МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

О.В. Цеслів

Основи програмування та веб-дизайн для студентів економічних спеціальностей

Навчальний посібник

Київ
2020

УДК 004.738.5(075.8)+330.46

Рецензенти:

О.Ю.Чубукова, завідувач кафедри “Економічної кібернетики”, Київського національного університету технології та дизайну, д.е.н., професор.

О.І.Стасюк, завідувач кафедри “Автоматизація та комп'ютерно-інтегровані технології транспорту”, Київський університет економіки і технології транспорту, д.т.н., професор.

Відповідальний

редактор **В.О. Капустян**, проф.д.ф-м.наук

Цеслів О.В.

Основи програмування та веб-дизайн: Навч. посіб. –К.,2020 –149с.

Даний посібник сформований за результатами викладання курсу лекцій студентам першого курса факультету менеджменту та маркетингу Національного технічного університету України «Київський політехнічний інститут». Розглянуто основи веб - програмування. Розглядається мова розмічування веб сторінки HTML, каскадні таблиці стилів CSS, мова програмування JavaScript.

ЗМІСТ

ВСТУП

На сучасному етапі розвитку ринкової економіки під впливом нових інформаційних технологій відбуваються докорінні зміни в технології управління. Компанії не представляють своєї роботи без Internet, електронної пошти, FTP, WWW. Всесвітня павутина дозволяє оперативно отримати необхідну інформацію про діяльність компанії. Звичайно, ця інформація повинна оперативно обновлюватися. В зв'язку з цим важливим завданням вищої школи слід вважати випуск спеціалістів, які можуть поєднувати знання економічних спеціальностей з широким використанням інформаційних технологій на практиці.

На сьогоднішній день опубліковано багато праць, присвячених практичному створенню WEB - сайтів [1-5]. Водночас майже відсутні роботи методологічного плану, навчальні посібники, які б систематизували теоретичні і практичні результати.

Даний посібник сформований за результатами викладання курсу лекцій студентам старших курсів факультету менеджменту та маркетингу , "Київський політехнічний інститут імені Ігоря Сікорського".

Посібник орієнтований на студентів спеціальності 075 Маркетинг, при вивченні дисципліни "Основи програмування та веб-дизайн". Розглядається мова розмічування веб сторінки HTML, каскадні таблиці стилів CSS, мова JavaScript. Автор сподівається, що матеріали посібника будуть корисними для студентів та аспірантів інших спеціальностей, які цікавляться сучасними інформаційними технологіями та їх практичним застосуванням.

Навчальний посібник складається з восьми розділів, які слід вивчати послідовно. Перший розділ містить основні відомості про мову розмітки сторінок HTML, каскадні таблиці стилів CSS.

У другому розділі ми приступимо до вивчення базових елементів мови JavaScript, тут описуються типи змінних,логічні операції, методи confirm, alert, prompt. Третій розділ присвячений умовним операторам. В четвертому

розділі розглянуті цикли. Як працювати з масивами, ви довідаєтеся в п’ятому розділі.

У шостому розділі розглядаються об’єкти. Як працювати з об'єктною моделлю документа, ви довідаєтеся в сьомому розділі. Публікація сайту на сервері описана в восьмому розділі.

Автор висловлює глибоку вдячність рецензентам —д.е.н., професору О.Ю. Чубуковій, д.т.н., професору О.І. Стасюку за цінні поради, які сприяли покращенню посібника.

РОЗДІЛ 1. МОВА HTML

Ідея рішення проблеми обміну документами між різними комп'ютерами й додатками через Internet заснована на мові розмічування гіпертексту HTML (HyperText Markup Language). Ця мова була створений більше 15 років тому, як стандарт оформлення документів і була прийнята переважною більшістю користувачів Internet, а головне, - всіма виробниками програмного забезпечення й устаткування для Web. Документи, розмічені згідно HTML, можуть читатися на будь-якому комп'ютері, на якому встановлена всього лише одна програма перегляду таких документів - браузер.

Завдяки мові розмітки HTML клієнт Web може на екрані свого комп'ютера переглянути документ у тому вигляді, у якому його задумав розроблювач: з певними розмірами шрифту й розбивкою на абзаци, з потрібним розташуванням малюнків, гіперпосилань. Текстової документ, складений на HTML, має розмір у байтах у кілька разів менший, чим розмір аналогічного документа, підготовленого в текстовому процесорі (наприклад, Word).

Розмітка документа — це опис різних фрагментів документа і їхнього взаємного розташування. Виконується розмітка за допомогою символів ASCII, а точніше, арабських цифр, символів латинського алфавіту й деяких розділових знаків. Із цих символів набираються команди мови HTML - теги, або, інакше кажучи, дескриптори.

У мові HTML є безліч тегів, серед яких - теги створення заголовка документа, завдання параметрів шрифту, креслення ліній, вставки гіперпосилань, графічних елементів, звуків, відео.

Історія мов розмітки почалася в 60-і роки 20 століття, коли співробітники компанії IBM взялися за рішення завдань перенесення документів між різними платформами й операційними системами. Результатом їхніх зусиль стала мова GML (General Markup Language - загальна мова розмітки), яка призначалася для використання на ЕОМ

сімейства IBM. Мова GML надалі була розширена, а в 80-х роках пройшла стандартизацію в ISO (Міжнародна організація стандартизації). Ця потужна й універсальна мова розмітки, яка отримала назву SGML (Standard General Markup Language), використовувалася військовим відомством США при оформленні технічної документації. Однак SGML широкого поширення не одержала через свою складність і високу вартість.

Наступний етап розвитку мов розмітки пов'язаний з іменами вчених-фізиків, співробітників CERN у Женеві. Так, наприкінці 80-х років Тім Бернерс-Лі зайнявся проблемою зберігання й відображення даних, отриманих колегами. В основу нової мови Бернерс-Лі поклав мову SGML і прийоми роботи з гіпертекстом, так появилася мова - HTML.

Можливість застосування гіпертексту була запропонована Ваневаром Бушем в 1945 році, а термін «гіпертекст» уперше був введений Тедом Нельсоном в 60-х роках, тобто задовго до появи Internet. Поняття «гіпертекст» позначає електронний документ, що містить у собі посилання на інші документи.

Створення HTML привело до появи нової технології поширення гіпертекстових документів в Internet. Однак для широкого впровадження World Wide Web, крім мови HTML, потрібна була розробка протоколу передачі гіпертексту HTTP (HyperText Trasport Protocol - протокол передачі гіпертексту), що дозволив здійснювати обмін документами HTML. Саме цей протокол дав можливість додатку-клієнта знаходити й використовувати ресурси, що зберігаються на іншому комп'ютері.

Стандарт HTML 4.01. був затверджений в 1999 році. У посібнику ми будемо дотримуватися рекомендацій консорціуму W3C по застосуванню HTML-елементів і використанню каскадних таблиць стилів.

Гіпертекст — це електронний документ, який містить гіперпосилання на інші документи. ***Гіпертекстова технологія*** це інформаційна технологія, що базується на використанні гіпертекстів.

Декілька web-документів на одну тему, що є на деякому комп'ютері чи належать одному власникові, утворюють - web-сайт.

Web-дизайн - це сукупність правил і рекомендацій, якими мають користуватися автори, якщо вони хочуть, щоб їхні сторінки були інформативними і виглядали привабливо.

Одне з найважливіших правил web-дизайну полягає у структуризації інформації, вдалому поділі її на окремі частини і налагодженні зв'язків між ними. Згідно з чинним стандартом абзаци на класичній web-сторінці відокремлюються порожнім рядком і не мають відступів у першому рядку.

Сучасні редактори, такі як FrontPage, Dreamweaver, MS Word тощо, дають змогу створювати WEB - сторінки методом конструювання, тобто без застосував кодів мови HTML (Hyper Text Markup Language). Вони генерують цей код автоматично.

Для підготовки html-файлу можна використати текстовий редактор NotePad(Блокнот), що дає змогу готувати файли у текстовому форматі. Після написання html-файл потрібно зберегти його на диску з деякою назвою з розширенням назви html.

1.1.Тегова модель файлу

Команди мови HTML називаються *тегами.* Теги бувають одинарними і парними. Більшість тегів парні, як, наприклад, тег означення html-файлу: <HTML> ... </HTML>. Такі теги називаються інакше *контейнерами.* Контейнер може містити текст та інші теги.

Парні теги позначають початок і кінець області дії відповідної команди. Теги записують у кутових дужках. Тег, що закриває область дії, має косу риску. Не забувайте її писати, інакше тег працюватиме неправильно.

Тег може містити параметри, які користувач записує у першому блоці тега через пропуск чи з нового рядка, наприклад:

<BODY TEXT=”red”>...</BODY>.

Нечислові значення параметрів прийнято записувати у лапках.

У середині пари тегів <HEAD>...</HEAD> описують заголовок документа. Головна частина заголовка документа заголовок Windows-вікна, який пишуть у середині пари тегів <TITLE>...</TITLE>. Інші елементи заголовка вивчатимемо пізніше.

Тег <!— текст —> позначає коментар. Текст у цього тега виводитися на екран не буде. Коментар можна записати також у середині парного тега <COMMENT> текстовий коментар </COMMENT>.У середині пари тегів <BODY параметри >...</BODY> записують увесь текст сторінки. Цей текст відображатиметься у вікні браузера. Розглянемо головні параметри тега BODY:

Таблиця 1.1. **Параметри тега BODY**

BACKGROUND = "d:\myweb\1.bmp"	Задає шлях до картинки для тла.
BGCOLOR = "white"	Задає білий колір тла, якщо не використо-вується тло – картинка.
BGPROPERTIES="fixed"	Фонове зображення не прокручується.
TEXT = "black"	Задає колір тексту (тут чорний) на сторінці.

1.1.1. Форматування тексту для web-сторінок

Розглянемо теги, які використовують для форматування тексту. Теги форматування символів тексту (вони парні):

Таблиця 1.2. **Теги форматування символів тексту**

<B> текст </B>	Товстий шрифт тексту.
<I> текст </I>	Шрифт- *курсив.*
<U> текст </U>	Підкреслений шрифт.
_{текст}	Нижній індекс. Наприклад, щоб отримати вираз H_2O_2, пишуть H₂O₂.

^{текст}	Верхній індекс, наприклад, I^a, a^2.
<BIG> текст </BIG>	ВЕЛИКИЙ шрифт.
<SMALL>текст </SMALL>	Малий шрифт.
<EM> текст </EM>	Виокремлений *курсивом* текст (те саме, що тег <I>).
<B><I>текст</I></B>	*Товстий курсив.* Цей приклад демонструє застосування принципу вкладення тегів.

Теги для розміщення тексту (вони одинарні):

<P> Цей тег означає початок нового абзацу.

 Наступний за цим тегом текст буде наведено у новому рядку без пропуску рядка.

<HR> У рядку буде проведена горизонтальна лінія.

Заголовки і теги вирівнювання. Заголовок - окремий абзац. Є шість видів заголовків, які відрізняються розмірами символів:

Таблиця 1.3.**Теги заголовків**

Теги	Результат на екрані
<H1>Заголовок 1</H1>	Заголовок 1
<H2>Заголовок 2</H2>	Заголовок 2
<H3>Заголовок 3</H3>	Заголовок 3
<H4>Заголовок 4</H4>	Заголовок 4
<H5>Заголовок 5</H5>	Заголовок 5
<H6>Заголовок 6</H6>	Заголовок 6

Заголовок за замовчуванням вирівнюється до лівого краю вікна. Якщо вирівнювання заголовка чи іншого елемента *N* сторінці потрібно задати явно, то використовують теги вирівнювання:

Таблиця 1.4. **Теги вирівнювання тексту**

<CENTER> елемент </CENTER>	Вирівнювання до центру
<LEFT> елемент </LEFT>	Вирівнювання до лівого краю

<RIGHT> елемент </RIGHT>	Вирівнювання до правого краю

Ненумерований список утворюють за допомогою наступних тегів

<LH> Назва списку </LH>

<UL>

<LI>елемент 1

<LI>елемент 2

</UL>

Нумерований список створюють за допомогою парного тега <OL>... </OL> з необов'язковим параметром TYPE і одинарних тегів <LI> так:

<LH> Назва списку </LH>

<OL TYPE="1">

<LI> елемент 1

<LI> елемент2

</OL>

Значення "i" чи "I" параметра TYPE задає нумерацію римськими малими (i, ii, iii, iv, ...) чи великими (I, II, III, IV, ...) цифрами, а значення "a" чи "A" - латинськими малими (a, b, c, d, ...) чи великими (A, B, C, ...) літерами.

Список тлумачень використовують для пояснення термінів, створення словників тощо. Його утворюють за допомогою парного тега <DL>...</DL> і двох одинарних тегів <DT>, <DD> так:

<LH>Заголовок</LH>

<DL>

<DT> термін

<DD> тлумачення 1

<DD> тлумачення 2

</DL>

Приклад 1.1. Створити файл з назвою file1.html.

<HTML><HEAD>

<tItle> Мій факультет</tItle>

<B> <H1>Факультет менеджменту та маркетингу </H1> </CENTER> </B></HEAD>

<Body Bgcolor="yellow"> (Декан - проф.<B><I>Гавриш Олег Анатолійович </B></I>)

<small> <I>Навчальний корпус 1, кімн. 246</I> </small>

<small> <I>Телефон для довідок: 441-18-86</I> </small>

<p>
<I><LH>ЕКОНОМІКА І ПІДПРИЄМНИЦТВО</I> </LH></BR></P> </CENTER>

<UL>

<LI>Економіка підприємства

<LI>Економічна кібернетика

<LI>Міжнародна економіка

<LI>Маркетинг

</UL>

<LH>
<I>МЕНЕДЖМЕНТ</I></BR> </LH>

<UL>

<LI>Менеджмент організацій

<LI>Менеджмент зовнішньоекономічної діяльності

</UL>

<UL><left>

<LI>Факультет здійснює підготовку висококваліфікованих фахівців з менеджменту, економіки і підприємництва, які здатні приймати

обґрунтовані рішення з управління персоналом, ринком, виробництвом, зовнішньоекономічною діяльністю та спроможні забезпечити ефективне функціонування національної економіки в умовах ринку.

<LI>Термін підготовки: <I>бакалавр</I>-4 роки;

<I>спеціаліст</I>-5років;<I>магістр</I>-5 років.

<LI>Випускники обіймають керівні посади в державних органах, підприємствах та фірмах, фінансових, страхових, консалтингових структурах ринкової економіки.

<LI>Абітуріенти складають вступні випробування з <B>математики, економічної і соціальної географії України, української мови та літератури.</B></UL></Body></HTML>

1.2. Створення таблиць

У звичайних текстових редакторах таблиці використовують для наочного подання числової чи текстової інформації. У web-дизайні таблиці відіграють велику роль. Часто їх використовують для позиціювання графічних чи інших об’єктів на екрані. Такі таблиці утворюють невидимими межами (рамками), а в клітинках розташовують картинки, тексти тощо.

Таблиці створюють за допомогою таких тегів:

<TABLE параметри>

<TC> Заголовок таблиці </TC>

Тут пишемо теги для заповнення клітинок таблиці рядок за рядком

</TABLE>

Для заповнення клітинок таблиці використовують такі теги (закриваючі теги можна не зазначати):

Таблиця 1.5. **Теги заповнення клітинок таблиці**

<TR>...</TR>	Формують рядок таблиці.
<TH>текст</TH>	Формують клітинку з заголовком рядка чи стовпця
<TB>текст</TB>	Формують текст кожної клітинки.

Заголовки рядків і стовпців виводитимуться товстішим шрифтом

Приклад 1.2. Створити файл з назвою file2.html

```
<HTML><HEAD>
<TITLE>Студентська поліклініка "КПІ"</TITLE></HEAD>
<BODY bgcolor="silvery">
<center><H2><b>
<LH><font color="red">Студентська поліклініка НТУУ
    КПІ</font></LH></b></H2></center></font>
<center>
<TABLE border=3 bgcolor="yellow" bordercolor="BLACK">
<TC><center><h3><b><LH><I><font color="red">Графік роботи лікарів факультетів (інститутів)</font></I></LH></b></h3></center></TC>
<TR><H6>
<TH>Факультет</TH>
<TH>Прізвище, ім'я, по-батькові</TH>
<TH>каб</TH>
<TH>Режим роботи по днях</TH>
</H6></TR>
<TR><B><TH align="left">ВІТІ, ФЕА, ФПМ, ІПСА</TH></B>
<TD>БОГАЧЕВА Лариса Корніївна</TD>
<TD>3-05</TD>
<TD><BR>непарні 09.00-14.00</BR><BR>парні 14.00-19.00</BR></TD>
</TR>
<TR><B><TH align="left">ІТС, ЗФ, ММІ, ІХФ</TH></B>
<TD>ДМИТРЕНКО Марія Броніславівна</TD>
<TD>3-10</TD>
<TD><BR>непарні 09.00-14.00</BR><BR>парні 14.00-19.00</BR></TD>
```

```
</TR>
<TR><b><TH align="left">ФАКС, ФТІ, ФЛ, ФЕЛ, ІФФ</TH></b>
<TD>ВОРОНА Едуард Миколайович</TD>
<TD>3-07</TD>
<TD><BR>непарні 14.00-19.00</BR><BR>парні 09.00-
14.00</BR></TD>
</TR>
<TR><b><TH align="left">ФММ, ФП, ІЕЕ, ПБФ, Іноземні студенти
</TH></b>
<TD>ШИРАНТ Наталія Петрівна</TD>
<TD>3-07</TD>
<TD><BR>непарні 09.00-14.00</BR><BR>парні 09.00-
14.00</BR></TD>
</TR></TABLE></center></BODY></HTML>
```

1.3. Графічні об'єкти і гіперпосилання

Вставляння графічних і відео файлів. Графічні зображення, такі як фотографії, картинки, піктограми тощо, зберігаються на серверах в окремих файлах з розширеннями bmp, та іншими і відображаються на web-сторінці за допомогою тега <IMG> з параметрами:

```
<IMG  SRC="адреса графічного файлу" ALT = "альтернативний текст"
ALIGN="left" WIDTH=240 HEIGHT=200>
```

Дія тега. Обов'язковим є лише перший параметр SRC. Альтернативний текст - це текст, який виводитиметься на місці картинки, якщо браузер не може прийняти графічний файл або якщо режим відображення графіки вимкнено. ALIGN задає місце розташування картинки на екрані, а параметри WIDTH і HEIGHT - її розміри за шириною і висотою в пікселях або відсотках. Зображення можна подати в рамці. Графічний об’єкт можна створити в Paint.

1.3.1 Вставляння звуку і відеозображення

Важливо пам'ятати, що звукові файли мають розширення: au, wav, mid, midi, ra, а відеофайли - avi, vivo, mpeg. Щоб вставити звук чи відео, достатньо як значення параметра HREF у тезі гіперпосилання задати шлях до відповідного звукового чи відеофайлу, який вже є на диску, наприклад, так:

<A HREF="sound.wav">прослухайте лекцію </A>.

Текст прослухайте лекцію стане гіперпосиланням, клацнувши на якому можна почути лекцію, деяку інформацію, яка була заздалегідь записана.

Приклад 1.3. Створити файл з назвою file3.html

```
<html><head>
<title>Карта "КПІ"        </title></head>
<body bgcolor="Blue">
<font color="Yellow">
<center><h1>Карта НТУУ "КПІ" </h1><hr></center>
</font>
<center><IMG src="picture/map.gif" width="650" height="550" border="1"></center>
</body></html>
```

1.4. Фрейми

Поняття про фрейми. Якщо матеріали web-сайту структуризовані логічно за темами і мають базуватися на декількох сторінках з навігацією за допомогою гіперпосилань, то такий сайт реалізовують із застосуванням фреймів. Фрейм у перекладі з англійської означає кадр, рамка, прямокутна область. Фрейми поділяють вікно браузера на частини, в яких відображають зміст сторінок сайту. Кожній сторінці має відповідати свій html-файл. Кожна сторінка повинна мати логічний заголовок.

1.4.1 Файлова структура сайту з фреймами

Для створення із застосуванням фреймів потрібно скласти як мінімум три html-файли: один основний і два чи більше допоміжні. Один допоміжний

файл потрібний для заповнення стартовою інформацією лівого фрейму, інший – правого.

Гіперпосилання вставляють за допомогою парного тега <A параметр>...</A>, де параметр HREF = "адреса файлу". Гіперпосиланням може бути текст або картинка. Розглянемо випадок, коли гіперпосиланням є текст. Нехай у реченні "Я навчаюсь в КПІ" слово "КПІ" потрібно зробити гіперпосиланням на файл "file1.htm", який містить додаткові відомості про КПІ. Це роблять так:

Я навчаюсь в <A HREF = "file1.html"> КПІ </A>.

Приклад 1.4. Створити файл з назвою index.html

```
<HTML><HEAD>
<title>Це мій сайт з фреймами </title></HEAD>
<FRAMESET COLS="25% ,75% ">
 <FRAME SRC="leftframe.html"
NAME= "left"
<!--або "лівий" або інша назва фрейму-->
SCROLLING="no"
<!--або "yes" або "auto"-->
FRAMEBORDER="1"
<!--або "О" межа фреймів є чи ні-->
BORDER = "15"  <!--товщина межі->
MARGINHIGHT= "10"
<!--  відступи від країв->
MARGINWIDTH=" 10"
NORESIZE
<!--не можна пересувати межу-->
BORDERCOLOR = "red" >
 <FRAME SRC = "rightframe.html"
NAME="right" <!--або "правий" тощо-->
 </FRAMESET>
```

<NOFRAME>

<!--Текст, що відображатиметься у браузері, який не підтримує фреймів, наприклад: >

Вибачте, цей сайт містить фрейми. Скористайтесь іншим браузером для його перегляду. </NOFRAME></HTML>.

Допоміжний rightframe.html

```
<html><head><title>Це мій правий фрейм</title></head>
<body bgcolor="Blue">
<center><IMG src="picture/logo.gif" width="500" height="70"
    border="1"> </center>
<font color="Yellow">
<center><h1>НТУУ "КПІ" запрошує</h1><hr></center></font>
</body></html>
```

Допоміжний leftframe.html

```
<html><head>
      <title>Це мій лівий фрейм</title></head>
<body bgcolor="Yellow">
<font color="Blue">
<H2>Сайт НТУУ "КПІ"</H2></font><br><br>
<a href ="file5.html" target ="right">
<IMG src="picture/1.gif" width="60" height="100"border="0"></a>
<br><a href ="file1.html" target="right">Мій факультет</a> <br>
      <a href  ="file2.html" target ="right">Поліклініка</a><br>
</body></html>
```

Розглянемо ще один файл з фреймами.

Приклад 1.5. Файл з фреймами.

```
<HTML><HEAD>
<TITLE></TITLE>
</HEAD>
<FRAMESET
ROWS="25%,50%,25%">
<FRAME
SRC="1.HTML">
<FRAMESET
COLS="25%,75%">
<FRAME SRC="2.HTML"
> <FRAME
SRC="file1.HTML">
</FRAMESET>
<FRAME SRC="1.HTML">
</FRAMESET>
</HTML>
```

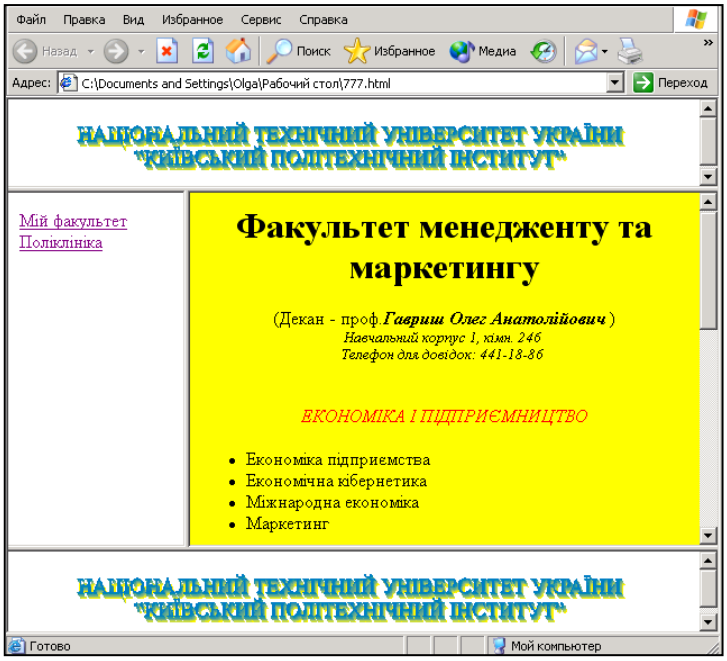


Рис 1.1. **Вікно з фреймами прикладу 1.5**

1.4.2. Форми

Форми являють собою найбільш важливі інтерактивні елементи HTML. З їхньою допомогою користувач може повертати коментарі із приводу відвідування певного вузла, пересилати запити або реєструватися. Розроблювач задає питання створюючи форму, а користувач відповідає на них заповнюючи її. Зміст форми або передається сценарію CGI (*Common Gateway Interface*) - стандарт інтерфейсу, для зв'язування зовнішньої програми з веб-сервером; або по електронній пошті посилається розроблювачу. Сам процес створення форми складається із двох етапів. Перший полягає в створенні самої форми, а другий містить у собі створення на сервері сценарію CGI. Форма створюється за допомогою різних тегів й атрибутів, вкладених у пару тегів <FORM></FORM>.

