2005. – 240 с.

19. Антоненко В. М. Сучасні Internet технології: курс лекцій та лабораторний практикум / Антоненко В. М., Терейковський І. А., Терейковська Л. О. Частина І. Основи Web-дизайну. – Ірпінь: Академія ДПС України, 2007. – 232 с.

20. Інтернет для користувача: [навч. посібник] / Антонен- ко В. М., Пацай Б. Д., Терейковський І. А., Терейковська Л. О. – Ірпінь: Академія ДПС України, 2010. – 246 с.