Елемент <FORM>

Елемент **<FORM>** є необхідною умовою для всіх форм. Він може мати наступні атрибути:

- **method**

Цей атрибут визначає спосіб пересилання даних сценарію CGI. Тут протокол GET обраний за замовчуванням, але в більшості випадків розроблювачі користуються протоколом POST, що дозволяє передавати більші обсяги даних.

- **action**

Цей атрибут визначає шлях до сценарію CGI або адресу електронної пошти.

- **enctype**

Цей атрибут визначає спосіб кодування вмісту форми. Іншими словами він повідомляє браузеру про спосіб кодування інформації перед відсиланням серверу. За замовчуванням використовується значення **x-www-form-encoded**.

Синтаксис форми для сценарію: **<FORM method="get" або "post" action="URL сценарію" ></FORM>.**

Синтаксис форми для пошти: **<FORM method="get" або "post" action="mailto:адреса" ></FORM>.**

Елемент <INPUT>

Елемент **<INPUT>** є базовим для всіх елементів форми. Він використовується для впровадження у форму кнопок, графічних зображень, прапорців, перемикачів, паролів і текстових полів. Незважаючи на зовнішні відмінності форм всі вони пересилають сценарію **CGI** дані у вигляді пари **ім'я: значення**. Елемент може мати вісім атрибутів, що позначуються як **type:**

- **TEXT**

Однорядкове текстове поле, використовується для введення інформації, яку не можна ввести в жодному з інших елементів форми. Сюди вводяться імена,

адреси, посади, телефони, та дані практично будь-якого типу. Елемент може мати атрибути:

- **maxlength** – задає максимально припустиму довжину значення, що вписується, у символах;
- **size** – задає максимально припустиму довжину поля в символах;
- **value** – задає значення за замовчуванням, яке можна міняти.

Синтаксис :**<INPUT type="TEXT" name="Hobby" maxlength="35" size="20" value="Shopping">**

- **PASSWORD**

Однорядкове поле, у якому замість символів, що вводять, відображаються зірочки. Елемент може мати атрибути:

- **maxlength** – задає максимально припустиму довжину значення, що вписується, у символах;
- **size** – задає максимально припустиму довжину поля в символах;
- **value** – задає значення за замовчуванням, яке можна міняти.

Синтаксис :**<INPUT type="PASSWORD" name="PASSWORD_BOX" maxlength="35" size="20">**

- **HIDDEN**

Ще один тип прихованого введення інформації. Дозволяє пересилати сценаріям інформацію, що не може бути змінена користувачем. Деякі програми **CGI** використовують приховані поля для передачі інформації з однієї сторінки в іншу, наприклад, ім'я або номер. Такий підхід істотно полегшує роботу користувача, рятуючи його від необхідності повторного введення даних. Наприклад для пересилання файлу з вихідним кодом **HTML** використовується наступна конструкція :

<INPUT type="HIDDEN" name="file" value="anyfile.html">

- **CHECKBOX**

Прапорці використовуються для надання можливості користувачеві відповісти односкладово: **так/немає істина/неправда більше/менше** й т.д.

Виглядає звичайно у вигляді хрестика або пташки. Елемент може мати атрибути:

- **checked** – задає початковий статус прапорця за замовчуванням;
- **value** – задає значення за замовчуванням, яке можна міняти.

Синтаксис:**<INPUT type="checkbox" name="send_mail" value="yes" checked>**

- **RADIO**

Перемикачі багато в чому нагадують прапорці, відрізняючись лише більш широкими функціональними можливостями вибору. У групі перемикачів може бути обраний лише один. Для кожного перемикача вказується окремий елемент **INPUT**.

Приклад 1.6. Приклад використання перемикачів.

Visa <INPUT type="radio" name="payment_type" value="visa">
Mastercard <INPUT type="radio" name="payment_type" value="mastercard">
American Express <INPUT type="radio" name="payment_type" value="AmEx" checked>

Visa ○ Mastercard ○ American Express ○

- **SUBMIT**

Клацання на цій кнопці приводить до пересилання змісту форми сценарію, що був заданий атрибутом **action** в елементі **<FORM>**. За допомогою кнопок можна обчислювати суму, завантажувати сторінки, пересилати дані, скидати значення.

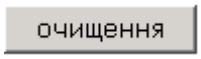
Синтаксис: <FORM method="get" або "post" action="mailto:name@domen.ru" >

<INPUT type="submit" value="послати"></FORM>

- **RESET**

Кнопка використовується для відновлення значень, заданих за замовчуванням. Якщо значення за замовчуванням не передбачено, то воно просто обнулиться. Ширина кнопки може мінятися залежно від інших елементів. Має так само атрибут **value**.

Синтаксис :**<INPUT type="reset" value="очищення">**



- **IMAGE**

Багато в чому схожий на кнопку **SUBMIT**, тільки як кнопку використовує зображення. Однією з переваг є можливість передачі координат миші користувача, що дозволяє організувати карту зображень. Елемент може мати атрибути:

- **src** – задає **URL**(адресу) файлу із зображенням;
- **align** – задає вирівнювання зображення щодо тексту за допомогою значень **TOP , MIDDLE** і **BOTTOM**;
- **name** –задає ім'я карти, яке так само пересилається сценарію разом з координатами.

Синтаксис :**<INPUT type="image" src="knopka.gif">**

- **BUTTON**

Створює іншу кнопку, браузери користувачів можуть використовувати значення атрибута **value** як вихідне ім'я файлу.

Синтаксис :**<INPUT type="button" value="кнопка">**

- **FILE**

Створює керуючий елемент – вибір файлу.

Синтаксис :**<INPUT type="file">**

- **ACCESSKEY**

Задає кнопку, при натисканні якої відбувається обробка поля.

Синтаксис :**<INPUT accesskey="a">**

- Приклад натисніть Alt+a:
- 

- **ID**

Задає ім'я для посилання.

Синтаксис :**<INPUT id="ім'я">**

- **SIZE**

Задає ширину елемента в пікселях.

Синтаксис :**<INPUT size="число">**

- **DISABLED**

Відключає можливість змінювати вміст поля або положення кнопки.

Синтаксис:**<INPUT disabled=”Shopping”>**

Елемент <TEXTAREA>

За допомогою цього елемента створюється область для введення й перегляду тексту. Він може використовуватися й не в складі форми, а як самостійна деталь сторінки. Область введення допомагає заощадити місце завдяки смугам прокручування. Може мати атрибути:

- **name** – задає ключове слово, по якому сценарій може звертатися до його змісту;
- **rows** – задає висоту області в рядках;
- **cols** – задає ширину області в символах.

Приклад 1.7.

Синтаксис :<FORM><H3>Введи текст</H3>

<TEXTAREA name="ключове слово" rows=5 cols=30>Область для введення тексту

</TEXTAREA>

<INPUT type="reset" value="очищення"></FORM>

Елемент <SELECT>

Елемент **<SELECT>** може приймати форму списку, що розкривається, або меню елементів. Має вкладений тег **<OPTION>** й атрибути :

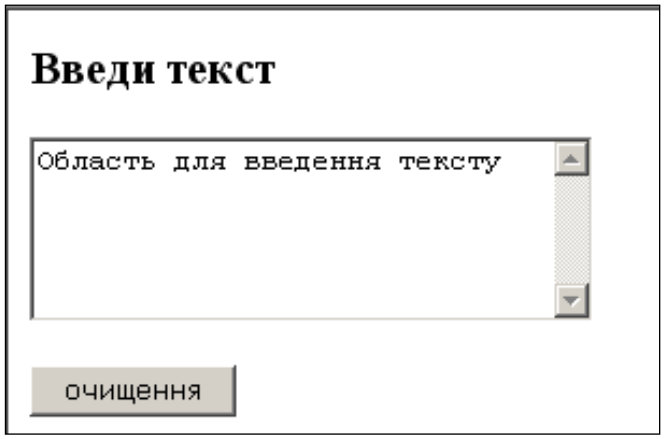


Рис.1.2 **Форма прикладу 1.7.**

- **name** – задає ім'я;
- **size** – задає максимальна кількість елементів списку, що одночасно відображаються на екрані;
- **multiple** – задає можливість одночасного вибору декількох значень.

Елемент <OPTIONS>

Елемент же **<OPTIONS>** задає можливі варіанти вибору меню **<SELECT>**

Синтаксис:**<OPTION value="n" selected>значення**

має атрибути:

- **selected** – задає вибране слово;
- **value** – задає значення вибраного слова для сценарію.

Приклад 1.8. Приклад використання меню.

<H3>Вибери потрібне</H3>

<SELECT multiple>

<OPTION value=a>Перший</OPTION>

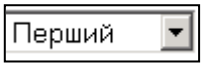
<OPTION value=b>Другий</OPTION>

<OPTION value=c>Третій</OPTION>

<OPTION value=d>Четвертий</OPTION>

</SELECT>

Вибери потрібне



Приклад 1.9. Приклад використання оператора SELECT.

<SELECT size=1>
<OPTION selected value=1>Виберіть:</OPTION>
<OPTION value=2> Перший</OPTION>
<OPTION value=3> Другий</OPTION>
<OPTION value=4>Третій</OPTION>
<OPTION value=5> Четвертий</OPTION>
</SELECT>

Елемент <OPTGROUP>

Елемент **<OPTGROUP>** застосовується для логічного згруповування елементів **<OPTION>** усередині тега **<SELECT>** має атрибут **label**:

Синтаксис : <SELECT size=1>
<OPTGROUP label="Перша група» >
<OPTION selected value=1>Виберіть:</OPTION>
<OPTION value=2> Перший </OPTION>
<OPTION value=3> Другий </OPTION>
</OPTGROUP>
<OPTGROUP label="Друга група">
<OPTION value=4>Третій</OPTION>
<OPTION value=5> Четвертий</OPTION>
</OPTGROUP>
</SELECT>

Елемент <ISINDEX>

Це найпростіший елемент, що дозволяє створити подібність форми й уведення рядка, що містить текст і генерує запит.

Приклад : **<ISINDEX prompt="Рядок для уведення критерію пошуку" >**

Допустимо, що на поточній сторінці заданий базовий **URL** за допомогою елемента

<BASE href="URL пошукового засобу в Internet">

тоді, якщо користувач уведе в поле ключові слова для пошуку **слово1, слово2, слово3**, то браузер згенерує й відішле запит для пошукової машини сервера у вигляді:

http://www.назва.домен/?слово1+слово2+слово3.

Якщо пошукова програма сервера підтримує стандартний синтаксис запиту з використанням знаків **?** і **+**, пошук буде здійснений.

Елемент <BUTTON>

Елемент **<BUTTON>** є альтернативою елементу **<INPUT>** з більшими можливостями – наприклад із завданням альтернативного тексту.

Синтаксис :**<BUTTON><IMG src=...>**

</BUTTON>

- **name –** задає ім'я елементу;
- **value –** задає значення елементу;
- **type –** при використанні як кнопка приймає значення : **button**, **submit** й **reset**;
- **disabled –** робить недоступним даний елемент;
- **tabindex –** визначає положення в послідовності переходу клавішею **Tab**, відключені поля форм не беруть участі у черговості;
- **accesskey –** задає клавішу доступу;
- **id –** задає ім'я для посилання.

Приклад: **<BUTTON name="submit" type="submit">відправити </BUTTON >** послати

Елемент <LABEL>

Елемент **<LABEL>** застосовується для альтернативного завдання інформації для керуючих полів форми. Підтримує атрибут **for**, що зв'язує елемент **<LABEL>** з іншими елементами форми. Значення атрибута **for** повинне збігатися зі значенням атрибута **id** зв'язаного керуючого елемента.

Приклад 1.10. Використання елемента <LABEL>.

```
<FORM action="URL" method="post">
 <TABLE>
 <TR>
 <TD><LABEL for="fname">Ім'я</LABEL>
 <TD><INPUT type="text" name="firstname" id="fname">
 <TR>
 <TD><LABEL for="lname">Прізвище</LABEL>
 <TD><INPUT type="text" name="lastname" id="lname">
 </TABLE>
 </FORM>
```

Приклад 1.11. Приклад створення форми.

```
 <FORM  action="URL" method="post">
<P>
 <LABEL for="firstname">Ім'я: </LABEL>
 <INPUT type="text" id="firstname"><BR>
 <LABEL for="lastname">Прізвище: </LABEL>
 <INPUT type="text" id="lastname"><BR>
 <LABEL for="email">email: </LABEL>
 <INPUT type="text" id="email"><BR>
 <INPUT type="radio" name="sex" value="Чоловічий">Чоловічий<BR>
 <INPUT type="radio" name="sex" value="Жіночий">Жіночий<BR>
```

<INPUT type="submit" value="Відправити"> <INPUT type="reset">
</P></FORM>

Елемент <FIELDSET>

Елемент **<FIELDSET>** дозволяє логічно згруповувати елементи форми.

Синтаксис:**<FIELDSET> ім'я</FIELDSET>**

Елемент <LEGEND>

Елемент **<LEGEND>** дозволяє давати найменування логічним групам елементів форми.

Синтаксис: **<LEGEND>ім'я</LEGEND>**

Приклад 1.12. Використання тегу <LEGEND>.

```
<FORM>
<FIELDSET>
<LEGEND>Група 1</LEGEND>
<INPUT type="text" id="name1"><BR>
<INPUT type="text" id="name2"><BR>
</FIELDSET>
<FIELDSET>
<LEGEND>Група 2</LEGEND>
<INPUT type="text" id="name3"><BR>
<INPUT type="text" id="name4"><BR>
</FIELDSET></FORM>
```

Приклад 1.13. Створення форми реєстрації учасників конференції.

```
<HTML><HEAD><TITLE>ФОРМИ</TITLE>
<meta http-equiv="Content-Type" content="text/html;
  charset=windows-1251"></HEAD>
<BODY>
<H2>РЕЄСТРАЦІЯ УЧАСНИКІВ КОНФЕРЕНЦІЇ</H2>
<FORM METOD="GET">
```

ВВЕДІТЬ ІМ'Я ДЛЯ РЕЄСТРАЦІЇ <INPUT TYPE="TEXT" NAME="REGNAME"><p>

ВВЕДІТЬ ПАРОЛЬ: <INPUT TYPE="PASSWORD" NAME="PASSWORD1" MAXLENGTH=8><p>

ПІДТВЕРДЖЕННЯ ПАРОЛЯ: <INPUT TYPE="PASSWORD" NAME="PASSWORD2" MAXLENGTH=8><p>

ВАШ ВІК
<INPUT TYPE="RADIO" NAME="AGE" VALUE="LT20" CHECKED> ДО 20

<INPUT TYPE="RADIO" NAME="AGE" VALUE="20_30" CHECKED>20-30

<INPUT TYPE="RADIO" NAME="AGE" VALUE="30_50" CHECKED>30_50

<INPUT TYPE="RADIO" NAME="AGE" VALUE="ПЕ50" CHECKED> СТАРШЕ 50

ЯКИМИ МОВАМИ ВОЛОДІЄТЕ

<INPUT TYPE="CHECKBOX" NAME="LANGUAGE" VALUE="RUSSION" CHECKED> РОСІЙСЬКА

<INPUT TYPE="CHECKBOX" NAME="LANGUAGE" VALUE="ENGLISH" CHECKED> АНГЛІЙСЬКА

<INPUT TYPE="CHECKBOX" NAME="LANGUAGE" VALUE="FRENCH" CHECKED> ФРАНЦУЗСЬКА

<INPUT TYPE="CHECKBOX" NAME="LANGUAGE" VALUE="GERMAN" CHECKED> НІМЕЦЬКА

ЯКИЙ ФОРМАТ ДАНИХ ДЛЯ ВАС НАЙКРАЩИЙ

<SELECT NAME="FORMAT" SIZE=2>

<OPTION SELECTED VALUE="HTML">HTML

<OPTION VALUE="PLAIN TEXT">PLAIN TEXT

<OPTION VALUE="POSTSCRIPT">POSTSCRIPT

<OPTION VALUE="PDF">PDF

```
</SELECT>
<BR><BR>НАЗВА ВАШИХ ТЕЗІВ<BR>
<TEXTAREA NAME="WISH" COLS=40 ROWS=3>
</TEXTAREA><BR><BR>
<INPUT TYPE="SUBMIT" VALUE="OK"><INPUT TYPE="RESET" VALUE="ОТМЕНИТЬ"></FORM></BODY></HTML>
```

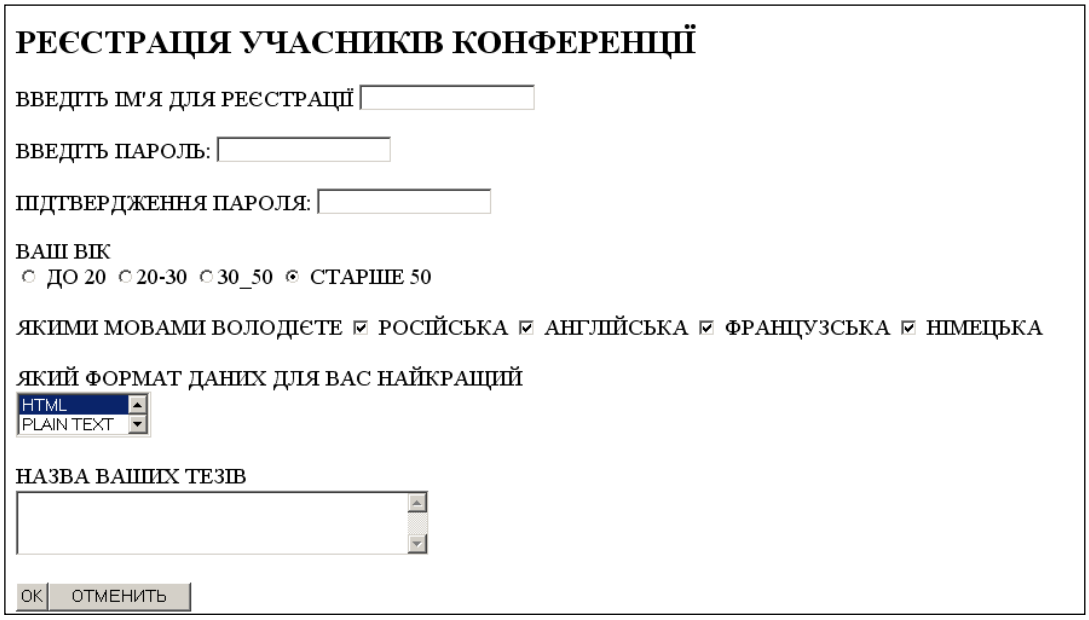


Рис.1.3.**Форма реєстрації учасників конференції прикладу 1.13**

1.5. Таблиці стилів

У попередніх розділах ви познайомилися із простими засобами, підготовки тексту для Web-сторінок - це створення абзаців, заголовків, назви шрифтів і т.д. Однак порівняння цих скромних засобів з можливості спеціалізованих програм підготовки текстів (текстових процесорі видавничих систем) говорить явно не на користь HTML.

Нагадаємо, що при роботі в текстовому процесорі (наприклад, Word) ви можете довільним чином форматувати текст. Причому можливо переносити атрибути формату з одного документа в інший. Це можливо завдяки тому, що своє бачення документа ви можете сформулювати мовою стилів. У міру необхідності можна створювати й редагувати стилі, які визначають розміри, відбиття абзаців, міжрядкові інтервали й т.д.

Наблизитися до подібних технологій при створенні Web-сторінок дозволяють таблиці стилів, які додаються до звичайних HTML-файлів. Можливості таблиць стилів навіть більше широкі, чим можливості стилів звичайного текстового процесора. Користуючись таблицями стилів, розроблювач може з точністю до пікселя визначити положення на сторінці будь-якого об'єкта, підняти накладення одного об'єкта на інший, автоматизувати застосування стилів до об'єктів і класів.

1.5.1. Поняття про каскадні таблиці стилів

Стандарти таблиці стилів для Web були розроблені консорціумом W3C в 1995-96 роках і були названі *Cascading Style Sheets (каскадні таблиці стилів),* або скорочено *CSS.* Присутність слова «каскадні» у цьому тексті означає, що таблиці стилів дозволяють створювати ієрархію стилів. В HTML-документ представляється як структура вкладених елементів різного рівня, у якій стиль елемента більше низького рівня має пріоритет перед стилем зовнішнього елемента високого рівня.

1.5.2. Підтримка таблиць стилів

Таблиці стилів дозволяють звільнитися від деяких специфічних елементів HTML, які консорціум W3C уважає застарілими й рекомендує використати замість них властивості стилів з CSS. Ситуація ускладнюється тим, що існують кілька типів таблиць стилів. Сьогодні все популярнішою стає розширювана мова стилів XSL (eXtensible Style Language), що використовує синтаксис мови XML (eXtensible Markup Language - розширювана мова розмітки).

Популярні браузери Internet Explorer й Netscape, починаючи з версій 4.0, забезпечують підтримку таблиці стилів *CSS1 (Cascading Style Sheets Level 1 – таблиці каскадних стилів, рівень I),* хоча й у різному ступені. Вже існує друга версія таблиць стилів *CSS2.*

HTML-документ являє собою безліч вкладених один в одного елементів(наприклад, елементи BODY, P, HI, H2 й інші). Для кожного елемента передбачений припустимий набір властивостей, що визначається

специфікацією HTML. У коді HTML прийнято властивості записувати у вигляді атрибутів, які вказуються в початковому тегу кожного елемента, (наприклад, гарнітуру шрифту конкретного абзацу, кольори тла документа).

Властивість стилю - *це параметр стильового оформлення документа, що визначається специфікацією CSS.*

В таблицях стилів значення властивості приєднується до нього за допомогою двокрапки. Наприклад, призначення червоних кольорів задається записом:

color:red

а розмір шрифту

font-size:14pt

Значення властивостей стилю записуються, без лапок. Однак якщо значення складається з декількох слів із пропусками (наприклад, найменування шрифту), тоді лапки (одинарні ' або подвійні ") ставляться.

1.5.3. Визначення вбудованого стилю (атрибут style)

Перш ніж почати розглядати використання CSS, зупинимось на найпростішому способі завдання стилю елемента HTML через атрибут style. Ви можете в процесі розробки сторінки швидко змінити стиль елемен додавши до тегу атрибут style із вказівкою необхідних властивостей. До-

пустимо, ви хочете, щоб абзац був набраний шрифтом розміром 6pt. Для цього потрібно записати в такий спосіб:

<P style=”font-size: 6pt ”>

При відображенні елемента P браузер відповідно до цього запису замінить розмір основного шрифту, прийнятого за замовчуванням, на шрифт розміром 6pt. Наведений спосіб форматування називається *вбудованим стилем.*

Вбудований стиль можна застосовувати до рядкових елементів (наприклад SPAN). Наступний код привласнює червоний кольор одному слову в заголовку:

<H1><SPAN style=”color:red”> Захист </SPAN>комп'ютерних мереж</H1>

Ви можете встановити вбудований стиль для блоку, при цьому властивості стилю будуть застосовані до всіх елементів, що входять у блок.

Приклад 1.14. Використання вбудованого стилю для блока.

<html>

<DIV style="font-family:serif;color:brown">

<h1>Частина перша</H1> </h1>Розділ 1</h1>

<h3>ТАБЛИЦІ СТИЛІВ</h3> Стандарти таблиць стилів для Web були розроблені консорціумом W3C

</DIV></html>

У всіх вкладених елементах HI, H2, H3 буде використаний шрифт сімейства serif коричневих кольорів. Значення атрибута style обов'язково обрамляють в лапки (одинарні ' або подвійні "), причому властивості відокремлюють один від одного крапками з комами (пробіли не обов'язкові).

Вбудовані елементи. Елементи, які не форматуються як блокові, є вбудованими елементами.

Приклад 1.15. Вбудовані елементи.

<html>

<B style="background-color:

red"> Кафедра

промислового маркетингу

</B>

готує

конкурентоспроможних

</style>

<EM style="background-

color: lightsteelblue"> фахівців </EM>.</P> </style>

<B style="background-color:#009933"></B>

</style></html>

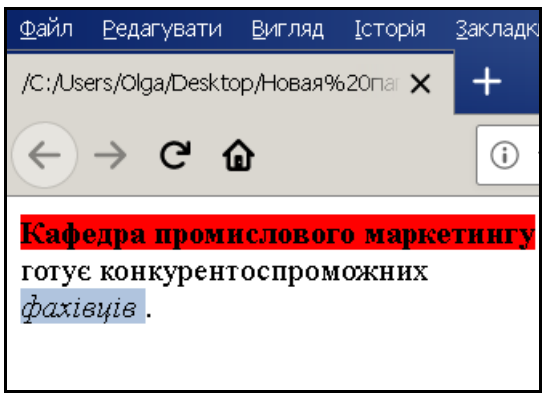


Рис.1.4.Форма реєстрації учасників конференції прикладу 1.15

1.5.4. Формування таблиць стилів. CSS-правила

Відповідно до специфікації CSS стильові властивості вводяться за допомогою *визначення стилю,* що прийнято позначати фігурними дужками. Наприклад, призначення курсивного накреслення тексту виконується за допомогою визначення:

{font-style:italic}

В одному визначенні стилю може записуватися кілька властивостей. Властивості перераховуються через крапку з комою, наприклад:

{color:blue;font-family:Arial;text-align:right}

Відповідно до специфікації CSS призначення стилю тому або іншому елементу HTML здійснюється так:

Елемент{ Визначення стилю }

Наприклад, якщо ви хочете, щоб всі елементи абзаців P відображалися шрифтом Arial розміром 14 pt, запишіть наступний рядок:

P{font-family:Arial; font-size: 14pt}

Приведемо ще один приклад - це призначення стилю елементу, позначеному в документі ідентифікатором id="f1". Нехай цей елемент уводить у документ малюнок, і ви хочете, щоб останній мав певні розміри: 150 пікселів завширшки й 100 пікселів у висоту. Це досягається за допомогою наступного коду:

#fl{width: 150; height:100}

Елемент, до якого ставиться обумовлений стиль, узагальнено називається *селектором.* У попередніх двох прикладах селектори були представлені: HTML - елементом P й ідентифікатором f1. Селектори, що збігаються з іменами тегів, позначають *класи,* а селектори, що починаються зі знака #(«рамка»), відповідають *унікальним ідентифікаторам* елементів.

Селектори позначаються по імені елементів, до яких застосовується визначення стилю. Наприклад, селектор TABLE указує на те, що визначення стилю ставиться до елементів Table. Селектори нечутливі до регістра, однак

ми будемо дотримуватися прийнятої в CSS запису елементів прописними буквами.

Запис, що складається із селектора й визначення стилю, називається CSS-правилом. Таблиці стилів - це фактично набори CSS-правил, які задають властивості форматування елементів документа.

1.5.5. Внутрішні й зовнішні таблиці стилів

Раніше ми розглянули найпростіший спосіб стильового форматера – вбудовані стилі, що мають істотний недолік - вони не дозволяють відокремити засобів форматування документа від його змісту. Крім того, оголошення вбудованого стилю доводиться повторювати для кожного документа, що форматується, протягом усього Web-сайту.

Від цих недоліків вільний інший спосіб введення стилів — це розміщення таблиці стилів у заголовній частині документа (в елементі HEAD). Заголовний стиль, дає можливість керувати оформленням документа.

Вбудовані й заголовні стилі належать відповідно до специфікації CSS до ***внутрішніх таблиць стилів.*** *Це назва вказує на те, що визначення стилів виконується в рамках одного документа. У той же час можливе завдання стилів у файлі с розширеннями .css або jss. Такий спосіб визначення називається* ***зовнішніми таблицями стилів.***

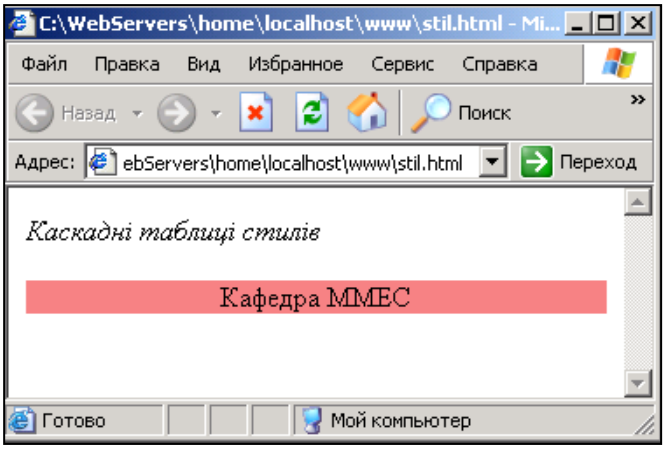


Рис.1.5. **Вікно прикладу 1.16**

Приклад 1.16. Зв'язування таблиць стилів з документом.

Створюємо файл “my.css”

```
<STYLE TYPE="text/css">
<!-i
I{text-transform: uppercase;
background-color: Aqua;
}
```

```
P{background-color: #F08080;
text-align : center;
}
-i>
</STYLE>
```

Зв'язування дозволяє зберігати таблицю стилів в окремому файлі й приєднувати її до документів за допомогою тега, що задає в розділі <head>

```
<html><head>
<LINK REL="stylesheet" TYPE="text/css" href="my.css">
</head><body>
<I>Каскадні таблиці стилів</I>
<P>Кафедра ММЕС</P>
</body></html>
```

1.5.6. Введення заголовного стилю (елемент STYLE)

Як відзначалося вище, CSS являє собою мову опису інформації про стиль форматування елементів HTML-документа. Тому стосовно HTML-документа таблиці стилів CSS є сторонніми елементами. Для розміщення інформації про стиль застосовується спеціальний контейнер <STYLE>, що вставляється в заголовну частину документа й має вигляд:

```
<STYLE type="text/css”>
<!— Опис таблиць стилів —> </STYLE>
```

У цьому коді значення text/css атрибута type повідомляє браузеру, що в поточному файлі застосований текст мовою CSS. Всі формальні описи стилів, що являють собою CSS-правила, містяться в елементі STYLE.

Пояснимо визначення стилів у наступному прикладі. Всі елементи будуть відображені шрифтом розміру 15pt, а деякі елементи H2, виділені в клас curs, будуть представлені курсивом з розміром 16pt. У той же час елемент, позначений ідентифікатором id="uk", буде виведений білим шрифтом на зеленому тлі.

Поєднання CSS-правил з конкретними елементами документа виконується за допомогою атрибутів class або id. Наведений приклад таблиць стилів може застосовуватися до такого елемента BODY.

Приклад 1.17. Внутрішні таблиці стилів.

```
<head><title> Внутрішні таблиці стилів</title>
 <STYLE type="text/css">
<!— Нижче опис стилів —>
h2      {font-size:15pt}
H2.curs   {font-size:16pt;font-style.italic}
#uk {background-color:green;color:white}
</STYLE>
<!- Кінець опису стилів >
</HEAD>
<body>
<h2>Заголовок н2</h2>
<h2 class=”curs”>Перший заголовок класу curs</H2>
<h2 id="uk">Заголовок uk    </H2>
<h2 class="curs">Другий заголовок класу curs</H2>
</body>
```

1.5.7. Види селекторів

У специфікаціях CSS визначено кілька видів селекторів, основні з них використані в попередньому прикладі. Розглянемо можливі види селекторів докладніше.

Селектори типу

Селектор типу встановлює відповідність між визначенням стилю й типом HTML елементів, до яких цей стиль буде застосований. Наприклад:

P {font-family:arial; font-size:14pt}

Використається селектор типу й задає шрифт для всіх абзацних блоків HTML, а інше правило:

H2 {font-family:monospace; background-color:magenta}

визначає для всіх заголовків другого рівня H2 моноширинний шрифт, пурпурне тло.

Селектори класу

HTML-елемент або група елементів, оформлених одним стилем, утворюють клас. Клас може позначатися так само, як і тип HTML-елемента, наприклад абзаци документа утворюють клас P. Однак частіше класу привласнюється формальне ім'я, що надалі буде пов'язане з певним набором стильових властивостей. Допустимо, ви можете заголовки першого рівня H1 кольори згрупувати в клас із ім'ям blue:

H1.blue {color:blue}

Ім'я класу відокремлюється від імені елемента крапкою. Інший приклад завдання класу:

H2.curs {font-size:16pt;font-style-.italic}

Як бачимо, до складу класу входять не всі елементи одного типу H2**,** а тільки ті, які форматуються однаковим стилем. У той же час тому самому класу можуть належати різнотипні HTML- елементи. Допустимо, ви хочете задати відображення різних елементів (абзаців P, DIV і т.д.) на жовтому тлі. Для цього в список стилів уводиться CSS-правило із крапкою перед ім'ям класу, але без вказівки назви елементів, наприклад

.bgyellow {background-color:yellow}

Селектор .bgyellow позначає клас, а вираз у фігурних дужках – визначення стилю, що буде ставитися до будь-якого елемента bgyellow. Приналежність HTML-елемента цьому класу повинна бути зазначена в атрибуті class, наприклад:

<DIV class="bgyellow">

<P class="bgyellow”>

<SPAN class="bgyellow">

Селектори, що ставляться до будь-якого елемента документа, у специфікації прийнято називати *універсальними селекторами.*

Селектори id

При розробці Web-сторінки часто виникає необхідність змінити стиль тільки одного елемента. Простіше всього це зробити в такий спосіб. Задайте за допомогою селектора - id індивідуальний стиль, і цей стиль буде автоматично привласнений конкретному елементу, позначеному ідентифікатором id.

Наприклад, щоб виділити червоними кольорами текст абзацу, позначеного ідентифікатором id= "red", можна скористатися наступним CSS-правилом: #red {color:FF0000}. Потрібний елемент абзацу повинен містити одноіменне значення атрибута id.

Приклад 1.18. Приклад використання селектора.

```
<head>
<style type="text/css">
 #red {color:red}
 </style></head>
<body>
<p id="red"> абзац буде відображений червоним шрифтом </p>
</body>
```

В одному документі цей атрибут з певним значенням може зустрічатися тільки один раз.

1.6. Значення властивостей

Розрізняють числові й символьні значення властивостей.

Числові значення застосовуються для завдання розмірів, наприклад, ширини блоку, розміру шрифту, товщини рамки, міжрядкових інтервалів. Значення виражається десятковим числом, за яким звичайно слідує розмірність. Розмірність записується після числа без пробілу. Наприклад, 8pt, 1cm, 2.5in, 130%. У випадку негативного значення перед числом ставиться знак мінус (наприклад, -15px). Числові значення можуть виражатися в абсолютних або відносних одиницях.

Абсолютні значення задають точний розмір елемента й приводяться в стандартних одиницях виміру довжини, наприклад, у дюймах, сантиметрах або міліметрах.

Відносні значення визначають розмір елемента щодо іншого, елемента. Наприклад, ширина зображення може виражатися у відсотках щодо ширини блоку, у який вкладений малюнок, або щодо розміру вікна браузера. Міжсимвольний інтервал часто задається в одиницях ***em*** ширини символу основного шрифту (букви «m»). Інші одиниці відносних значень наведені в таблиці.

Таблиця 1.6. **Одиниці відносних значень**

Умовне позначення	Найменування одиниці	Приклад
Абсолютні одиниці		
in	дюйм	width:.25in
cm	сантиметр	height:1.5cm
mm	міліметр	margin-left: 12
pt	пункт (lpt= l/72in)	font-size:16pt
pc	пік(1pc= 12pt)	Line-height:1.2pc;
Відносні одиниці		
px	піксель	left:200px
em	ширина букви «т»	letter-spacing:
ex	висота букви «х»	font-size:2ex
%	відсоток	width:150%

Символьні значення складаються з букв латинського алфавіту або з комбінацій букв, цифр і спеціальних символів. Форма запису визначається згідно правилам CSS. Символьні значення привласнюються багатьом стильовим властивостям наприклад, властивостям шрифтів font-family (гарнітура шрифту) і style (стиль виведення шрифту), властивості тексту text-align (вирівнювання). Іншим прикладом використання символьних значень є властивості кольорів color й background-color, значення яких задаються в символьному форматі RGB-моделі. Відзначимо, що в CSS для опису кольорів застосовуються й інші формати RGB-моделі, які не підтримуються в HTML:

десятковий формат (наприклад, rgb (128,0,128)) і процентний формат (rgb(50%,0,50%)).

1.6.1. Властивості шрифтів

В CSS передбачена безліч властивостей для керування шрифтами (завдання розміру, накреслення й т. д). Ці властивості можна призначати разом із властивостями, що визначають кольори шрифту, тло, відступи, міжсимвольну відстань. Розглянемо коротко тільки *властивості шрифтів:*

font-family - задає гарнітуру шрифту, що буде використана для виведення тексту. Значенням цієї властивості може бути назва конкретного шрифту (наприклад, Arial) або назва сімейства шрифтів;

font-size- розмір шрифту;

font-style - задає стиль виведення символів. Можливі наступні значення властивості: normal (звичайний), italic (курсив) і oblique (похилий);

font-weight - визначає ступінь «жирності» шрифту. Для цього використовуються цілі числа з діапазону від 100 до 900 із кроком 100 одиниць. Однак застосовують ключові слова: bold (жирний), bolder (більше жирний) і lighter (більше тонкий);

font-variant - вказує варіант накреслення поточного шрифту. Для цієї властивості браузери підтримують тільки два значення: small-caps (відображення малими прописними буквами) і normal (не впливає на відображення).

1.6.2. Властивості тексту

Зрозуміло, що задати шрифт - ще не значить визначити зовнішній вигляд тексту. Необхідно також вказати властивості, які відповідають за міжрядковий інтервал, відстань між словами й буквами й т.д. Ці властивості, називається ***властивостями тексту,*** визначаються для абзаців і навіть для всього документа, тобто задаються на рівні блоків (P, DIV, BODY, SPAN, STRONG й ін.). Перелічимо властивості тексту, передбачені специфікацією CSS:

letter-spacing - встановлює відстань між буквами (міжрядковий інтервал). За замовчуванням цій властивості привласнюється значення normal. Можна вказувати відстань у будь-яких абсолютних одиницях;

word-spacing - встановлює відстань між словами. Аналогічно letter-spacing значенням цієї властивості за замовчуванням normal. Інші значення можуть бути задані в абсолютних одиницях (наприклад, 10px,2mm);

text-indent - задає відступ першого рядка абзацу (новий рядок) властивість застосовується до блоків, і її значення визначається в абсолютних одиницях (наприклад, 4mm, lcm, 20pt і т.д.) або у відсотках від ширини (наприклад, 5% або 10%). За замовчуванням значення властивості text-indent дорівнює нулю. Якщо цій властивості привласнити негативне значення, то замість абзацного відступу одержимо виступ першого рядка;

text-align - задає горизонтальне вирівнювання для тексту, розміщеного усередині елемента (наприклад, P або DIV). Ця властивість приймає: center (по центру), left (уліво), right (вправо) і justify (по ширині). За замовчуванню текст вирівнюється по лівому краю. Ефект вирівнювання помітний при великому розмірі шрифту й малій ширині вікна браузера;

vertical-align - встановлює розташування тексту й малюнків по вертикалі щодо базової лінії. Властивість може приймати, значення: baseline (вирівнювання по базовій лінії, приймається за замовчуванням), sub (вирівнювання по лінії нижнього індексу), super (вирівнювання по лінії верхнього індексу);

line-height - визначає міжрядковий інтервал. Значення його можна задавати в абсолютних одиницях (наприклад, 16pt, 3mm), у відсотках (130%), а також кількістю рядків.

Як приклад спільного використання властивостей тексту приведемо HTML-код сторінки, що містить блокові й рядкові елементи з різними значеннями параметрів форматування.

Приклад 1.19. Властивості шрифтів і тексту.

<html><head><title> Властивості шрифтів і тексту </title>

<body>

Абзац, що відтворений курсивним шрифтом Helvetica, що має розмір 12pt.

Цей абзац відтворюється шрифтом Helvetica, що має розмір larger і накреслення small-caps.

Це приклад набору формули, що має верхній і нижній індекси: S

<SPAN style="vertical-align:sub" >

result

</SPAN>

=Ax

 2.

<style="font-weight:bold; text-indent: 12mm">

Даний абзац має відступ 12 мм, текст відображається напівжирним шрифтом

Times New Roman

причому назва шрифту набрана з розрядкою – міжсимвольною відстанню 3pt.

Цей абзац демонструє стильові можливості по відображенню

&laquonaїхавших» рядків

Розмір шрифту large, міжрядковий інтервал 9pt.

Даний абзац має міжрядковий інтервал 20pt і відображається шрифтом 12pt.

Перший рядок має &laquoвиступ» 15 мм.

Щоб цей &laquoвиступ» помістився у вікно браузера

для блоку P задане ліве поле 15 мм.

</body></html>

1.6.3. Властивості кольорів і тла

Відзначимо, що вживання стильової властивості background-color у цьому випадку еквівалентно використанню атрибута bgcolor для елемента BODY. Замість назви властивості background-color в таблицях можна вказувати його скорочення background, наприклад:

Style="background:red".

background-image - визначає вставку фонового зображення. Значенням властивості є URL(адреса) малюнка;

background-repeat призначає повтор фонового малюнка. Ця властивість застосовується, якщо розмір малюнка менше видимої області елемента. Щоб малюнок повторювався по горизонталі й по вертикалі, задавати властивість background-repeat не потрібно (вона за замовчуванням має значення repeat). Для повторення малюнка тільки по горизонталі вибирається значення repeat-x, тільки по вертикалі - repeat-y, а для відключення повторення no-repeat.

Приклад 1.20. Приклад фонового зображення.

```
<html><head>
<title>Фоновий малюнок </title>
<STYLE type="text/css">
 h1 {color:red}
 body{background-
image:url(telecom.jpg) ;
 background-color:yellow;
 background-repeat:no-repeat}
</STYLE>
</head><body>
<h1>Приклад фонового зображення</h1>
</body></html>
```

Рис.1.6. **Вікно прикладу 1.20**

Код документа з фоновими малюнками, що повторюються уздовж осі відрізняється від наведеного тільки заміною значення властивості background-repeat на repeat-x.

1.6.4. Властивості блоку

Текстові HTML-елементи можна представити у вигляді прямокутних блоків. До таких елементів належать P, DIV, і навіть тіло документа Body. Для них в CSS передбачена спеціальна група властивостей, що дозволяє задавати поля, границі, відступи, розміри блоку. Розглянемо окремо кожну із властивостей блоку:

margin - це ключове слово позначає набір властивостей, що визначають кожне із чотирьох полів навколо блоку. У набір входять властивості margin-top (верхне поле), margin-right (праве поле), margin-bottom (нижнє поле), margin-left (ліве поле);

border - позначає набір властивостей відображення границі елемента. Розрізняють властивості border-style (керування виведенням границі), border-width (ширина границі), border-color (коліри границі). Властивості границь

можна застосовуватися не тільки до блокових елементів, але й до рядковим елементів (наприклад, SPAN). При цьому рядковий елемент автоматично перетворюється в блок, відділений від іншого тексту порожніми рядками.

Значеннями властивості border-style є dotted (границя з точок), dashed (пунктирна границя), solid (звичайна суцільна границя), double (подвійна лінія), groove («утоплена» лінія границі), ridge (опукла границя), insert (об'єкт, «втиснений» у сторінку), outset (опуклий; об'єкт).

Ширину границі (властивість **border-width)** можна задавати як в абсолютних одиницях (наприклад, 3px), так і за допомогою ключових слів: thin (тонка), medium (середня) і thick (товста). Кольори границі (властивість **border-color)** визначається символьною назвою кольорів, кодом #RRGGBB, у двійковому форматі (наприклад, rgb (56,28,18)) або процентному форматі (наприклад, rgb(25%,30%,70%));

padding - визначає пропуск між вмістом блоку й границею. Для кожної, сторони є своя властивість: padding-top, padding-right, padding-п й padding-left;

weight height - задають відповідно ширину й висоту блоку без підрахунку відступів, границь і полів;

float - визначає розміщення поточного елемента по горизонталі стосовно зовнішнього елемента;

clear - скасовує дія властивості float.

1.6.5. Властивості списків

Таблиці стилів дозволяють управляти відображенням списків: задавши гарнітуру, розмір і кольори шрифту, вибирати вид маркерів (кружок, малюнок і т.д.). Нагадаємо, що маркер - це символ або зображення яким починається кожен рядок списку.

Ми розглянемо властивості, призначені для форматування маркерів списків: list-style-type, list-style-image, list-style-posit: скорочена форма запису цих трьох властивостей має вигляд list-style.

list-style-type - задає маркери для впорядкованих (нумерованих) і неупорядкованих (маркірованих) списків. Набір припустимих значень властивості list-style-type включає 22 значення, більшість з яких не підтримуються розповсюдженими браузерами. Тому зупинимося на значеннях, які часто використовуються:

- square - маркер у вигляді квадратика;
- circle - кружок;
- disc - затемнений кружок (значення за замовчуванням);
- decimal - десяткові числа, починаючи з 1;
- lower-roman - рядкові римські цифри, наприклад, i, ii, iii;
- upper-roman - прописні римські цифри, наприклад, I, II, III;
- lower-latin або lower-alpha - рядкові латинські букви;
- upper-latin або upper-alpha - прописні латинські букви, наприклад A, B;
- none - маркер не відображається;

list-style-image - задає маркер у вигляді картинки (зображення);

list-style-position - встановлює позицію маркера в рядку списку. Значення цієї властивості звичайно задає додатковий відступ перед кожним рядком списку.

Приклад 1.21 Використання властивостей списків.

```
<html "><head><title>Untitled Document</title></head>
<body>
<style>
   ol li {border: 1px solid #06c;}
</style>
<ol>
<li>Шевченко Т.Г.
<ul>
<li>«Кобзар»</li>
<li>«Сон»</li>
<li>«Катерина»</li>
```

```
        </ul>
        </li>
        <li>Котляревський Іван Петрович
        <ul>
        <li>«Наталка Полтавка»</li>
<li>«Енеїда»</li>
<li>«Москаль-
чарівник»</li>
 </ul>
</li>
<li>Леся Українка
<ul>
<li>«Лісова
пісня»</li>
<li>«Думи і мрії»</li>
<li>«Contra spem
spero!»</li>
      </ul>
   </li>
</ol>
</body>
        </html>
```

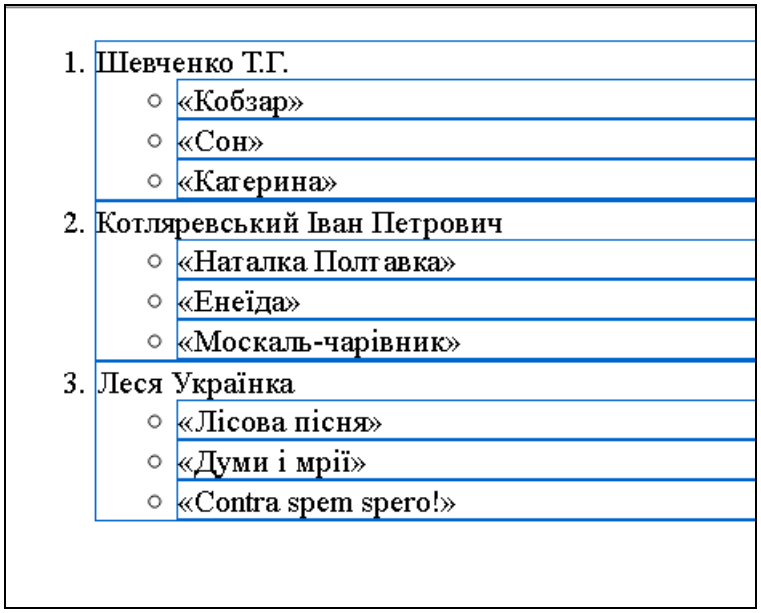


Рис.1.7. **Результат виконання прикладу 1.21**

1.7. Сусідні селектори

Сусідній селектор визначає знак "плюс" (+), що розділяє дві послідовності простих селекторів. Селектори знаходяться всередині одного контейнера і слідують другий за першим безпосередньо, не розділені ніякими іншими тегами.

Приклад 1.22. Сусідні селектори

```
        <html ><head>
        <title> Сусідні селектори </title>
```

```
</head>
<body>
<style>
   h3 + p { padding-left: 260px; font-style: italic;}
          ol {border: 1px solid #06c;}
</style>
<h3>Маркетинг</h3>
<p>
```

Це діяльність, спрямована на досягнення цілей підприємств, установ, організацій шляхом формування попиту та максимального задоволення потреб споживачів. У широкому сенсі призначення маркетингу полягає в «визначенні та задоволенні людських і суспільних потреб».

```
</p>
<p>
<ol>
   <li>Трьома головними аспектами маркетингу є:
<ul>
<li>аналітичний аспект (розуміння ринків);
<li>активний аспект (проникнення на ринки);
<li>ідеологічний аспект.
</li></li></li></li></ol></p></body></html>
```

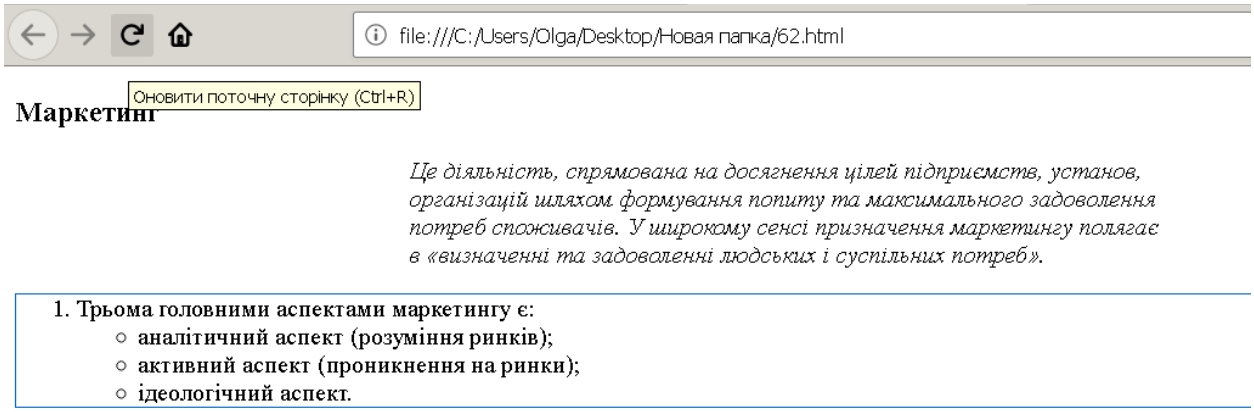


Рис.1.8. **Результат виконання прикладу 1.22**

Запитання для самоконтролю

1. Що таке гіперпосилання?
2. Що таке гіпертекст?
3. Що таке web-документ?
4. Для чого призначена програма-браузер?
5. Що таке web-сайт?
6. Яка структура простого web-документа?
7. Для чого призначена мова HTML?
8. Що таке тег і які є теги?
9. Які параметри може мати тег BODY?
10. Який тег позначає початок нового абзацу?
11. Які теги позначають товстий, курсивний і підкреслений шрифти?
12. Які теги призначені для вирівнювання елементів на сторінці?
13. Яке призначення тега FONT?
14. Яких значень можуть набувати параметри тега FONT?
15. Які є типи списків?
16. Як створити ненумерований список?
17. Як створити нумерований список?
18. Як створити список означень?
19. Яке призначення тега TABLE?
20. Які параметри може мати тег TABLE?
21. Які теги формують у таблиці рядки, клітинки-заголовки і звичайні клітинки?
22. Як у таблиці об'єднати декілька клітинок в одну?
23. Який параметр використовують для вирівнювання елементів?
24. Що таке фрейми?
25. Які файли потрібні для створення сайту з фреймами?
26. Яке призначення основного html-файлу?

27. Що відображають зазвичай у лівому фреймі сайту?
28. Які параметри може мати тег <FRAMESET>?
29. Які параметри може мати тег <FRAME>?
30. Як створюються складні конфігурації фреймів?
31. Що означає запис COLS = "30% ,*"?
32. Що означає запис COLS = "1*, 4*"?
33. Що означає запис COLS = "120, 240, *"?
34. Які фрейми створить параметр COLS = "25%, 50%, 25%"?
35. Яке призначення параметра TARGET?
36. Яке призначення тега-контейнера NOFRAME?
37. Яке призначення таблиці стилів?
38. Які є способи взаємодії таблиці стилів і html-файлу?
39. У чому полягає спосіб зв'язування?
40. Що таке каскадні таблиці стилів?
41. З чого складається таблиця стилів?
42. Назвіть властивості категорії border.
43. Назвіть властивості категорії font.
44. Назвіть властивості категорії text.
45. Назвіть властивості категорії margin.
46. Назвіть властивості категорії padding.
47. Які одиниці вимірювання застосовують у мові CSS?
48. Для чого групують властивості?
49. Що таке успадковування властивостей?
50. Яке призначення тега STYLE?
51. Яке призначення тега DIV?

РОЗДІЛ 2 БАЗОВІ ЕЛЕМЕНТИ МОВИ JAVASCRIPT

JavaScript ("JS" скорочено) – це повноцінна динамічна мова програмування, яка може надати динамічну інтерактивність веб-сторінкам. Вона була розроблена Бренданом Ейхом, співзасновником проекту Mozilla.

JavaScript був створений для надання динамічності HTML сторінок. JavaScript – це скриптова мова або мова сценаріїв. Код, написаний на JavaScript, можна вставляти прямо в HTML– код веб – сторінки. Сценарій JavaScript являє собою текстовий файл, тому написати його можна в простому текстовому редакторі, а для його роботи достатньо запустити в вікні браузера.

2.1. Навіщо потрібен JavaScript?

Завдяки скриптам JavaScript статичні HTML документи можна зробити динамічними і інтерактивними.

- Різні візуальні ефекти, на зразок слайдерів, галерей картинок і динамічного тексту;
- Перевірка призначених для користувача даних форми до їх відправки на сервер;
- Виведення інформації в нових вікнах в автоматичному режимі;
- Зміна вмісту вікна браузера, в залежності від дій користувача.

Дані – це інформація, яка зберігається в наших комп'ютерних програмах. В JavaScript є три основних типи даних: числа, рядки і булеві значення. Значенням в JavaScript можна давати імена, використовуючи змінні. Щоб створити нову змінну, використовується ключове слово var, після якого вказується ім'я змінної. Наприклад var name.

2.2. Змінні JavaScript

Змінна – це область пам'яті, яка має своє ім'я і зберігає деякі дані. Змінні в JavaScript оголошуються за допомогою оператора var, при цьому можна присвоювати або не присвоювати їм початкові значення:

```
var k;
var h = 'Привіт!';
```

Можна оголошувати відразу кілька змінних в одному операторі var (тим самим зменшуючи розмір коду), але тоді їх треба писати через кому. При цьому теж можна давати або не давати початкові значення:

```
var k, h = 'Привіт!';
var t = 37, e = 2.71828;
```

Тип змінної визначається значенням, яке їй присвоюється. Мова JavaScript – слабо типізований: в різних частинах програми можна привласнювати одній змінній значення різних типів, і інтерпретатор буде змінювати тип змінної. Дізнатися тип змінної можна за допомогою оператора typeof ():

```
var i = 5; alert (typeof (i));
i = new Array (); alert (typeof (i));
i = 3.14; alert (typeof (i));
i = 'Привіт!'; alert (typeof (i));
i = window.open (); alert (typeof (i));
```

Приклад 2.1. Присвоєння змінним JavaScript даних різних типів:

```
<script type="text/jscript">
var length = 16;  // Число
var lastName = "Сидоренко";  // Рядок
var  x = {firstName:"Іван", lastName:"Іваненко"}; // Об'єкт
</script>
```

Таблиця 2.1.**Типи даних**

Типи даних		Приклад
string	Рядковий	"Іван"
number	Числовий	5; 8.1
boolean	Логічний	True False

2.2.1.Рядкові змінні в JavaScript

Рядок (або текстовий рядок) це послідовність символів, наприклад, "Іван Петрук". Рядки повинні записуватися всередині лапок. Це можуть бути подвійні або одинарні лапки:

```
<script type="text/jscript">
var carName = "Петро Іваненко";   // Використовуються подвійні лапки
var carName = 'Василь Степаненко'; // Використовуються одинарні лапки
</script>
```

2.2.2.Числа в JavaScript

В JavaScript існує тільки один тип числових даних. Числа можуть записуватися як з десятковою крапкою, так і без неї:

```
<script type="text/jscript">
var x1 = 34.00;   // З десятковою крапкою
var x2 = 34;      // Без десяткового дробу
</script>
```

2.2.3.Логічні дані в JavaScript

Є два логічних значення: true (істина) і false (хибно).

```
<script type="text/jscript">
var x = 5;
var y = 5;
var z = 6;
(x == y) //  true
(x == z) //  false
</script>
```

В JavaScript всі типи даних є динамічними. Це означає, що одна і та ж змінна може використовуватися для зберігання даних різних типів:

```
<script type="text/jscript">
var x;       // x має тип undefined
 x = 5;       //  x - число
 x = "John";  //  x - рядок
</script>
```

2.3.Оператор typeof

Щоб визначити тип даних змінної в JavaScript використовується оператор typeof. Оператор typeof повертає тип даних змінної або виразу. У цьому випадку оператор typeof повертає один з наступних примітивних типів:

```
<script type="text/jscript">
typeof "" // повернеться "string"
typeof "John"// повернеться "string"
typeof "John Doe" // повернеться "string"
typeof 0  // повернеться "number"
typeof 314  // повернеться "number"
typeof 3.14 // повернеться "number"
typeof (3) // повернеться "number"
typeof (3 + 4) // повернеться "number"
</script>
```

JavaScript дозволяє виконувати основні математичні операції, такі як додавання, віднімання, множення і ділення. Для їх записи використовуються символи +, –, * , /, які називають операторами.

Приклад 2.2. Основні математичні операції.

```
<html><head>
<title>Змінні</title>
</head>
<body>
```

```
<script>
var a=3;
var b=4;
document.write(a+b);
</script>
</body>
</html>
```

2.4.Інкремент і декремент

Вам як програмісту знадобиться збільшувати або зменшувати значення числових змінних на одиницю.

Приклад 2.3. Використання інкременту і декременту.

```
<script>
var a=3;
var b=4;
document.write(a+b+'<br>');
++a;  // a=a+1
document.write(a+'<br>');
--b;  // b=b-1
document.write(b+'<br>');
</script>
```

Збільшення на 1 називають інкрементом, а зменшення на 1 - декрементом.

2.5.Об'єднання рядків

За допомогою оператора + можна об'єднувати рядки: результатом буде новий рядок, що складається з першого рядка, до кінця якого буде приєднаний другий рядок.

Рис.2.1. **Результат виконання прикладу 2.4.**

Приклад 2.4. Об'єднання рядків.

```
<script>
var name = "olga";
```

```
var lastname = "Ник";
document.write(name + lastname+"<br>");
document.write(lastname.length);
</script>
</body>
</html>
```

Щоб дізнатися довжину рядка, досить додати до її кінця .length:

Отримання зрізу рядка. Щоб отримати частину, рядка, використовується slice.

Приклад 2.5. Отримання зрізу рядка.

```
<script>
var string = "НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО";
document.write(string.slice(13, 76));
</script>
```

2.6. Логічні операції

Булеві значення можна об'єднувати за допомогою булевих (логічних) операторів. Результатом виразу, складеного з булевих значень і булевих операторів, завжди буде інше логічне значення (true, або false).

Три основних булевих оператора – це &&, || і!.

Приклад 2.6. Логічні операції.

```
<script>
var a = true;
var b = false;
document.write(a && b+"<br>");
document.write(a || b);
document.write("<br>");
document.write(!b+"<br>");
```

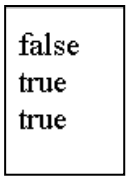


Рис.2.2. **Результат виконання прикладу 2.6.**

</script>

Щоб перевірити два числа на точну рівність, використовується потрійний знак рівності (===) – це оператор «дорівнює». Не плутайте === з одиночним знаком рівності (=), оскільки === означає « чи рівні ці два числа? », а = означає присвоїти значення. Інакше кажучи, === задає питання, а = привласнює змінній значення. В JavaScript є ще один оператор порівняння (==), який означає «практично дорівнює».

Приклад 2.7. Використання оператора порівняння.

```
<script>
var a = "4";
var b = 5;
document.write(a === b);
document.write("<br>")
document.write(a == b);
document.write("<br>")
document.write(a = b);
</script>
```

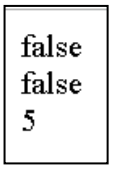


Рис.2.3. **Результат виконання прикладу 2.7.**

2.7.Особливі значення Undefined і null

В JavaScript є два особливих значення, вони називаються undefined і null. Обидва вони означають «порожньо», але зміст різний. JavaScript використовує значення undefi ned, коли не може знайти іншого значення. Наприклад, ви не присвоїли значення змінній за допомогою оператора =, її значення буде undefi ned.

Приклад 2.8. Використання значення undefined.

```
<script>
var a;
 var b=null;
document.write(a );
document.write("<br>" );
document.write(b);
```

```
</script>
```

Значення null використовується, щоб явно позначити – порожньо.

2.8.Перша програма

Щоб програма написана на JavaScript запустилася, її потрібно впровадити в HTML – код документа (або "зв'язати" програму з документом).

Для цього використовується тег <SCRIPT>, який прийнято розміщувати тег <SCRIPT> в тезі<HEAD>. Наприклад:

```
<script type="text/jscript">
document.write("<h1> Привіт, світ! <h1>");
</script>
```

Даний сценарій буде виконаний при відкритті сторінки і виведе рядок:

<Hl> Привіт, світ! </ Hl>

Однак не завжди потрібно, щоб ваша програма починала працювати відразу після відкриття сторінки. Програма може запускатися при певній події, наприклад при натисканні кнопки. Тоді при натисканні кнопки генерується подія onClick.

Приклад 2.9. Оброблювач для цієї події вказаний прямо в HTML коді.

```
<script type="text/jscript">
function hello(){
document.write("<hl>Привет, мир!</hl>");
}
</script>
<FORM>
<INPUT TYPE=button VALUE="Назад" onClick="hello()">
</FORM>
```

Приклад 2.10. Виклик функції при натисканні кнопки.

```
<body>
<script type="text/jscript">
function openWindow() {
msgWindow= open("un81.jpg")}
```

```
</script>
<form>
<input type="button" value="Натисни" onclick="openWindow()" />
</form>
</body>
```

2.9.Введення / виведення даних. Діалогові вікна

JavaScript – залежить від регістру. Чутливі до регістру ідентифікатори змінних, імена функцій, міток, ключові слова. Всі ключові слова використовують нижній регістр.

- Ідентифікатори можуть містити символи ASCII, цифри, символи підкреслення "_" і символ долара "$". Перший символ не повинен бути цифрою. В якості ідентифікаторів можна використовувати ключові і зарезервовані слова.
- Оператори розділяються крапкою з комою, яку можна опустити, якщо оператор закінчується символом нового рядка.
- Коментарі: // однорядковий коментар

/ * * / багато строковий коментар

- Змінні оголошуються за допомогою оператора var, який можна опускати, за винятком оголошення локальних змінних в тілі функції. Можливо оголошення с одночасною ініціалізацією, наприклад:
- var s = 123

2.10.Повідомлення про помилку

Помилки бувають двох типів: синтаксису і сценарію. Помилка синтаксису означає друкарську помилку або пропущений текст. Помилка сценарію означає, що ви переплутали місцями команди або вставили неправильні.

Існують програми, які допомагають виправляти помилки, цей процес називається «debugging» («знищення багів, помилок.). Рядок з помилкою

потрібно відраховувати від самого верху документа HTML, а не від першого рядка JavaScript.

2.11.Технологія розміщення коду JavaScript на Web-сторінці

1. Програмний код розміщується на HTML-сторінці. У загальному випадку можна виділити декілька способів розміщення коду JavaScript: 1. в тегів контейнері <BODY> ... </ BODY>. В тегах контейнера <HEAD> ... </ HEAD> - якщо код скрипта являє собою функцію, яка викликається у відповідь на будь-яку подію. Таке розміщення гарантує, що до моменту виклику функції вона вже буде знаходитися в пам'яті комп'ютера. Всі команди функції знаходяться всередині фігурних дужок {}.

2. розміщення в зовнішньому файлі з розширенням js, звернення до якого здійснюється з використанням тега.

3. <SCRIPT > з наступними атрибутами:

4. <SCRIPT type=”text/javascript" src="ім'я_файла">.

В JavaScript реалізовано 3 методи, які дозволяють виводити користувачеві діалогові вікна:

- alert
- confirm
- prompt

Розглянемо кожен метод більш детально.

2.12.Метод alert

Метод alert використовується для виведення найпростішого діалогового вікна, що містить текст повідомлення і єдину кнопку "Ok". Формат виклику даної функції:

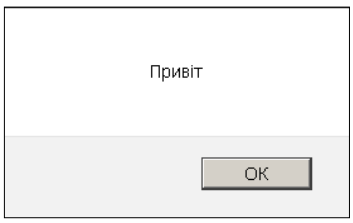


Рис.2.4 **Діалогове вікно, яке викликається методом Alert**

alert("Текст повідомлення");

Приклад 2.11 Використання методу alert.

<html > <head>

```
<title>Untitled Document</title>
</head>
<body>
<script type="text/jscript">
alert("Привіт");
</script>
</body>
</html>
```

2.13.Метод confirm

Метод confirm дозволяє вивести користувачеві діалогове вікно, що містить текст повідомлення і кнопки "Ok" і "Cancel". Використовується в тих випадках, коли користувач повинен зробити вибір. Формат виклику даної функції:

```
{
var result=confirm("Текст питання");
if(result) {
/* оператори */
}
```

Приклад 2.12. Використання методу confirm.

```
<html ><head>
<title> confirm </title>
</head>
<body>
<script type="text/jscript">
var result=confirm("Ви вивчаєте JavaScript?");
if(result) {
alert("Успіхів");
}
</script>
```

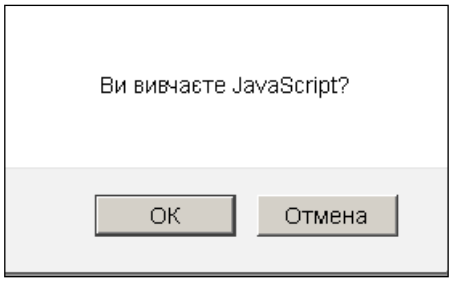


Рис.2.5 **Діалогове вікно, яке викликається методом confirm**

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

Метод confirm повертає логічне значення в залежності від натиснутої користувачем кнопки: "Ok" відповідає значенню true, "Cancel" – значенню false. Як правило, результат роботи функції привласнюють змінній, для подальшого аналізу, як це показано в прикладі вище.

2.14.Метод prompt

Метод prompt дозволяє вивести користувачеві діалогове вікно для введення даних. Використовується в тих випадках, коли користувач повинен ввести рядок тексту. Формат виклику:

Оголошення змінної

```
var str = prompt ("Запит на введення даних", значення_по_ замовчуванню);
```

Необхідно пам'ятати, що метод prompt повертає результат строкового типу. По-цьому, перш ніж його використовувати в арифметичних виразах, необхідно виконати перетворення типів до числових. Це можна зробити за допомогою наступних функцій:

parseInt("рядок") – перетворює рядок у ціле число;

parseFloat("рядок") – перетворює рядок в число з плаваючою точкою.

Розглянемо комплексний приклад, в якому для реалізації взаємодії з користувачем використовуються всі три методи, які описані вище. Зверніть увагу, що для введення числового значення, метод prompt використовується в комбінації з функцією parseInt.

Приклад 2.13. Використання методу prompt.

```
<script type="text/javascript">
var name=prompt("имя");
var last=prompt("отчество");
var fam=prompt("фамилия");
fio=name+" "+ last+" "+fam;
alert(fio);
</script>
```

2.15.Об’єкт document

Об'єкт document відповідає частині гіпертекстового документа, яка укладена в контейнер <body>. . . </ body>. Документи відображаються у вікнах браузера, тому кожен з них пов'язаний з певним вікном.

Всі HTML – об'єкти є властивостями об'єкта document, тому вони знаходяться в самому документі. Наприклад, в мові JS до першої форми документа можна звернутися, використовуючи вираз:document.forms[0].

Для звернення до властивостей і методів об'єкта document застосовується наступний синтаксис:

document.propertyName

document.methodName (parameters)

Приклад 2.14. Використання об'єкта document.

```
<html ><head>
<title> Document</title>
</head>
<body>
<script type="text/jscript">
document.write("Привіт");
</script>
</body>
</html>
```

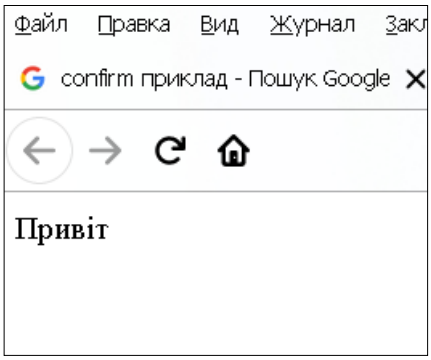


Рис.2.6 **Результат виконання прикладу 2.14**

2.15.1.Властивості об'єкту document

Об'єкт document має властивості, кожна з яких відповідає певному HTML– тегу в поточному документі:

- alinkColor – відповідає атрибуту alink тега <body>;
- anchors– масив, який відповідає всім мітках в документі;
- bgColor– відповідає атрибуту bgColor (колір фону) тега <body>;

- cookie– являє собою фрагмент інформації, записаний на локальний диск ("ключик");
- fgColor– відповідає атрибуту fgColor (колір тексту) тега <body>;
- forms– масив, що містить всі теги <form> в поточному документі;
- images– масив зображень, посилання на які задані в поточному документі;
- lastModified– дата останнього зміни поточного документа;
- linkColor– відповідає атрибуту linkColor (колір гіперпосилання по промовчуванню);
- links– масив, що містить всі гіперпосилання в поточному документі;
- location– відповідає адресі URL поточного документа;
- referrer– відповідає URL адресі документа, з якого користувач перейшов до поточного документа;
- title– відповідає вмісту контейнера <title>. . . </ Title>;
- vlinkColor– відповідає атрибуту vlinkColor (колір <FONT COLOR = "# 800080"> відвідується зв'язку) тега <body>

- Всі поля і методи Math статичні. Тобто до сталої π потрібно записати Math.PI, а функцію синуса викликати через Math.sin(x). Всі константи задані із максимальною для дійсних чисел у JavaScript точністю.

Таблиця 2.2.**Властивості об’єкту Math**

Методи Math	Значення
Math.E	Стала Ейлера, основа натуральних логарифмів. Приблизно дорівнює 2.718.
Math.LN2	Числове значення натурального логарифму від 2. Приблизно дорівнює 0.693
Math.LN10	Числове значення натурального логарифму від 10. Приблизно дорівнює 2.303.
Math.LOG2E	Логарифм від ее за основою 2,

	приблизно дорівнює 1.443.
Math.LOG10E	Логарифм від e e за основою 10, приблизно дорівнює 0.434.
Math.PI	Значення відношення довжини кола до його діаметру, наближено дорівнює 3.14159
Math.SQRT1_2	Квадратний корінь від 1/21/2; еквівалентно 12\frac {1}{\sqrt{2}}. Наближено дорівнює 0.707.
Math.SQRT2	Значення квадратного кореня від 2, наближено 1.414.

2.16. Математичні функції

Слід зазначити, що тригонометричні функції (sin(), cos(), tan(), asin(), acos(), atan(), atan2()) повертаються у радіанах. Для конвертації достатньо пам'ятати, що 1 кутовий градус - це (Math.PI / 180) радіан.

Точність багатьох математичних функцій залежить від конкретної реалізації платформи. Тобто одна і та ж функція у різних браузерах може дати дещо різний результат. Навіть більше - один і той самий JS-рушій на різних ОС чи архітектурах також може дати різні результати.

Таблиця 2.3. **Математичні функції**

Math.abs(x)	Повертає абсолютне значення (модуль) числа.
Math.acos(x)	Повертає арккосинус числа.
Math.acosh(x)	Повертає значення гіперболічного арккосинуса числа
Math.asin(x)	Повертає арксинус числа.
Math.asinh(x)	Повертає значення гіперболічного арксинуса числа
Math.atan(x)	Повертає арктангенс числа
Math.atanh(x)	Повертає значення гіперболічного арктангенса числа

Math.atan2(y, x)	Повертає значення арктангенсу частки поданих чисел.
Math.cbrt(x)	Повертає кубічний корінь числа.
Math.ceil(x)	Повертає число, округлене "до більшого".
Math.clz32(x)	Повертає кількість ведучих нулів 32-бітного цілочисельного уявлення даного числа.
Math.cos(x)	Повертає косинус числа.
Math.cosh(x)	Повертає значення гіперболічного косинуса числа
Math.exp(x)	Повертає результат обчислення exe^x, де xx - це аргумент функції, а ee - стала Ейлера (2.718…), основа натурального логарифму.
Math.expm1(x)	Повертає різницю exp(x) і 1
Math.floor(x)	Повертає результат округлення "до меншого".
Math.fround(x)	Повертає найближче число із рухомою комою (крапкою) одинарної точності від аргументу
Math.hypot([x[, y[, …]]])	Повертає квадратний корінь від суми квадратів аргументів
Math.imul(x, y)	Повертає результат 32-бітного цілочисельного множення аргументів
Math.log(x)	Повертає натуральний логарифм ($\log_e$, або ln) числа.
Math.log10(x)	Повертає логарифм за основою 10 від аргументу.
Math.log2(x)	Повертає логарифм за основою 2 від аргументу
Math.max([x[, y[, …]]])	Повертає найбільше із нуля чи більше аргументів
Math.min([x[, y[, …]]])	Повертає найменше із нуля чи більше аргументів.
Math.pow(x, y)	Повертає результат піднесення до степеня

	x^y
Math.random()	Повертає псевдовипадкове число з-поміж 0 і 1.
Math.round(x)	Повертає значення аргументу, округлене до найближчого цілого
Math.sign(x)	Повертає знак поданого числа. Визначає, чи являється аргумент додатним числом, від'ємним, чи дорівнює 0
Math.sin(x)	Повертає значення синуса аргументу.
Math.sinh(x)	Повертає значення гіперболічного синуса аргументу.
Math.sin(x)	Повертає значення синуса аргументу
Math.sinh(x)	Повертає значення гіперболічного синуса аргументу
Math.sqrt(x)	Повертає додатне значення квадратного кореня від аргументу
Math.tan(x)	Повертає значення тангенса аргументу.
Math.tanh(x)	Повертає значення гіперболічного тангенса аргументу
Math.trunc(x)	Повертає цілу частину аргументу, відкидаючи всю дробову частину

•

2.17.Методи JavaScript

Метод clear() призначений для очищення поточного документа. Краще використовувати для очищення методи open () і close (). Для запису інформації в браузер застосовують методи write () і writeln (). Оскільки ці методи записують текст в браузер в HTML –форматі, ви можете створювати будь-який HTML– документ динамічно, включаючи готові програми на мові JavaScript.

Якщо у вікно завантажений документ, то запис даних поверх нього може привести до збою. Тому в вікно слід записувати потік даних, для чого за допомогою методу document.open () потрібно відкрити документ, а потім,

викликавши необхідну кількість раз метод document.wtite (), записати дані в документ. В кінці слід викликати метод document.close ().

Приклад 2.15.Використання function alert().

```
<html ><HEAD>
<TITLE>The JavaScript's function alert()</TITLE>
</HEAD>
<BODY>
<H1>Begin</H1>
<SCRIPT>alert("Start");</SCRIPT>
<H1>Middle</H1>
<SCRIPT>alert("Finish");</SCRIPT>
<H1>End</H1>
</BODY>
</HTML>
```

Рис.2.7 Результат виконання прикладу 2.16

Приклад 2.16 Використання function write() AND writeln().

```
<html >
<HEAD>
<TITLE>The JavaScript's function write() AND writeln()</TITLE>
</HEAD>
<BODY>
<H1>Begin</H1>
<SCRIPT>
document.open();
document.write("Rezult<BR>");
document.writeln("Rezult");
document.write("Rezult<BR>");
```

```
document.write("<PRE>Rezult<BR>");
document.writeln("Rezult");
document.write("Rezult<BR></PRE>");
document.close();
</SCRIPT>
<H1>End</H1>
</BODY>
</HTML>
```

Приклад 2.17 Використання об’єкту "string".

```
<html >
<HEAD>
<TITLE>The JavaScript's object "string"</TITLE>
</HEAD>
<BODY>
<H1>Methods of the <I>string</I> object</H1>
<SCRIPT>
var x = "test" + "<BR>"; / * Створюємо об'єкт з ім'ям x * /
alert (x);
document.write (x.big ()); / * збільшуємо літери * /
document.write (x.italics ()+'<br>'); / * похилий * /
document.write ('-SUB-' + x.sub ()); / * надрядковий * /
document.write (x.bold ()); / * напівжирний * /
document.write ('- SUP -' + x.sup ()); / * підрядковий * /
document.write (x.fontcolor ('blue')); / * Синій колір шрифту * /
document.write (x.fontsize (1)); / * Розмір по відношенню до базового * /
document.write(x.fontsize(2));
document.write(x.fontsize(3));
document.write(x.fontsize(4));
document.write(x.fontsize(5));
document.write(x.fontsize(6));
```

```
document.write(x.fontsize(7));
</SCRIPT></HTML>
```

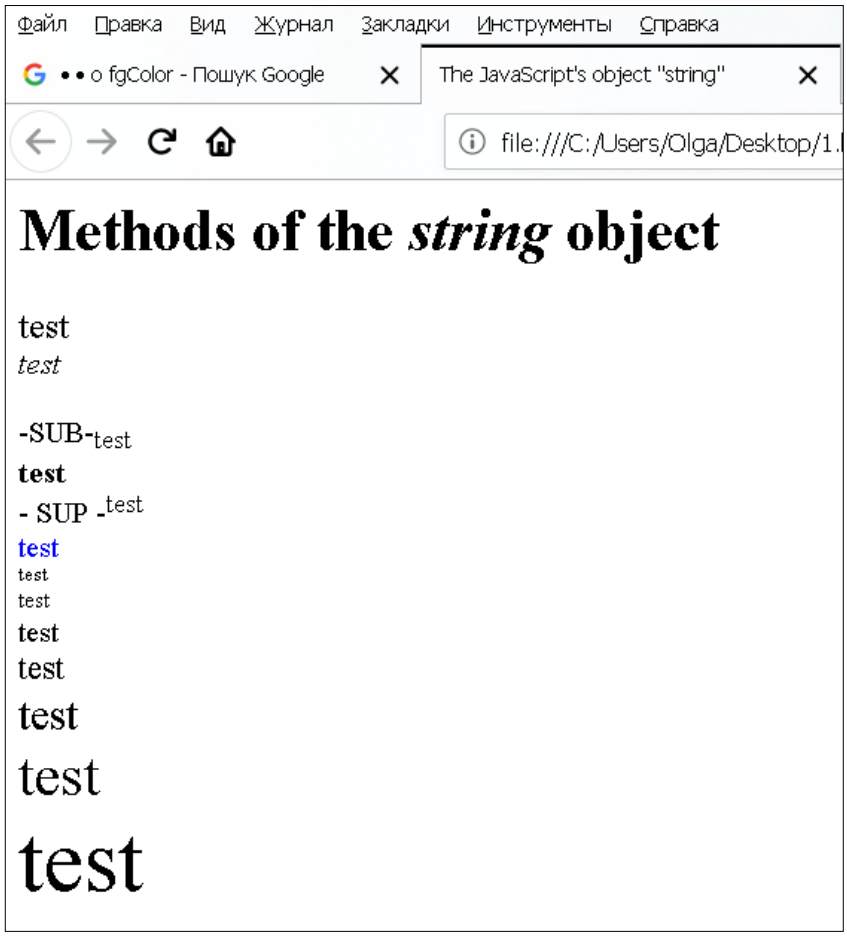


Рис.2.8 **Результат виконання прикладу 2.17**

Приклад 2.18 Використання властивостей шрифтів

```
<html >
<HEAD>
<TITLE>The JavaScript's Strings</TITLE>
</HEAD>
<BODY>
<SCRIPT>
x="Input Name:";
document.write( x );
document.write( x.big() );
document.write( x.bold() );
document.write( x.italics() );
```

```
document.write( (x.big()).italics() );
document.write( "<br>" );
document.write( x.fontcolor("iceblue") );
document.write( x.fontsize(1) );
document.write( x.fontsize(2) );
document.write( x.fontsize(3) );
document.write( x.fontsize(4) );
document.write( x.fontsize(5) );
document.write( x.fontsize(6) );
document.write( x.fontsize(7) );
</SCRIPT>
</BODY>
</HTML>
```

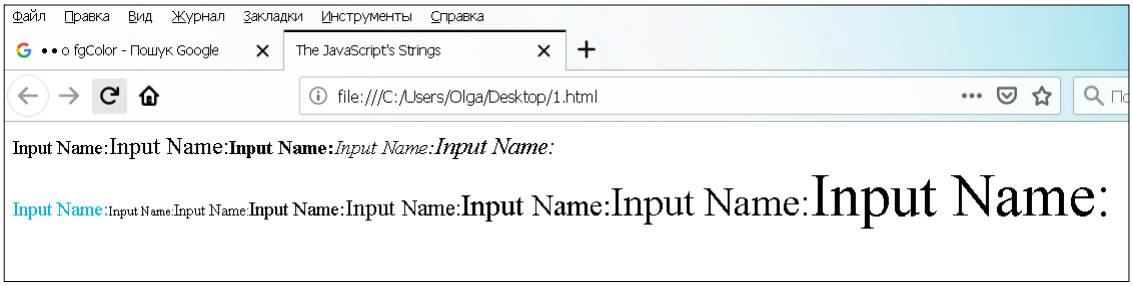


Рис.2.9 **Результат виконання прикладу 2.18**

2.18.Функції

Функції – це механізм для багаторазового використання частин коду. Вони дозволяють запускати один і той же код з різних місць програми, без необхідності його копіювати.

Функції дуже зручні, коли потрібно багато разів виконувати в програмі розрахунки або інші дії. Готові, вбудовані функції, це такі як: Math.random, Math.floor, alert, prompt, confirm. Можна створювати свої функції.

Оголошення функції.

Для створення функцій ми можемо використовувати оголошення функції. Приклад оголошення функції:

```
function showMessage() {
alert( 'Всім привіт!' );
}
```

Спочатку записується ключове слово function, після нього ім'я функції, потім список параметрів в круглих дужках через кому (у наведеному прикладі він порожній) та код функції, також званий «тілом функції», всередині фігурних дужок.

```
function ім'я (параметри) {
тіло  функції.          }
```

Наприклад:

```
function showMessage() {
alert( 'Всім привіт!' );
}

showMessage();// виклик функції
```

Змінна, оголошена оператором var поза функцією, є глобальною – вона "видна" всюди в скрипті. Змінна, оголошена оператором var всередині будь-якої функції, є локальною – вона "видна" тільки в межах цієї функції.

Наприклад, в наступному фрагменті нічого не буде виведено на екран, оскільки змінна k описана в середині функції:

```
function f ()
{Var k = 5; }
f ();
alert (k);
```

В даному прикладі змінна k є локальною, вона існує тільки в процесі роботи функції f (), а після закінчення її роботи знищується.

2.18.1.Передача аргументів в функцію

Щоб функція могла змінювати поведінку в залежності від значень, використовуються аргументи. Список аргументів вказується в дужках після імені функції – як при її створенні, такі при виклику. Функція f1 використовує аргумент (argument).

Приклад 2.19 Приклад функції.

```
function f1( argument ) {
alert("Передан аргумент: " + argument);
}
document.write (f1(5));
```

2.18.2.Передача в функцію декількох аргументів

У функцію можна передати декілька аргументів. Для цього слід перерахуйте аргументи в дужках після імені функції, розділивши їх комами.

Приклад 2.20. Передача в функцію двох аргументів

```
function f1( argument1,argument2 ) {
alert("Передан аргумент: " + argument1+argument2);
}
document.write (f1(5,9));
```

Приклад 2.21 Передача в функцію декількох аргументів.

```
<html><head><title>Untitled Document</title>
</head>
<body>
<script>
function showMessage(from, text) {
  if (text === undefined) {
    text = 'Іваненко';
  }
  alert( from + ": " + text );
}
</script>
```

<form>

Введіть прізвище<input type='text' name='text' /><p>

Введіть групу<input type='text' name='from' />

<input type='button' value="ok" onclick="showMessage(form.from.value, form.text.value);" >

</form></body></html>

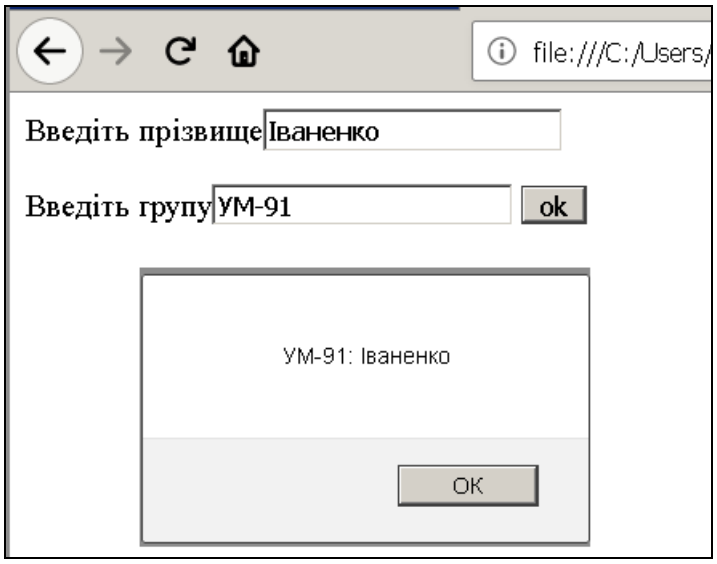


Рис.2.10 **Результат виконання прикладу 2.21**

2.18.3.Повернення значення з функції

Функції можуть повертати значення. Викликавши функцію, яка повертає значення, ми можемо потім використовувати це значення в своїй програмі

Приклад 2.22. Передача в функцію одного аргументу.

```
var double = function (number) {
return number * 2;
};

document.write(double(5));
```

2.18.4.Виклик функції

Коли функція викликається з коду програми, значення, що повертається цією функцією, підставляється туди, де відбувається виклик.

Приклад 2.23. Виклик функції.

```
var double = function (number) {
return number * 2;
};

document.write(double(5)+double(10));
```

Приклад 2.24. Декілька викликів return

```
function checkAge(age) {
  if (age > 18) {
    return true;
  } else {
    return confirm('А батьки дозволили?');
  }
}
let age = prompt('Скільки вам років?', 18);
if ( checkAge(age) ) {
  alert( 'Доступ получен' );
} else {
  alert( 'Доступ закрыт' );
}
```

Приклад 2.25 Використання змінних.

```
<html>
<head>
<title>Змінні</title>
</head>
<body>
<div>
<script type="text/javascript">
var d=365;
```

var z="Земля";

var d1="7млд";

var o="Сонце";

name=" Ми живемо на планеті "+z+ ", вона робить один оборот навколо "+ o +" за "+ d +" днів. "+"Населення нашої планети становить приблизно" + d1 + "людина."

document.write (name);</script>

</body>

</html>

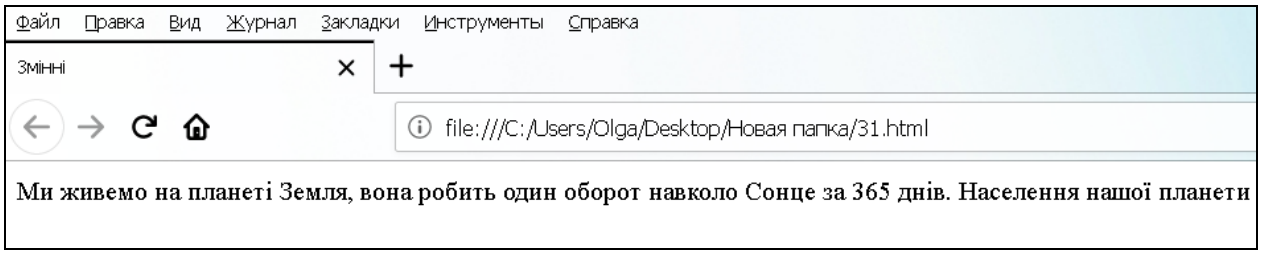


Рис.2.11 Результат виконання прикладу 2.24

Приклад 2.26. Створення програми, в якій при натисканні на кнопку в текстовому полі з'являється ім'я.

```
<body>
<h1>Задачі по JavaScript</h1>
<form name="form1">
 <input type="text" name="input1"/>
 <INPUT TYPE=button onClick="form1.input1.value='olga'">
</form>
</body>
</html>
```

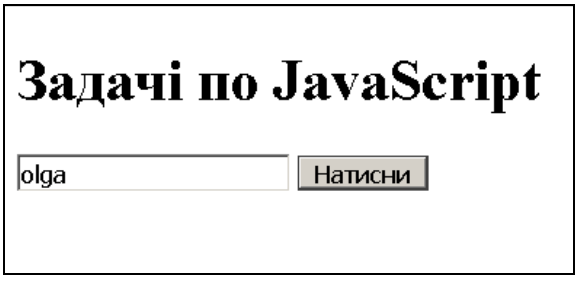


Рис.2.12 **Результат виконання прикладу 2.25**

Приклад 2.27 Створення програми, в якій при натисканні на кнопку виводить текст в текстовому полі.

<html ><head>

```
<title>Untitled Document</title>
</head>
<body>
<h1>Задачі по JavaScript</h1>
<script type="text/jscript">
function o(){
alert(form1.input1.value);
}
</script>
<form name="form1">
<input type="text" name="input1" />
<INPUT TYPE=button  value="Натисни" onClick="o();">
</form>
</body>
</html>
```

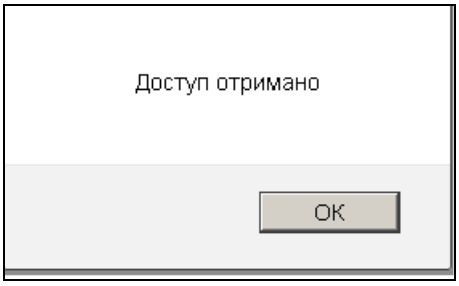


Рис.2.13 **Результат виконання прикладу 2.27**

Приклад 2.28. Програма запитує ваш вік, якщо він більше 18, з'являється повідомлення Доступ отримано.

```
<html><head>
<title>Untitled Document</title>
</head>
<body>
<script type="text/jscript">
function ces(age) {
if (age > 18) {
return alert('Доступ отримано' );;
} else {
return confirm('А батьки дозволили?');
}}
```

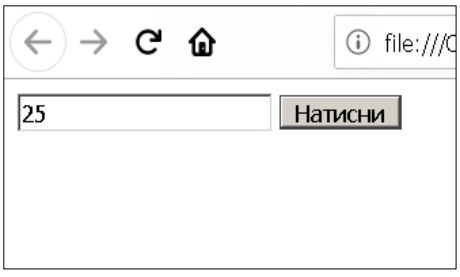
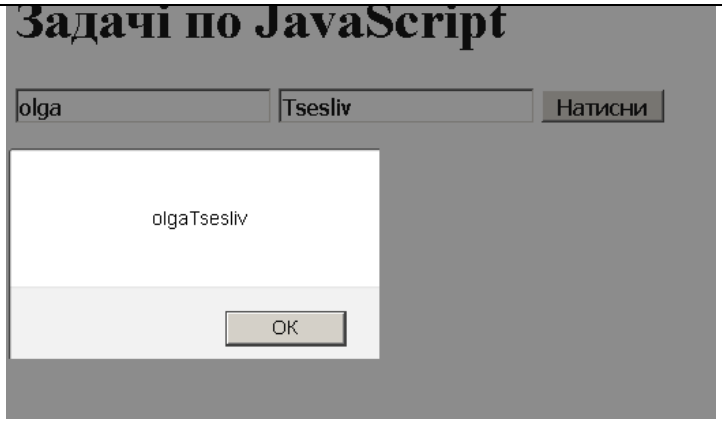


Рис.2.14 **Результат виконання прикладу 2.28**

```
</script>
<form name="form1">
<input type="text" name="input1" />
<INPUT TYPE=button  value="Натисни" onClick="ces(form1.input1.value);">
</form></body></html>
```

Завдання для самостійної роботи

Вариант№1	Завдання №1 Ваше завдання полягає в тому, щоб створити чотири змінних (назви вибирайте на свій розсуд). Першу - для зберігання кількості днів. Дайте їй значення - 5. Другу - для зберігання назви нашої групи. Третю - для зберігання кількості людей, надайте їй значення 25. Четверту - для зберігання слова ФММ. Далі, використовуючи текст і змінні, Ви повинні вивести на екран такий абзац: "Ми вчимося на ФММ, в групі 25 чоловік, відмінників 5 чоловік ”. Завдання №2 Створити функцію fio (), яка по черзі запитує у користувача ім'я, прізвище та по батькові, а в якості результату своєї роботи повертає ПІБ одним рядком з пробілами між словами.
Вариант№2	Завдання №1 Створюємо об'єкт з ім'ям х =’ Факультет менеджменту та маркетингу’. Роздрукувати:великими літерами, похилим шрифтом, напівжирний, синій шрифтом. Завдання №2 Написати програму, що містить форму. Вписати ім’я та прізвище в текстове поле та вивести вікном повідомлення.

	
Вариант№3	Завдання №1 Написати програму, в якій задайте людині питання через метод prompt (). Правильну відповідь заздалегідь помістіть в змінній var answer. Якщо відповідь вірна, то виведіть повідомлення, відповідь вірна або невірна. Завдання №2 Написати програму, що містить форму. Вписати ім'я, по-батькові та прізвище в текстове поле та вивести вікном повідомлення.

РОЗДІЛ 3 УМОВНИЙ ОПЕРАТОР

У цьому розділі будуть розглянуті умовні оператори JavaScript, які застосовується для розгалуження програми по деякій логічній умові.

3.1.Синтаксис умовного оператора

Загальний синтаксис умовного оператора має два варіанти синтаксису:
Короткий умовний оператор

if (логічний_вираз) оператор;

Повний умовний оператор

if (логічний_вираз) оператор_1; else оператор_2.

Логічний вираз – це вираз, який приймає значення true або false. У першому варіанті синтаксису (3.1): якщо логічний_вираз дорівнює true, то виконується вказаний оператор. У другому варіанті синтаксису: якщо логічний_вираз дорівнює true, то виконується оператор_1, якщо ж логічний_вираз має значення false оператор_2.

Приклад 3.1 Використання умовного оператора.

```
<script type="text/javascript">
{
var result=confirm("Ви вивчаєте JavaScript?");

if  (result)
alert ( 'Це добре');
else
 alert ( 'Вивчайте JavaScript');
}
</script>
```

Приклад 3.2. Знаходження найбільшого з трьох чисел.

```
<html><head>
<script language="JavaScript">
```

```
<!--
function verify(form)
{
var a=document.form.a.value;
var int=parseInt(a);
var b=document.form.b.value;
var int1=parseInt(b);
var c=document.form.c.value;
var int2=parseInt(c);
   //Знаходимо найбільше з трьох чисел
if ( a >=b && a >=c )  {alert(a);}
if ( b>=a && b>=c ) {alert(b);}
if (  c>=a && c>=b )  {alert(c);}
a=0;// Обнуляємо значення
b=0;// Обнуляємо значення
c=0;// Обнуляємо значення
document.form.a.value="";
document.form.b.value="";
document.form.c.value="";
}
//-->
</script>
</head>
<body >
<p>Введіть три числа:
<form  name="form" >
<input name="a" size="2"><br>
<input name="b" size="2"><br>
<input name="c" size="2"><br>
```

<input type="button" onClick="verify()" value="Підрахунок">

</form>

</body>

</html>

3.2.Знаходження інтегралу функції

Метод прямокутників – метод чисельного інтегрування функції однієї змінної. Якщо розглянути графік підінтегральної функції, то метод буде полягати в наближеному обчисленні площі під графіком. Площа розбивається на прямокутники, ширина яких буде визначатися відстанню між відповідними сусідніми вузлами інтегрування, а висота – значенням підінтегральної функції в цих вузлах.

$$\int_a^b f(x)dx \approx f(a)(b-a) \qquad (3.3)$$

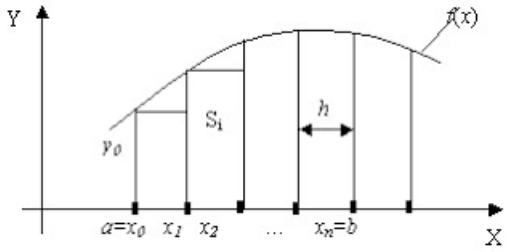


Рис.3.1 Графік підінтегральної функції

Приклад 3.3 Знайти інтеграл функції $f(x) = \int_0^1 e^x dx$

<html >

<head>

<title>Untitled Document</title>

</head>

<script type="text/javascript">

var h=0;

h = (1 - 0) / 20;

var s=0;

```
var i=0;
var x;
for(x=0;x<=20; x++)
{
s = s + Math.exp(i);
document.write(Math.exp(i)+"_____ "+i+"<br>");
i=i+0.05;
}
document.write(s*h);
</script>
<body>
</body>
</html>
```

3.3.Обчислення коренів квадратного рівняння

Припустимо є квадратне рівняння:

$$ax^2 + bx + c = 0$$

Розв’язок квадратного рівняння:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}, \qquad \text{якщо} \quad D = b^2 - 4ac \geq 0$$

$$x_{1,2} = \alpha \pm i\beta \ , \qquad \text{якщо} \quad D = b^2 - 4ac \leq 0 .$$

Приклад 3.4 Обчислення коренів квадратного рівняння**.**

```
<html>
<head>
<title> Обчислення коренів квадратного рівняння </title>
</head>
<body>
<script type="text/javascript">
var a=1;
```

```
var b=4;
var c=1;
d=Math.pow(b,2)-4*a*c;
document.write(d);
if(d>0){
x1=b+Math.sqrt(b*b-4*a*c);
document.write("x1="+x1+"<br>");
x2=b-Math.sqrt(b*b-4*a*c);
document.write("x2="+x2+"<br>");
}
else
alert("ytn");
</script>
</body>
</html>
```

Приклад 3.5. Програма по номеру місяця виводить пору року.

```
<html>
<head>
<script>
<!--
      function verify()
      {
         //Введення рядка
         var str=document.inp.m.value;
         //Перетворення рядка в число
         var int=parseInt(str);
         //Перевірка
         if ( int == 1 || int == 2 || int == 12 )
            {alert("Зима");
```

```
            return true;}
        else if ( 3 <= int && int <= 5 )
            {alert("Весна");
            return true;}
        else if ( 6 <= int && int <= 8 )
            {alert("Літо");
            return true;}
        else if ( 9 <= int && int <= 11 )
            {alert("Осінь");
            return true;}
        else
            {alert("Невірно!");
            return false;}
    }
    //-->
  </script>
</head>
<body>
<H2>Контроль введення</H2>
<form name="inp" OnSubmit="return verify()">
  <label>Введіть номер місяця:
    <input name="m" size="2">
  </label>
</form>
</body>
</html>
```

Завдання для самостійної роботи

Варіант№1	1. Обчислити корені квадратного рівняння $2x^2+4x+1=0$ 2. Ввести чотиризначне число знайти найменше. 3. Знайти значення інтегралу $\int_0^1 (e^x+1)dx$
Варіант№2	1.Обчислити корені квадратного рівняння $4x^2+3x+5=0$ 2.Ввести чотиризначне число знайти найбільше. 3. Знайти значення інтегралу $\int_0^1 (1+\frac{1}{1+x})dx$
Варіант№3	1. Обчислити корені квадратного рівняння $x^2+2x+1=0$ 2. Ввести чотиризначне число знайти найменше. 3. Знайти значення інтегралу $\int_0^1 (\frac{1}{1+x})dx$
Варіант№4	1. Обчислити корені квадратного рівняння $5x^2+11x+4=0$ 2. 2.Ввести чотиризначне число знайти найбільше. 3. Знайти значення інтегралу $\int_0^1 (\frac{1}{e^x+1})dx$

РОЗДІЛ 4. ЦИКЛИ

4.1.Оператори циклів в JavaScript

Цикли дозволяють в залежності від певних умов виконувати певну дію безліч разів. У мові JavaScript є такі види циклів:

- for
- while
- do...while

4.2.Оператор циклу while

Синтаксис оператора циклу while

while (умова_продовження_циклу) тіло_циклу;

Тіло_циклу може бути простим або складеним оператором. Складений оператор обов'язково пишеться у фігурних дужках. *Умова_ продовження_циклу* є логічним виразом. Тіло циклу, повторюється до тих пір, поки виконується умова. Формально, цикл при роботі працює наступним чином:

1. перевіряється умова_продовження_циклу:

якщо вона хибна (false), цикл завершується,

якщо ж умова вірна (true), продовжуємо розрахунки далі;

2. виконується тіло_циклу ;
3. переходити до пункту 1.

Такий цикл застосовується, коли невідома кількість ітерацій.

Приклад 4.1. Використання циклу while.

```
<script type="text/jscript">
var s='';  // оголошуємо змінну s
while (s.length<6)   //цикл, який перевіряє довжину рядка
{
s=prompt('Ввдіть рядок довжиною не менше 6:','');  // вводимо рядок
}
```

```
alert('Ваш рядок: ' + s + '.Дякую!');  //вікно повідомлення
</script>
```

4.3.Оператор циклу FOR

Синтаксис оператора циклу for:

```
for ([ініціалізація лічильника]; [умова]; [зміна лічильника])
{
// Тіло циклу
}
```

Цикл For працюється наступним чином:

1. виконується ініціалізація_змінних_циклу ;
2. перевіряється умова_продовження_циклу :
   якщо умова хибна (false), цикл закінчується,
   якщо ж правда (true), то продовжуємо далі;
3. виконується тіло_циклу ;
4. виконуємо модифікацію_змінних_циклу ;
5. переходимо до пункту 2.

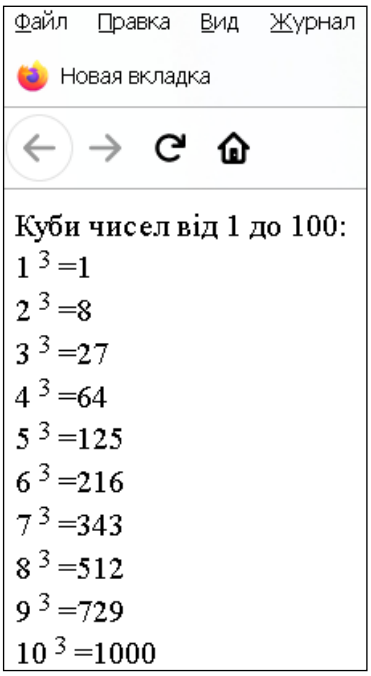


Рис.4.1 Результати виконання прикладу 4.1.

Приклад 4.2. Роздруковати куби чисел від 1 до 100

```
<script type="text/jscript">
```

```
document.write ('Куби чисел від 1 до 100:');
for(n = 1; n <= 100; n ++)
document.write ('<BR>' + n + '<sup> 3 </sup> =' + Math.pow (n, 3));
</script>
```

Math – вбудований об'єкт, який представляє багаточисельні математичні константи та функції, а Math.pow (n, m) виходить на ступінь функціональності n^m.

Приклад 4.3 Обчислити функції.

$$y = \begin{cases} a^2 + 5 & x < 3 \\ 1 + 1/a & 3 < x < 7 \\ a^2 + (1/a) & x > 7 \end{cases}$$

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Untitled Document</title>
</head>
<body>
<script>
var x=1;
var y;
var a=1;
alert(x);
for(x=1;x<=8;x++)
{
if(x<3) y=Math.pow (a, 2)+5;
if(x>=3 & x<=7) y=1+1/a;
if(x>7) y=Math.pow (a, 2)+(1/a);
document.write('<br>');
document.write("x=",x,"\t      ","y=",y);
```

x=1 y=6
x=2 y=6
x=3 y=2
x=4 y=2
x=5 y=2
x=6 y=2
x=7 y=2
x=8 y=2

Рис.4.2 Результати виконання прикладу 4.3.

```
}
</script>
</body>
</html>
```

4.4.Оператор break

Оператор break дозволяє завчасно покинути тіло циклу. Повертаємося до нашого прикладу з кубами чисел, розглядаємо куби чисел, значення яких <5000 .

Приклад 4.4. Роздрукуємо куби чисел, значення яких <5000 .

```
<script type="text/jscript">
document.write('Кубы чисел, меньшие 5000:');
for (n=1; n<=100; n++)
{
 s=Math.pow(n,3);
 if(s>5000) break;
 document.write('<BR>'+n+'<sup>3</sup> = '+s);
 }
</script>
```

Незважаючи на те, що змінну n ми змусили пробігати від 1 до 100, тобто свідомо з запасом, реально ж цикл виконається для значень n від 1 до s>5000

4.5.Оператор continue

Оператор continue дозволяє перейти до наступної ітерації циклу, пропустивши виконання всіх операторів, які стоять нижче, в тілі циклу.

Приклад 4.5. Потрібно вивести куби чисел від 1 до 100, що перевищують 10 000.

```
<script type="text/jscript">
document.write('Кубы чисел от 1 до 100, большие 10 000:');
for (n=1; n<=100; n++)
{
```

```
 s=Math.pow(n,3);
 if(s <= 10000) continue;
document.write('<BR>'+n+'<sup>3</sup> = '+s);
}
</script>
```

Для більшої гнучкості можна використовувати в циклах обидва оператори break і continue.

4.6.Оператор return

Оператор return використовують для повернення значення з функції.

Приклад 4.6. Виведення знаку функції (n).

```
function sign(n)
{
 if (n>0) return 1;
 if (n<0) return -1;
 return 0;
}

alert( sign(-3) );
```

Зверніть увагу: оператор return не тільки вказує, яке значення має повернути функція, але і припиняє виконання подальших операторів в тілі функції.

Приклад 4.7. Використання функцій.

```
<FORM ACTION="newpage.html" METHOD=post>
<INPUT TYPE=submit VALUE="Отправить?"
onClick="alert('Не отправим!'); return false;">
</FORM>
```

У цьому прикладі без оператора return false користувач побачив би вікно попередження "Не відправимо!" і далі був би перенаправлений на

сторінку newpage.html. Оператор же return false дозволяє скасувати відправку форми, і користувач лише побачить вікно попередження.

Аналогічно, щоб скасувати дію за замовчуванням для параметрів подій onClick, onKeyDown, onKeyPress, onMouseDown, onMouseUp, onSubmit, onReset, потрібно використовувати return false. Для події onMouseOver з цією ж метою потрібно використовувати оператор return true. Для деяких же подій, наприклад onMouseOut, onLoad, onUnload, скасувати дію за замовчуванням неможливо.

Приклад 4.8. Програма з картинками, які змінюються.

```
<script type="text/jscript">
function onOver(){
document.images["a"].src="2.jpg";
}
</script>
<img name="a" src="1.bmp" onmousedown="onOver()";
</script>
```

Приклад 4.9 Знайти суму чисел.

```
<script type="text/jscript">
<!--
s=0;
i=0;
d=prompt('Введіть число', );
while(i<d)
{
s=s+i;

i=i+1;
document.write('s'+s+"i"+i);
}
```

```
alert(s);
-->
</script>
```

4.7.Оператор циклу do

Цикл **do** спочатку виконує код циклу, а потім перевіряє умову в інструкції while. Цикл повторюється поки умова істинна.

Приклад 4.10. Використання циклу do.

```
<script>
j = 7;
do{
   document.write(j+"<br>");
   j--;
}
while (j > 0);
</script>
```

В даному випадку код циклу спрацює 7 разів, поки j не стане рівним нулю. Важливо відзначити, що цикл do гарантує хоча б одноразове виконання дій, навіть якщо умова в інструкції while не буде істина.

Приклад 4.11. Цикл do гарантує одноразове виконання дій.

```
<script>
 j = -1;
do{
   document.write(j+"<br>");
   j--;
}
while (j > 0);;
</script>
```

Хоча змінна j спочатку менше 0, цикл все одно один раз виконається.

Завдання для самостійної роботи

Варіант№1	1. Вивести числа від 1 до 10 2. Скласти програму, в якій є дві картинки, що змінюються при проведенні миші. 3. Знайти факторіал 15!
Варіант№2	1. Вивести числа від 10 до 20 2. Скласти програму, в якій є дві картинки, що змінюються при проведенні миші. 3. Знайти добуток чисел від 10 до 100
Варіант№3	1. Вивести квадрати чисел від 1 до 10 2. Скласти програму, в якій є дві картинки, що змінюються при проведенні миші. 3. Знайти факторіал 10!
Варіант№4	1. Вивести куби чисел від 1 до 10 2. Скласти програму, в якій є дві картинки, що змінюються при проведенні миші. 3. Знайти суму чисел від 100 до 10
Варіант№5	1. Вивести квадрати чисел від 10 до 100 2. Скласти програму, в якій є дві картинки, що змінюються при проведенні миші. 3. Знайти факторіал 9!

Варіант№6	1. Вивести куби чисел від 13 до 15 2. Скласти програму, в якій є дві картинки, що змінюються при проведенні миші. 3. Знайти факторіал 5!
Варіант№7	1. Вивести квадрати чисел від 50 до 100 2. Скласти програму, в якій є дві картинки, що змінюються при проведенні миші. 3. Знайти факторіал 5!
Варіант№8	1. Вивести куби чисел від 11 до 90 2. Скласти програму, в якій є дві картинки, що змінюються при проведенні миші. 3. Знайти добуток чисел від 10 до 100

РОЗДІЛ 5 МАСИВИ

Масив – це пронумерована послідовність величин однакового типу, що позначається одним ім'ям. Елементи масиву розташовуються в послідовних комірках пам'яті, позначаються ім'ям масиву та індексом. Кожне із значень, складових масив, називається його компонентою (або елементом масиву).

Масиви мають спеціальну внутрішню організацію. JavaScript зберігає елементи масиву в безперервній області пам'яті, один за іншим. Існують і інші способи оптимізації, завдяки яким масиви працюють дуже швидко.

Але всі вони втратять ефективність, якщо ми перестанемо працювати з масивом як з «впорядкованої колекцією даних» і почнемо використовувати його як звичайний об'єкт.

5.1.Оголошення масиву

Існує два варіанти синтаксису створення масиву:

var con=["Україна ","Молдова", "Польща","Румунія", "Угорщина"];

var n=[47,157,70,45,36];

Елементи масиву нумеруються, починаючи з нуля.

Індекс – це номер елемента, в якому зберігається значення. Звернутися до елементу масиву можемо, вказавши назву масиву та його індекс у квадратних дужках:

```
fruits = ["Яблуко", "Апельсин", "Слива"];

alert (fruits [0]); // Яблуко // виводимо елемент масиву
alert (fruits [1]); // Апельсин//
alert (fruits [2]); // Слива//
```

Ми можемо замінити елемент:

```
fruits [2] = 'Груша'; // тепер ["Яблуко", "Апельсин", "Груша"]
```

Список елементів масиву, як і список властивостей об'єкта, може закінчуватися комою:

```
var fruits = [
  "Яблуко",
  "Апельсин",
  "Слива",
];
alert(fruits[2]);
```

У масиві можуть зберігатися елементи будь-якого типу.

Приклад 5.1. Виведення елементу масиву.

```
<script type="text/javascript">
var arr = [ 'Яблуко', { name: 'Джон' }, true, function() { alert('привіт'); } ];
alert( arr[3] );
</script>
```

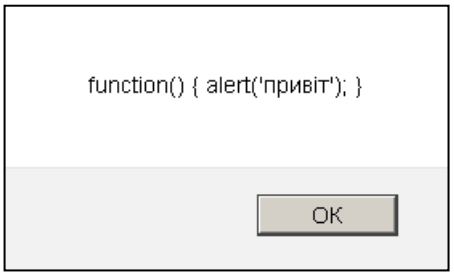


Рис.5.1. Результати виконання прикладу 5.1.

5.2.Методи pop / push

Черга – один з найпоширеніших варіантів застосування масиву. В області комп'ютерних наук так називається впорядкована колекція елементів, що підтримує два види операцій:

push додає елемент в кінець.

shift видаляє елемент на початку, зрушуючи чергу, так що другий елемент стає першим.

Приклад 5.2. Приклад використання методу push**.**

Додає елемент в кінець масиву:

```
var fruits = ["Яблуко", "Апельсин"];
fruits.push ("Груша");
alert (fruits); // Яблуко, Апельсин, Груша
```

Приклад 5.3 Приклад використання методу shift**.**

```
<script type="text/javascript">
var fruits = ["Яблуко", "Апельсин", "Груша"];
alert( fruits.shift() ); // видаляємо  Яблуко и виводимо масив
alert( fruits ); // Апельсин, Груша
</script>
```

5.3.Приклади використання методів pop, push

Приклад 5.4 Приклад використання методу pop.

Видаляє останній елемент з масиву і виводимо його:

```
var fruits = ["Яблуко", "Апельсин", "Груша"];
alert (fruits.pop ()); // видаляємо "Груша" і виводимо масив
alert (fruits); // Яблуко, Апельсин
```

Приклад 5.5 Приклад використання методу push.

Додає елемент в кінець масиву:

```
Var  fruits = ["Яблуко", "Апельсин"];
fruits.push("Груша");
alert( fruits ); // Яблуко, Апельсин, Груша
```

5.4.Методи shift, unshift

Приклад 5.6 Приклад використання методу shift.

Видаляє з масиву перший елемент і виводить масив:

```
<script type="text/javascript">
var fruits = ["Яблуко", "Апельсин", "Груша"];
alert( fruits.shift() ); // видаляємо Яблуко и виводимо масив
alert( fruits ); // Апельсин, Груша
</script>
```

Приклад 5.7. Приклад використання методу Unshift.

Додає елемент на початок масиву:

```
var fruits = ["Апельсин", "Груша"];
fruits.unshift('Яблуко');
alert( fruits ); // Яблуко, Апельсин, Груша
```

Методи push и unshift можуть додавати відразу декілька елементів:

```
var fruits = ["Яблуко"];

fruits.push("Апельсин", "Груша");
fruits.unshift("Ананас", "Лимон");

// ["Ананас", "Лимон", "Яблуко", "Апельсин", "Груша"]
alert( fruits );
```

5.5.Виведення елементів масиву

Одним з найстаріших способів перебору елементів масиву є цикл for по цифровим індексам:

```
{
var arr = ["Яблуко", "Апельсин", "Груша"];
for (let i = 0; i < arr.length; i++) {
alert( arr[i] );
}
```

Приклад 5.8. Виведення двох масивів.

```
<script type="text/javascript">
var con=["Україна ","Молдова", "Польща","Румунія", "Угорщина"];
var n=[47,157,70,45,36];
var i=1;
while (i<con.length){
document.write(con[i]+"-"+n[i]+"<br />");
i++}
</script>
```

Властивість length автоматично оновлюється при зміні масиву. Якщо бути точними, це не кількість елементів масиву, а найбільший цифровий індекс плюс один.

Приклад 5.9.Використання властивості length.

```
<script type="text/javascript">
var fruits = [];
fruits[123] = "Яблуко";
alert( fruits.length ); // 124
</script>
```

Приклад 5.10 Виведення елементів масиву.

```
{
var students = ["Абаканов Сергій", "Гагарин Матвій", "Хантухова Лейла", "Алаліна Аліна",  "Сакуров Иоган"];
var i=1;
while (i<students.length){
document.write(students[i]+"<br />");
i++;
}
```

Якщо ми вручну збільшимо масив, нічого цікавого не станеться. Зате, якщо ми його зменшимо, масив стане коротшим. Цей процес є незворотнім, ми можемо зрозуміти з прикладу 4.10:

Приклад 5.11 Зменшення довжини масиву елементів масиву.

```
<script type="text/javascript">
var  arr = [1, 2, 3, 4, 5];
arr.length = 2; // укорочуємо до двох елементів
alert( arr ); // [1, 2]
arr.length = 5; // повертаємо length як було
alert( arr[3] ); // undefined: значення не відновилися
</script>
```

5.6.Створення масиву – new Array ()

Існує ще один варіант синтаксису для створення масиву:

```
var  arr = new Array ("Яблуко", "Груша", "і тд");
```

Він рідко застосовується, так як квадратні дужки [] коротше. Крім того, у нього є особливість. Якщо new Array викликається з одним аргументом, який являє собою число, він створює масив без елементів, але із заданою довжиною.

5.7.Об'єднання декількох масивів

Щоб «склеїти» два масиви, створивши таким чином новий масив, використовується команда firstArray.concat(otherArray). Метод concat створює масив, в якому елементи з firstArray будуть розташовані перед елементами otherArray.

Приклад 5.12. Об'єднання масивів.

```
<script>
var furryAnimals = ["Собака", "Кішка", "Миша"];
var scalyAnimals = ["Удав", "Мавпа"];
var furryAndScalyAnimals = furryAnimals.concat(scalyAnimals);
for(i=0;i<=furryAndScalyAnimals.length-1;i++){
document.write(furryAndScalyAnimals[i]+"<br>");
}
</script>
```

Приклад 5.13 Створення масиву – new Array ()

```
<script type="text/javascript">
var arr = new Array(2); // створюється масив [2]
alert( arr[0] ); // undefined! немає елементів.
alert( arr.length ); // length 2
</script>
```

5.8.Пошук індексу елемента в масиві

Щоб з'ясувати, який у певного елемента індекс в масиві, використовується .indexOf ("елемент").

Приклад 5.14.Пошук елемента в масиві.

```
<script>
var colors = ["червоний", "зелений", "синій"];
```

```
document.write(colors.indexOf("червоний"));
</script>
```

5.9.Перетворення масиву в рядок

Скориставшись методом .join (), можна з'єднати всі елементи масиву в один великий рядок.

Приклад 5.15. Перетворення масиву в рядок

```
<script>
var colors = ["червоний", "зелений", "синій"];
document.write(colors.join());
</script>
```

5.10.Багатовимірні масиви

Масиви можуть містити елементи, які теж є масивами. Це можна використовувати для створення багатовимірних масивів, наприклад, для зберігання матриць:

Приклад 5.16. Багатовимірні масиви.

```
<script type="text/jscript">
var arr = [
  [1,2,3],
  [4,5,6],
  [7,8,9]
];
for(var j = 0; j < arr.length; j++) {
  for(var n = 0; n < arr.length; n++) {
    if (n === arr.length - 1)
    document.write(arr[j][n]);
    else
    document.write(arr[j][n] + ", ");
  }
  document.write("<br>");
}
```

```
</script>
```

Приклад 5.17. Сортування числового масиву функцією sort().

```
<script type="text/jscript">
var a = [2,3,7,5,3,7,1,3,4];
a.sort();
for(var i=0;i<8;i++){
document.write(a[i]);
}
</script>
```

Приклад 5.18. Сортування числового масиву.

```
 <html><head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Untitled Document</title>
</head>
<body>
<script type="text/jscript">
function BubbleSort(A)       // A - масив, який  потрібно
{    var t;                        // відсортувати по зростанню
   var n = A.length;
        alert(n);
   for (var i = 0; i < n-1; i++)
    { for (var j = 0; j < n-1-i; j++)
      { if (A[j+1] < A[j])
        { var t = A[j+1]; A[j+1] = A[j]; A[j] = t; }
      }
                while (i<A.length){
document.write(A[i]+"<br />");
i++};
    }
```

```
   return A;    // На виході відсортований по зростанню масив A.
}
var n=[47,157,70,45,36];
var i=0;

while (i<n.length){
document.write(n[i]+"<br />");
i++};

document.write("_____");
document.write("<br>");
BubbleSort(n);
while (i<n.length){
document.write(n[i]+"<br />");
i++};
</script>
</body>
</html>
```

Приклад 5.19. Знаходження максимального елементу масиву.

```
<script type="text/jscript">
var a = [2,3,7,5,3,7,1,3,4];
for(var i=0;i<a.length;i++){
document.write("<br>"+a[i]);
}
var max=a[0];
for(var i=0;i<a.length;i++){
if(a[i]>max) max=a[i];
}
document.write("<br>"+"max="+max);
```

```
</script>
```

Приклад 5.20 Мінливі картинки.

```
<SCRIPT>
var i=0;
function movie()
{
 document.i.src='slash'+i+'.jpg';
 i++; if(i>4) i=0;
 setTimeout('movie();',500);
}
</SCRIPT>
<BODY onLoad="movie();">
<IMG NAME=i>
</BODY>
```

Приклад 5.21Змінюємо надпис на кнопці.

```
<SCRIPT LANGUAGE="JavaScript">
<!--
var a = 1;
var b = 1;
stringm = new Array(4);
      stringm[1] = "Ця кнопка";
      stringm[2] = "змінила";
      stringm[3] = "свій надпис";
      stringm[4] = "Вам подобається?!";
function my() {
if (b==1) {
 document.form1.abutton.value = stringm[a]
   b-- ;
    a++;
 if (a>4) {a = 1;
```

```
  document.form1.back.value = "Повторити";
 };}
else {
 document.form1.abutton.value = "ok";
document.form1.back.value = "Назад";
  b++ }
}
</script></p>
<form name="form1">
<input type="button" value="НАЗАД"  name="back">
<input type="button" name="abutton" value="ok" onclick="my()">
</form>
```

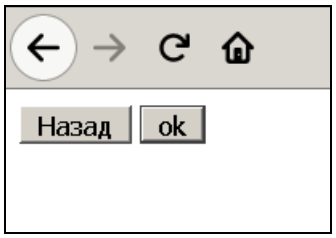


Рис.5.2. Результати виконання прикладу 5.21.

5.11.Події миші

OnMouseOver. Переміщення покажчика миші на елемент.

OnMouseOut. Переміщення покажчика миші за межі елемента.

OnMouseDown. Натискання будь-якої кнопки миші.

OnMouseUp. Відпускання будь-якої кнопки миші.

OnMouseMove. Переміщення покажчика миші.

OnClick. Клацання лівою кнопкою миші на елементі або натискання <Enter>.

OnDblClick. Подвійне клацання лівою кнопкою миші на елементі.

OnDragStart. Запуск операції перетягування.

OnSelectStart. Запуск операції виділення елемента.

OnSelect. Виділення елемента.

5.12.Події клавіатури

OnKeyPress. Натиснення і відпускання клавіші. Подія виникає багато разів, якщо клавіша утримується.

OnKeyDown. Натискання клавіші.

OnKeyUp. Відпускання клавіші.

5.13.Подія прокручування

OnScroll. Використання смуги прокрутки або прокручування елемента неявно за допомогою іншого способи

5.14.Події фокуса

OnFocus. Виникає, коли елемент активізується після клацання по ньому мишею або за допомогою клавіатури. Фокус можуть отримати тільки елементи призначеного для користувача введення і тіло документа, а не елементи змісту документа.

OnBlur. Виникає, коли елемент втрачає фокус. Використовується для контролю коректності введення.

Події mousedown і mouseup в основному використовуються, коли натискаємо на кнопку, а потім мишу відпускаємо. Наприклад, при виділенні тексту, або перенесенні об'єкта.

Події click і dblclick в основному потрібні для роботи з простими кліками. Клік відбувається при послідовних mousedown-> mouseup на одному і тому ж об'єкті.

Подія contextmenu виникає при правому кліку мишею, і за замовчуванням викликає контекстне меню. Однак не на всіх браузерах показ меню можна відключити.

Події mouseover і mouseout спрацьовують кожен раз, коли миша заходить на елемент або виходить з нього.

Подія mousemove спрацьовує при кожному пересуванні миші.

Приклад 5.22. Зміна виду елемента.

```
<html><head><title>Зміна виду елемента</title>
</head>
<body>
<H2>Зміна виду даного елемента</H2>
<p onmousedown="this.style.fontStyle='italic';
   this.style.color='red'" onmouseup="this.style.fontStyle='';
   this.style.color='blue'">
```

Для зміни кольору і стилю шрифту даного тексту натискайте і відпускайте кнопку миші</p>

</body></html>

Зверніть увагу, що в прикладі 5.22. область дії обробника події визначена за допомогою покажчика this.

5.15.Методи об'єкта document

•clear. Видалення вмісту документа з вікна перегляду.

•write. Запис в документ довільної HTML-конструкції.

•writeln. Аналогічний write, але з додаванням символу переведення рядка в кінець.

•open. Відкриття вихідного потоку для запису в HTML-документ даних типу MIME за допомогою методів write і writeln.

•close. Закриття потоку даних, відкритого методом open.

•getElementById(a) Отримати об'єкт по його ID.

•getElementsByTagName(tag) Отримати масив об'єктів по імені тега.

5.16.Посилання в документі

Для кожного посилання, розміщеного в HTML–документі, створюється окремий об'єкт. Всі такі об'єкти знаходяться в об'єкті document, як елементи масиву links. Аналізуючи ці елементи, сценарій JavaScript можна визначити властивості кожного посилання в HTML– документі:

• length. Кількість посилань в HTML-документі, тобто кількість елементів в масиві links.

• hash. Ім'я локального посилання, якщо воно визначена в URL.

• host. Ім'я вузла і порта, зазначені в URL.

• hostname. Ім'я вузла і доменне ім'я вузла мережі

• href. Повний URL.

• pathname. Шлях до об'єкта, зазначений в URL.

• port. Номер порта, який використовується для передачі даних з сервером, зазначеного у посиланні.

• protocol. Рядок назви протоколу передачі даних.

• search. Рядок запиту, зазначена в URL після символу "?".

• target. Ім'я вікна, куди буде завантажений документ при виконанні посилання. Це може бути ім'я існуючого вікна фрейму, визначеного тегом <FRAMESET>, або одне із зарезервованих імен - _top, _parent, _self, _blank.

Приклад 5.23. Посилання в документі.

```
<html><head>
<meta charset=utf-8">
</head>
<body>
<P><A
HREF="http://mses.kpi.ua
/1/tsesliv2.html"
 TARGET="_top">Моя
сторінка</A>
<P><A
HREF="https://lpnu.ua/">
Тут вчився</A>
<P><A
HREF="https://kpi.ua/">T
ут працюю</A>
<HR>
<H1>Властивості
посилань</H1>

<script type="text/javascript">
   <!--
   // Цикл по всіх посиланнях
   for(i=0; i<document.links.length; i++)
   {
// розділова лінія
```

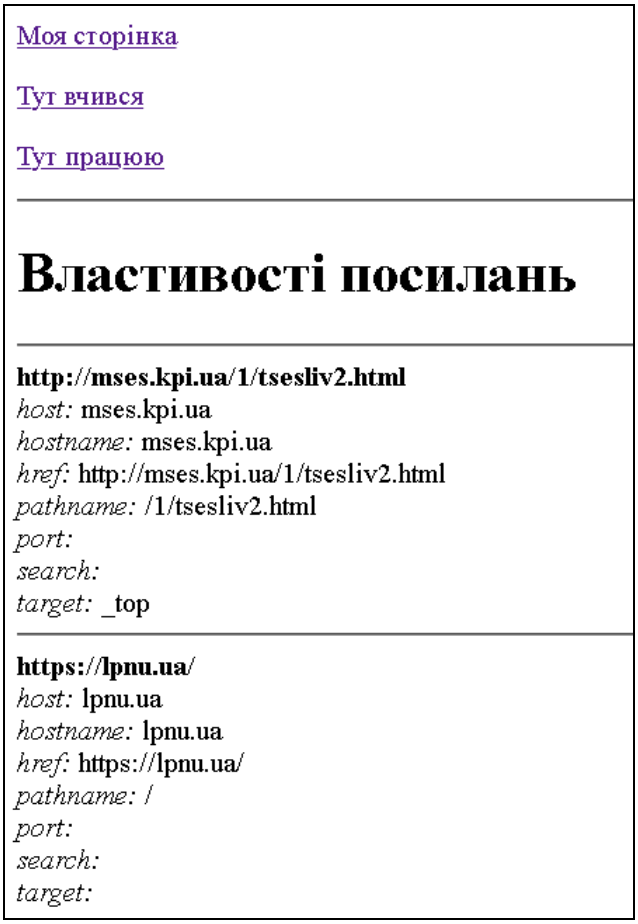


Рис.5.3. Результати виконання прикладу 5.23.

```
        document.write("<HR>");
// Текст посилання, виділений жирним шрифтом
        var Txt=document.links[i] + "<br>";
        document.write(Txt.bold());
// Всі властивості посилання
        document.write("host: ".italics() + document.links[i].host +
           "<br>");
        document.write("hostname: ".italics() +
           document.links[i].hostname + "<br>");
        document.write("href: ".italics() + document.links[i].href +
           "<br>");
        document.write("pathname: ".italics() +
           document.links[i].pathname + "<br>");
        document.write("port: ".italics() + document.links[i].port +
           "<br>");
        document.write("search: ".italics() + document.links[i].search
           + "<br>");
        document.write("target: ".italics() + document.links[i].target
           + "<br>");
    }
    //-->
</script></body></html>
```

Завдання для самостійної роботи

Варіант №1	1.	Ввести та вивести двовимірний масив.
	2.	Знайти максимальний елемент двовимірного масиву.
	3.	Знайти суму елементів двовимірного масиву.
	4.	Написати програму мінливі картинки.
Варіант №2	1.	Ввести та вивести одновимірний масив.

	2. Знайти суму елементів одновимірного масиву. 3. Знайти максимальний елемент одновимірного масиву. 4. Написати програму, яка змінює надпис на кнопці, використовуючи подію dblclick.
Варіант №3	1. Ввести та вивести двовимірний масив. 2. Знайти суму діагональних елементів двовимірного масиву 3. Знайти мінімальний елемент двовимірного масиву. 4. Написати програму мінливі картинки.
Варіант №4	1. Ввести двовимірний масив. 2. Знайти суму елементів другого стовпця двовимірного масиву 3. Знайти мінімальний елемент двовимірного масиву. 4. Написати програму, яка змінює надпис на кнопці, використовуючи подію OnMouseOver.
Варіант №5	1. Ввести двовимірний масив. 2. Знайти суму елементів другого рядка двовимірного масиву 3. Відсортувати одновимірний масив. 4. Написати програму, яка змінює надпис на кнопці, використовуючи подію OnFocus.

РОЗДІЛ 6 ОБ’ЄКТИ

Об'єкти JavaScript схожі на масиви, але для доступу до елемента об'єкта використовують рядки, а не числа. Ці рядки називають ключами, а елементи, які їм відповідають – значеннями. Утворюються пари «ключ-значення ».

6.1.Створення об'єктів

Для зберігання різноманітної інформації про студента підійде JavaScript-об'єкт.

```
var student = {
"age": 18,
"name": "Іван",
"lastname": "Петренко"
};
```

Створюючи перший об'єкт, ми писали імена ключів в лапках, проте це не обов'язково.

6.2.Доступ до значень всередині об'єктів

Значення, які зберігаються в об'єктах, можна отримати як елементи масиву. Єдина відмінність в тому, що замість індексу (число) використовується ключ (рядок).

```
var cat = {
legs: 3,
name: "МУМУ",
color: "Сірий"
};
document.write(cat['name']);
```

Якщо треба дізнатися, які ключі у даного об'єкта. Для цього в JavaScript є зручний засіб – команда Object.keys ():

```
var cat = {
legs: 3,
name: "МУМУ",
color: "Сірий"
```

```
};
document.write(Object.keys(cat));
legs,name,color
```

6.3.Додавання елементів об'єкта

Порожній об'єкт схожий на порожній масив, тільки замість квадратних душок при створенні використовуються фігурні:

```
var object = {};
```

Додавати елементи об'єкта можна так:

```
cat["legs"] = 3;
cat["name"] = "Мур";
cat["color"] = "Сірий";
document.write(cat["color"]);
```

6.4.Масив об'єктів

Розглянемо більш складний приклад – масив об'єктів з відомостями про друзів, де в кожен з об'єктів вкладено по ще одному масиву.

```
var anna = { name: "Анна", age: 17, luckyNumbers: [2, 4, 8, 16] };
var danja = { name: "Даня", age: 18, luckyNumbers: [3, 9, 40] };
var kate = { name: "Катя", age: 19, luckyNumbers: [1, 2, 3] };
```

Ми створили три об'єкти та зберігли їх у змінних anna, danja, kate. У кожного з цих об'єктів є по три властивості: name, age, luckyNumbers. Кожному ключу name відповідає строкове значення, ключу age – числове, а ключу luckyNumbers – масив, в якому кілька чисел. Тепер створимо масив друзів:

Приклад 6.1 Робота з масивами.

```
var anna = {name: "Анна", age: 17, luckyNumbers: [2, 4, 8, 16]};
var danja = {name: "Даня", age: 18, luckyNumbers: [3, 9, 40]};
var kate = {name: "Катя", age: 19, luckyNumbers: [1, 2, 3]};
var friends = [anna, danja, kate];
for(i=0;i<friends.length; i++){
```

```
document.write(friends[i].name+"",friends[i].age+"", friends[i].luckyNumbers
+"<br>");
```

Приклад 6.2. Масив інформації о фільмах на комп'ютері.

```
<script>
var movies = {
"В пошуках Немо": {
releaseDate: 2003,
duration: 100,
actors: ["Альберт Брукс "," Еллен Дедженерес "," Олександр Гоулд
Al′bert Bruks "," Ellen Dedzheneres "," Oleks "],
format: "DVD"
},
" Зоряні війни: Епізод VI - Повернення джедая ": {
releaseDate: 1983,
duration: 134,
actors: ["Марк Хемілл "," Харрісон Форд "," Керрі Фішер "],
format: "DVD"
},
" Гаррі Поттер і кубок вогню ": {
releaseDate: 2005,
duration: 157,
actors: ["Дэниел Рэдклифф", "Эмма Уотсон", "Руперт Гринт"],
format: "Blu-ray"
}
};
var findingNemo = movies["Звездные войны: Эпизод VI — Возвращение
джедая"];
document.write(findingNemo.releaseDate+"<br>",findingNemo.duration+"<
br>",findingNemo.actors+"<br>",findingNemo.format);
</script>
```

Крім того, у колекцію легко додавати нові фільми:

```
var cars = {
releaseDate: 2006,
duration: 117,
actors: ["Оуен Уїлсон "," Бонні Хант "," Пол Ньюман "],
format: "Blu-ray"
};
var findingNemo = movies["cars"];
document.write(cars.releaseDate+"<br>",cars.duration+"<br>",cars.actors+"
<br>",cars.format);
```

Приклад 6.3. Доступ до елементів документа.

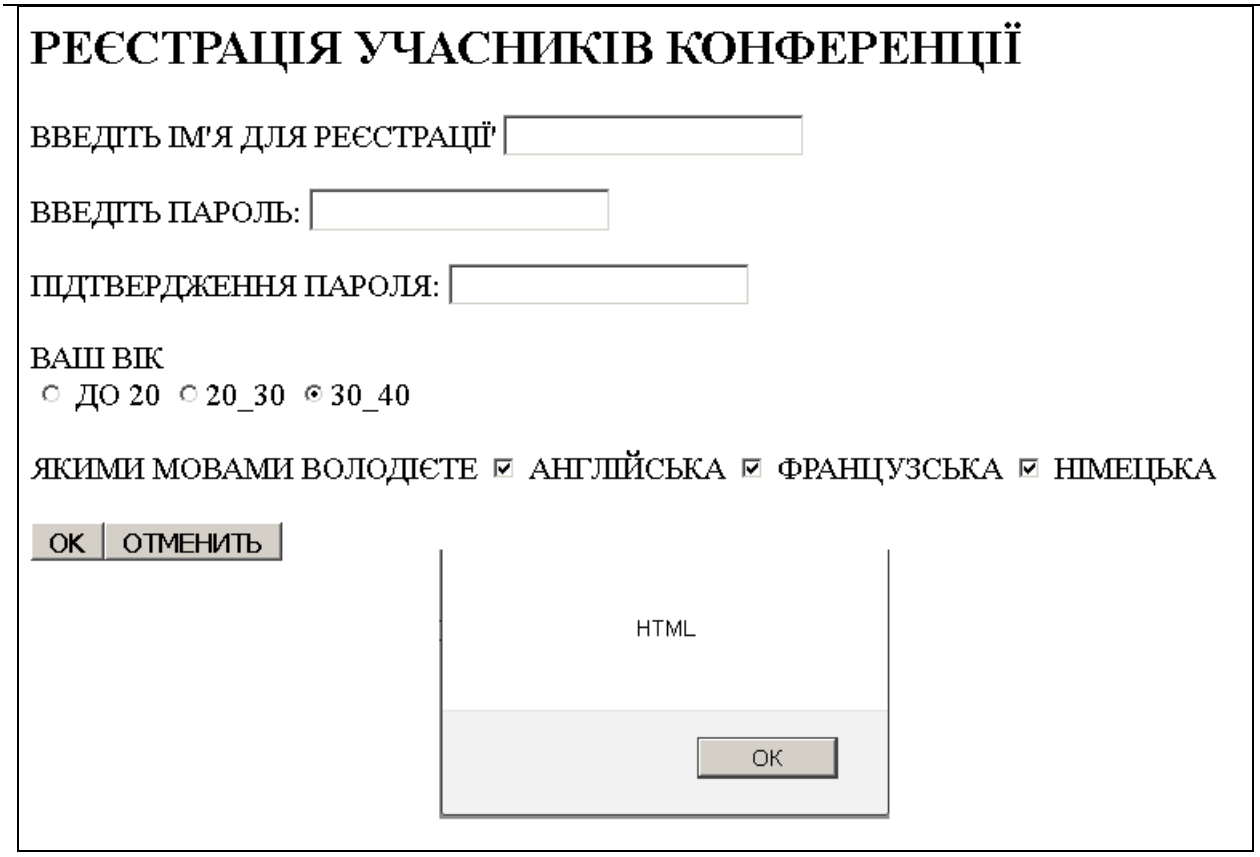


Рис.6.1 **Результат виконання прикладу 6.3.**

```
<HTML><HEAD><TITLE>ФОРМИ</TITLE>
<meta charset=UTF-8"> </HEAD>
```

```
<BODY>
<H2>РЕЄСТРАЦІЯ УЧАСНИКІВ КОНФЕРЕНЦІЇ</H2>
<script>
function ola(){
for(var i=0;i<document.all.length; i++)
alert(document.all.item(i).tagName);
}
</script>
<FORM  name="form">
ВВЕДІТЬ ІМ'Я ДЛЯ РЕЄСТРАЦІЇ' <INPUT TYPE="TEXT" NAME="REGNAME"><p>
ВВЕДІТЬ ПАРОЛЬ: <INPUT TYPE="PASSWORD" NAME="PASSWORD1" MAXLENGTH=8><p>
ПІДТВЕРДЖЕННЯ ПАРОЛЯ: <INPUT TYPE="PASSWORD" NAME="PASSWORD2" MAXLENGTH=8><p>
ВАШ ВІК<br>
<INPUT TYPE="RADIO" NAME="AGE" VALUE="LT20" CHECKED id="one"> ДО 20
<INPUT TYPE="RADIO" NAME="AGE" VALUE="20_30" CHECKED id="one">20_30
<INPUT TYPE="RADIO" NAME="AGE" VALUE="30_40" CHECKED id="one">30_40
<BR><BR>
ЯКИМИ МОВАМИ ВОЛОДІЄТЕ
<INPUT TYPE="CHECKBOX" NAME="LANGUAGE" VALUE="ENGLISH" id="two" CHECKED > АНГЛІЙСЬКА
<INPUT TYPE="CHECKBOX" NAME="LANGUAGE" VALUE="FRENCH" id="two" CHECKED > ФРАНЦУЗСЬКА
<INPUT TYPE="CHECKBOX" NAME="LANGUAGE" VALUE="GERMAN" id="two" CHECKED > НІМЕЦЬКА
```

```
<BR><BR>
<INPUT TYPE="button" VALUE="OK" onClick="ola()"><INPUT TYPE="RESET" VALUE="ОТМЕНИТЬ">
</FORM></BODY></HTML>
```

Завдання для самостійної роботи

Варіант№1	1. Створити об’єкт Викладачі. 2. Вивести прізвища викладачів. 3. Додати нового викладача. 4. Масив об'єктів Кафедра.
Варіант№2	1. Створити об’єкт Студенти. 2. Вивести прізвища викладачів. 3. Додати нового викладача. 4. Масив об'єктів Група.
Варіант№3	1. Створити об’єкт Кафедра. 2. Вивести прізвища викладачів. 3. Додати нового викладача. 4. Масив об'єктів Факультет.
Варіант№4	1. Створити об’єкт Лікар. 2. Вивести прізвища викладачів. 3. Додати нового викладача. 4. Масив об'єктів Поліклініка.

РОЗДІЛ 7 ОБ'ЄКТНА МОДЕЛЬ ДОКУМЕНТА

DOM(Document Object Model) – це засіб, що дозволяє JavaScript-коду взаємодіяти з вмістом веб-сторінок. Браузери використовують DOM для структурування сторінок та їх елементів (параграфів, заголовків і т. д.). JavaScript може різними способами маніпулювати елементами DOM.

В даному розділі познайомимось з інструмент jQuery, який кардинально спрощує роботу з DOM. jQuery містить набір функцій, які дозволяють знайти потрібні вам елементи і проводити з ними певні дії.

Коли ви відкриваєте HTML-документ, браузер перетворює його елементи в деревоподібну структуру – дерево DOM. На рис. 6.1 зображені просте дерево DOM. Браузер дає JavaScript-програмістам можливість доступу до цієї деревовидної структури за допомогою спеціальних методів DOM.

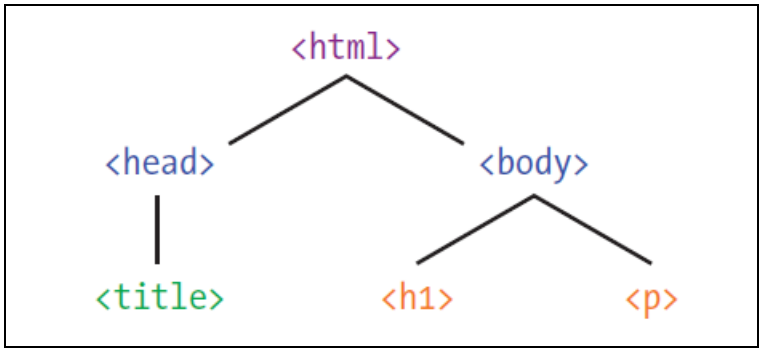


Рис 7.1. Дерево DOM

7.1. Ідентифікація елементів по id

HTML-елементу можна привласнити унікальне ім'я, або ідентифікатор, за допомогою атрибута id. Наприклад, у елемента h1 заданий атрибут id:

```
<h1 id="main1">Привет, мир!</h1>
```

Задавши атрибуту id значення (в даному випадку "main1"), отримуємо можливість знайти цей конкретний заголовок по його id, щось з ним зробити, не зачіпаючи інші елементи, навіть заголовки рівня h1.

7.2.Пошук елемента за допомогою getElementById

Позначивши елемент унікальним id можемо скористуватися DOM-методом document.getElementById, щоб знайти елемент "main1":

Приклад 7.1. Пошук елемента за допомогою getElementById.

```
<body>
<h1 id="main1"> Привіт, світ! h1>
<script>
var headingElement = document.getElementById("main1");
alert(headingElement.innerHTML);
</script>
</body>
```

Приклад 7.2. Виводимо теги HTML сторінки.

```
<html><head><title>AllElements</title>
<script>
function changeAllElements(){
var i,origLength;
origLength=document.all.length;
document.writeln('document.all.length='+origLength+"<br />");
for(i=0;i<origLength;i++)
{
document.write("document.all["+i+"]="+document.all[i].tagName+"<br/>");
document.all[i];
}
}
</script>
</head>
<body>
<form>
<img src="3.gif">
<input type="button" onClick="changeAllElements()" value="Go"/>
</form></body></html>
```

7.3.Міняємо текст заголовка через DOM

Приклад 7.3. Міняємо текст заголовка.

```
<html><head>
<title>Изучаем DOM</title>
</head>
<body>
<h1 id="main1"> Привіт, світ! <h1>
<script>
var headingElement =
document.getElementById("main1");
document.write(headingElement.innerHTML);
 var newHeadingText = prompt("Введіть новий заголовок:");
 document.write(headingElement.innerHTML = newHeadingText);
</script>
</body></html>
```

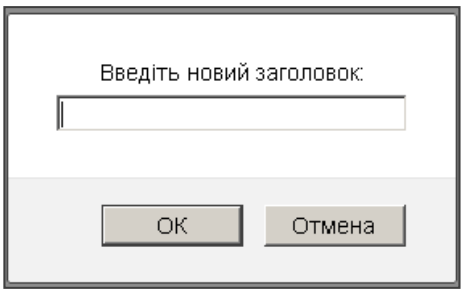


Рис.7.2. Результат виконання прикладу 7.2

За допомогою document.getElementById знаходимо елемент h1 (id якого «main1») і зберігаємо його в змінної headingElement.

Вивели рядок з заголовком document.write(headingElement.innerHTML);

У рядку var newHeadingText = prompt("Введіть новий заголовок:"); відкриваємо діалог prompt, щоб отримати від користувача новий заголовок, та зберігаємо введене значення.

Приклад 7.4. Виведення даних з елементу Textarea .

```
<html>
<body>
<h3>Демонстрація зчитування даних з елементу TEXTAREA </h3>
Адреса:<br>
```

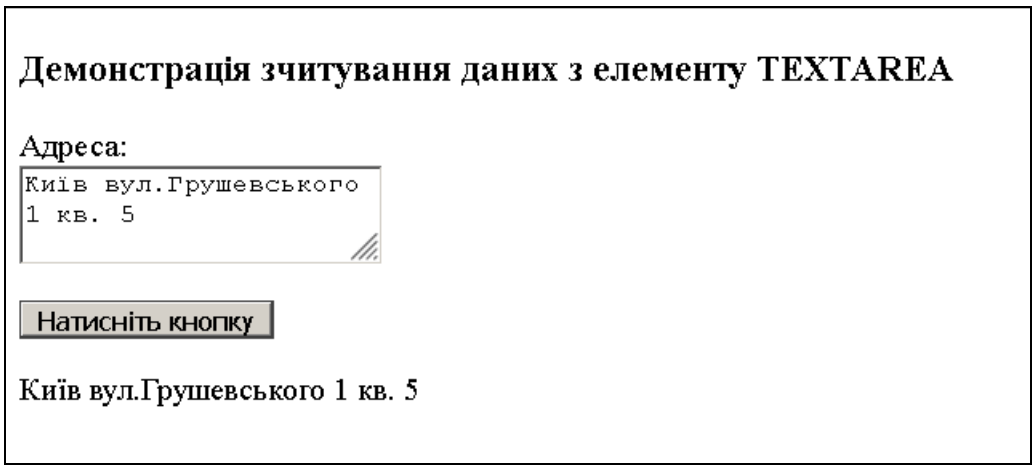


Рис.7.3. **Результат виконання прикладу 7.4**

```
<textarea id="myTextarea">
Київ вул.Грушевського 1 кв. 5
</textarea>
<p></p>
<button type="button" onclick="myFunction()">Натисніть кнопку</button>
<p id="demo"></p>
<script>
function myFunction() {
  var x = document.getElementById("myTextarea").value;
  document.getElementById("demo").innerHTML = x;
}
</script>
</body>
</html>
```

Приклад 7.5. Виведення даних з елементу checkbox .

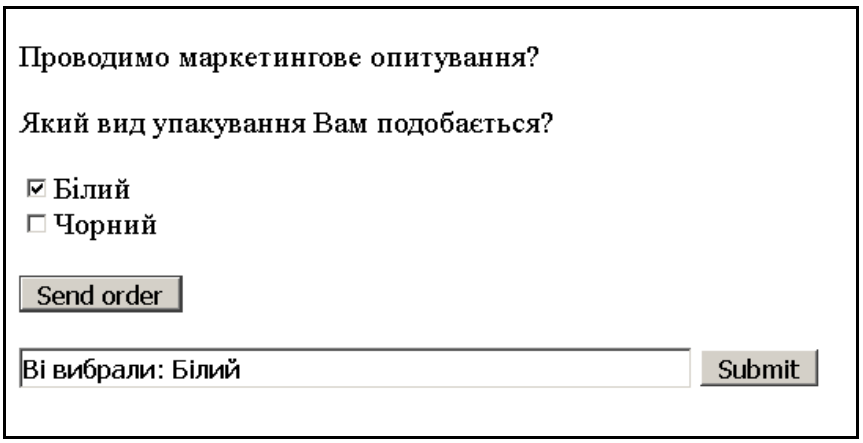


Рис.7.4. **Результат виконання прикладу 7.5**

```
<html>
<body>
<p>Проводимо маркетингове опитування?</p>
<p>Який вид упакування Вам подобається?</p>
<form>
<input type="checkbox" name="coffee" value="Білий">Білий<br>
<input type="checkbox" name="coffee" value="Чорний">Чорний<br>
<br>
<input type="button" onClick="myFunction()" value="Send order">
<br><br>
<input type="text" id="order" size="50">
<input type="submit" value="Submit">
</form>
<script>
function myFunction() {
  var coffee = document.forms[0];
  var txt = "";
  var i;
  for (i = 0; i < coffee.length; i++) {
    if (coffee[i].checked) {
      txt = txt + coffee[i].value + " ";
    }}
  document.getElementById("order").value = "Ві вибрали: " + txt;
```

```
}
</script>
</body>
</html>
```

Приклад 7.6. Виведення даних форми.

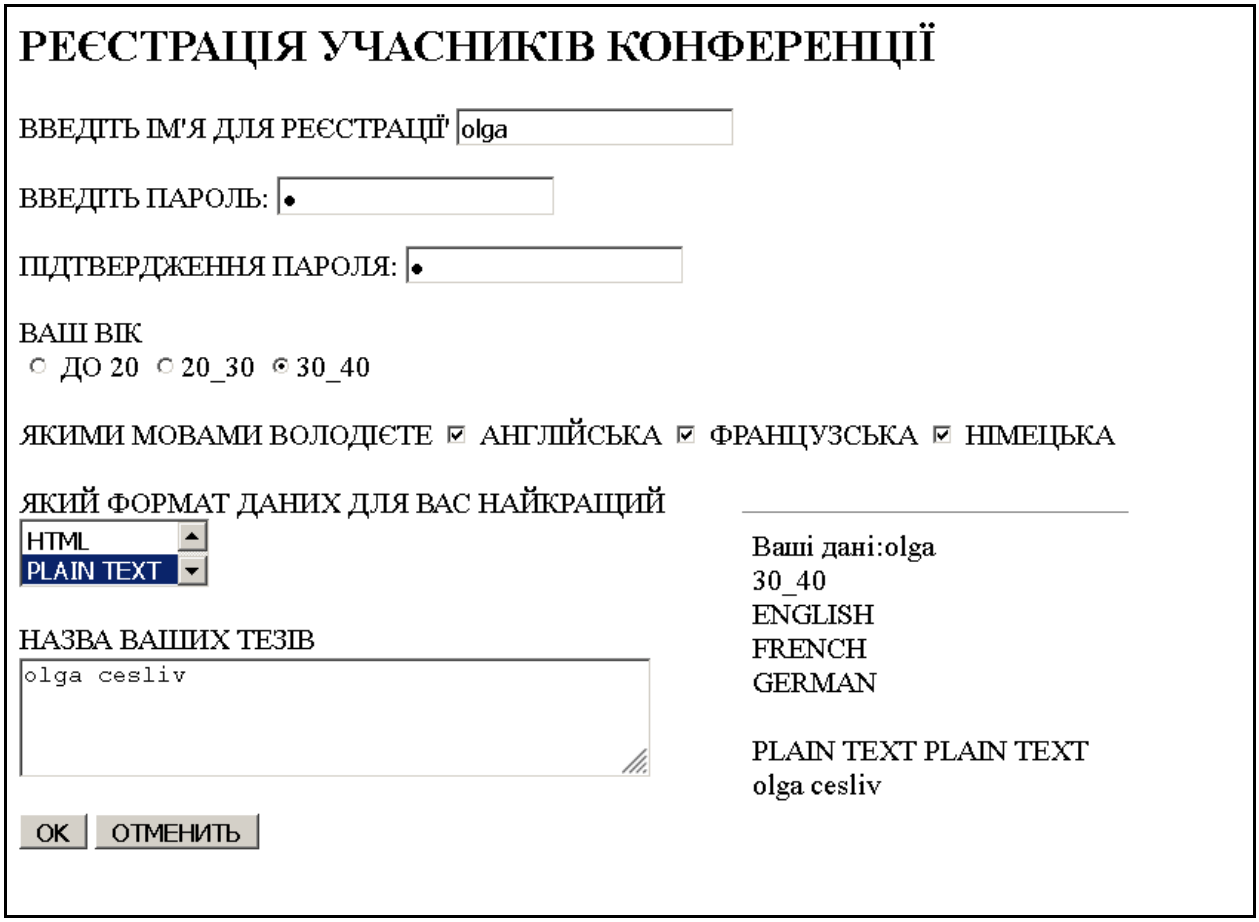


Рис 7.5 **Результати прикладу 7.6**

```
<HTML><HEAD><TITLE>ФОРМИ</TITLE>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8 ">
</HEAD>
<BODY>
<H2>РЕЄСТРАЦІЯ УЧАСНИКІВ КОНФЕРЕНЦІЇ</H2>
<script>
function myFunction(text1) {
```

```
var LANGUAGE = document.forms[0];
var txt = "";
var i;
for (i = 0; i < LANGUAGE.length; i++) {
   if (LANGUAGE[i].checked) {
     txt = txt + LANGUAGE[i].value + "<br> ";
   }
 }
var r;
var option = document.getElementsByTagName('option');
for(var i = 0; i < option.length; i++){
if(option[i].selected){
r=(option[i].value + " " + option[i].text);
}}
 var x = document.getElementById("three").value;
  document.write("Ваші дані:"+text1+"<br>"+txt+"<br>"+r+"<br>"+x) ;
}
</script>
<FORM  name="form">
ВВЕДІТЬ ІМ'Я ДЛЯ РЕЄСТРАЦІЇ'  <INPUT TYPE="TEXT"
NAME="REGNAME"><p>
ВВЕДІТЬ ПАРОЛЬ: <INPUT TYPE="PASSWORD" NAME="PASSWORD1"
MAXLENGTH=8><p>
ПІДТВЕРДЖЕННЯ ПАРОЛЯ:  <INPUT TYPE="PASSWORD"
NAME="PASSWORD2" MAXLENGTH=8><p>
ВАШ ВІК<br>
<INPUT TYPE="RADIO" NAME="AGE" VALUE="LT20" CHECKED
id="one"> ДО 20
<INPUT TYPE="RADIO" NAME="AGE" VALUE="20_30" CHECKED
id="one">20_30
```

<INPUT TYPE="RADIO" NAME="AGE" VALUE="30_40" CHECKED
id="one">30_40

ЯКИМИ МОВАМИ ВОЛОДІЄТЕ
<INPUT TYPE="CHECKBOX" NAME="LANGUAGE" VALUE="ENGLISH"
id="two" CHECKED > АНГЛІЙСЬКА
<INPUT TYPE="CHECKBOX" NAME="LANGUAGE" VALUE="FRENCH"
id="two" CHECKED > ФРАНЦУЗСЬКА
<INPUT TYPE="CHECKBOX" NAME="LANGUAGE" VALUE="GERMAN"
id="two" CHECKED > НІМЕЦЬКА

ЯКИЙ ФОРМАТ ДАНИХ ДЛЯ ВАС НАЙКРАЩИЙ

<SELECT NAME="FORMAT" SIZE=2>
<OPTION SELECTED VALUE="HTML">HTML
<OPTION VALUE="PLAIN TEXT">PLAIN TEXT
<OPTION VALUE="POSTSCRIPT">POSTSCRIPT
<OPTION VALUE="PDF">PDF
</SELECT>

НАЗВА ВАШИХ ТЕЗІВ

<TEXTAREA NAME="WISH" COLS=40 ROWS=3 id="three">
</TEXTAREA>

<INPUT TYPE="button" VALUE="OK"
onClick="myFunction(form.REGNAME.value)" >
<INPUT TYPE="RESET" VALUE="ОТМЕНИТЬ">
</FORM>
</BODY></HTML>

7.4.Робота з деревом DOM через jQuery

Вбудовані в браузер методи DOM всім хороші, але користуватися ними буває нелегко, тому багато програмістів застосовують спеціальну бібліотеку під назвою jQuery. jQuery – це набір інструментів (в основному функцій), які сильно спрощують роботу з DOM-елементами. Підключивши цю бібліотеку нашу сторінку, ми зможемо викликати її функції і методи в доповнення до функцій і методів, вбудованим в JavaScript і в браузер.

7.5.Підключаємо jQuery до HTML-сторінці

Перш ніж скористатися бібліотекою jQuery, потрібно, щоб браузер її завантажив, для чого достатньо одного рядка HTML-коду:

```
<script src = "https://code.jquery.com/jquery-2.1.0.js"> </ script>
```

Приклад 7.7. Вивчаємо DOM.

```
<html>
<head>
<title>Вивчення DOM</title>
</head>
<body>
<h1 id="main1">Привіт, світ!</h1>
 <script src="https://code.jquery.com/jquery-2.1.0.js"></script>
 <script>
var newHeadingText = prompt("Введіть новий заголовок:");
 $("#main1").text(newHeadingText);
</script>
</body></html>
```

Функція $ приймає один аргумент, який називається *рядок селектора*. Цей рядок вказує, який елемент або елементи потрібно знайти в дереві DOM. У нашому випадку це "#main1". Символ # означає id, тобто селектор "# main1" посилається на елемент, id якого дорівнює main1.

Функція $ повертає об'єкт jQuery, відповідний знайденим елементам. Наприклад, виклик $ ("# main1") поверне jQuery-об'єкт для елемента h1 (id якого дорівнює "main1").

Приклад 7.8. Отримати елемент за вказаним id.

```
<html><head><title>AllElements</title>
<script>
function changeAllElements(){
var obElement = document.all.item("test");
obElement.style.color='red';
}
</script>
</head>
<body >
<p id="test">JavaScript</p>
<form>
<input type="button"
onClick="changeAllElements()" value="Go"/>
</form></body></html>
```

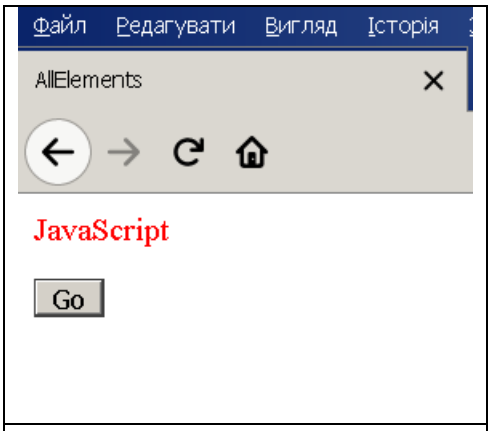


Рис 7.6 Результати прикладу 7.8

7.6.Створення нових елементів через jQuery

Крім зміни існуючих елементів, за допомогою jQuery можна створювати нові елементи і додавати їх в дерево DOM.

```
$("body").append("<p>Це новий параграф</p>");
```

Перша частина цієї команди викликає функцію $ з рядком селектора "body", щоб знайти тіло нашого HTML-документа. Пошук не обов'язково повинен відбуватися по id – коду, $ ("body ") шукає елемент body, так само можемо викликати $ ("p") для пошуку всіх елементів p.

Далі ми викликаємо для знайденого об'єкта метод append - переданий йому рядок перетвориться в DOM-елемент, а потім додається в body.

Також append можна використовувати в циклі for для додавання декількох елементів:

Приклад 7.9. Додавання елементів.

```
<script src="https://code.jquery.com/jquery-2.1.0.js"></script>
 <script>
for (var i = 0; i < 3; i++) {
var hobby = prompt("Назви одне з своїх хобі!");
$("body").append("<p>" + hobby + "</p>");
}
</script>
```

7.7.Анімація елементів засобами jQuery

На багатьох сайтах при показі і приховуванні частин сторінки використовується анімація.

За допомогою jQuery анімувати елементи зовсім не складно. Наприклад щоб елемент повільно зникав, скористаємося методом fadeOut. Замініть вміст другого елементу script в dom.html на такий код:

```
<body>
<h1 id="main1">Привіт, світ!</h1>
 <script src="https://code.jquery.com/jquery-2.1.0.js"></script>
 <script>
$("h1").fadeOut(3000);
</script>
</body>
```

Щоб знайти всі елементи h1, ми використовували функцію $. Оскільки в нашому файлі елемент h1 всього один (це заголовок з текстом «Привіт, світ!»), саме його ми і отримаємо у вигляді jQuery-об'єкта. Викликаючи для цього об'єкта метод fadeout (3000), ми запускаємо загасання заголовка до повного його зникнення протягом трьох секунд.

Анімація fadeIn змушує невидимий елемент проявитися. В результаті елемент h1 буде затухати протягом трьох секунд, а потім протягом двох секунд проявлятися.

Приклад 7.10. Приклад затухання елементу.

```
<script>
$("h1").fadeOut(3000);
$("h1").fadeIn(3000);
</script>
```

В jQuery є два методи для анімації– це slideUp і slideDown. При виклику slideUp елементи зникають, відпливаючи вгору, а при виклику slideDown з'являються, опускаючись згори.

```
$("h1").slideUp(1000).slideDown(1000);
```

Завдання для самостійної роботи

Варіант №1	1. Створити програму, яка знаходить елемент за допомогою метода getElementById(). 2. Створіть масив з іменами кількох друзів. У циклі for створіть для кожного імені по одному елементу p і додайте ці елементи в кінець <body>, викликаючи jQuery-метод append. За допомогою jQuery змініть текст елемента h1, щоб замість "Привіт світ!" там було "Мої друзі". використовуйте метод fadeOut і метод fadeIn, щоб імена плавно виникали на екрані. Тепер змініть створені вами елементи p, додавши після кожного імені слово "кращий!". Підказка: якщо знайти відразу всі елементи p за допомогою $("p"), метод append можна викликати для них усіх разом.
Варіант №2	1. Створити програму, яка виводить теги HTML сторінки. 2. Створіть масив з іменами кількох студентів. У циклі for створіть для кожного імені по одному елементу p і

	додайте ці елементи в кінець <body>, викликаючи jQuery-метод append. За допомогою jQuery змініть текст елемента h1, щоб замість "Привіт світ!" там було "Студенти". використовуйте метод fadeOut і метод fadeIn, щоб імена плавно виникали на екрані. Тепер змініть створені вами елементи p, додавши після кожного імені слово "кращий!". Підказка: якщо знайти відразу всі елементи p за допомогою $("p"), метод append можна викликати для них усіх разом.
Варіант №3	1. Створити програму, яка виводить дані з елементу checkbox. 2. Створити програму - Миготливий заголовок. Як за допомогою fadeOut і fadeIn зробити так, щоб заголовок миготів п'ять разів з інтервалом в секунду?
Варіант №4	1. Створити програму, яка додає елементи на веб-сторінку. 2. Створити програму - Миготливий заголовок. За допомогою циклу for, зробити так, щоб заголовок миготів п'ять разів з інтервалом в секунду?
Варіант №5	1. Створити програму, яка виводить дані з форми. 2. Створити програму - Миготливий заголовок. За допомогою циклу зробити, щоб заголовок з'являвся і зникав в перший раз за секунду, потім за дві, потім за три.

РОЗДІЛ 8. ПУБЛІКАЦІЯ САЙТУ НА СЕРВЕРІ

8.1. Хостинг

У попередніх розділах розглядалися мови HTML та JavaScript, їх можливості по створенню Web-сторінок. Тепер вам потрібно з окремих сторінок сконструювати свій Web-сайт і розмістити його в Інтернеті.

Приступаючи до розробки свого сайту, потрібно чітко визначити його призначення. Звичайно сайти створюються для того, щоб заявити про себе чи про свою організацію, повідомити про результати роботи чи про свої досягнення, налагодити ділові зв'язки, дати рекламу про товари чи послуги тощо.

Крім призначення сайту, потрібно визначити коло його потенційних відвідувачів, тобто *аудиторію.*

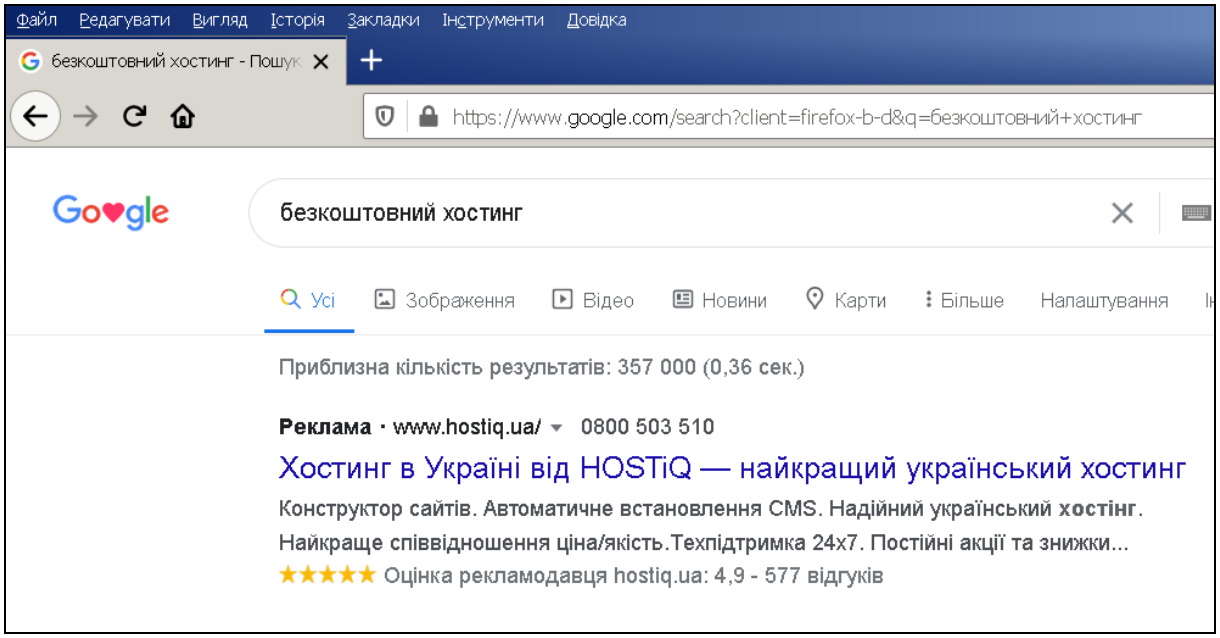


Рис.8.1. **Безкоштовний Хостинг**

Наступним етапом буде *підбір матеріалу.* Не весь матеріал по тематиці сайту, що ви маєте, варто публікувати в Інтернеті. Потрібно оцінити якість матеріалу та його цікавість для відвідувачів. При плануванні сайту,

призначеного для якої-небудь організації, важливим є питання фінансування робіт.

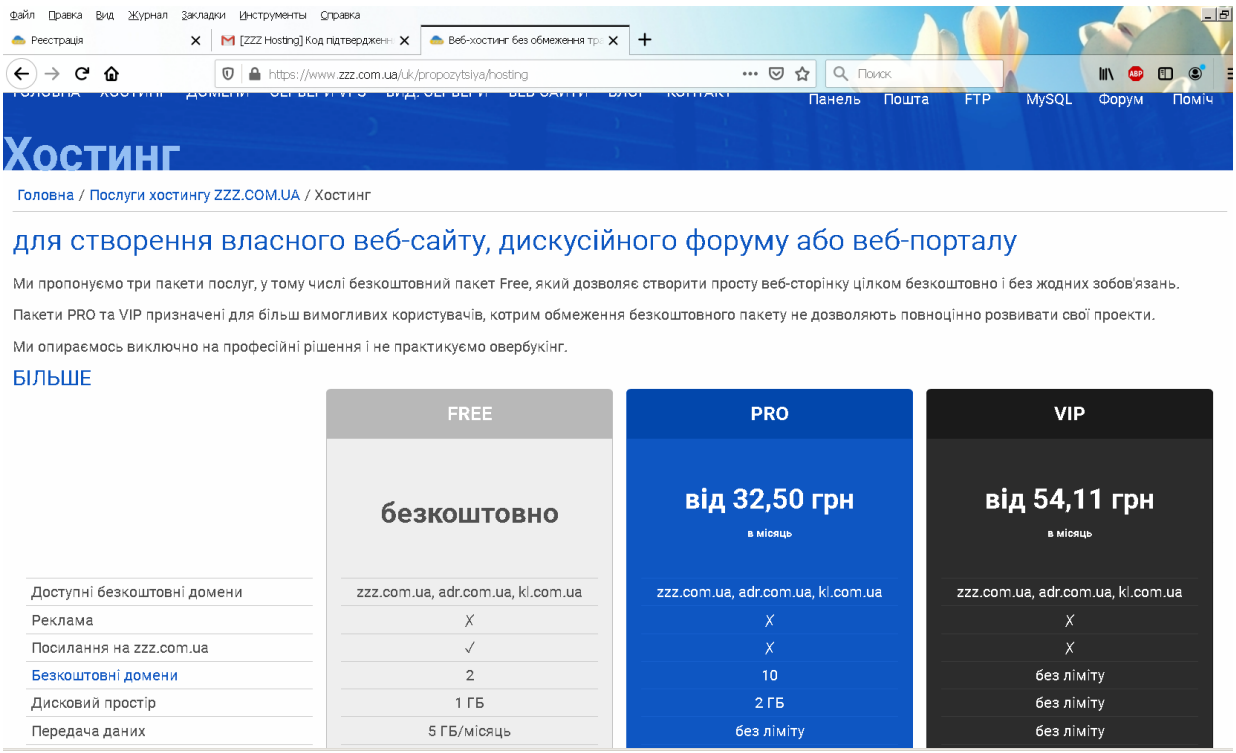


Рис.8.2. **Приклад безкоштовного хостингу**

Матеріали, що ви плануєте опублікувати на сайті, потрібно організувати у визначену структуру. Найчастіше для Web-сайтів обирається деревоподібна структура організації інформації. Коли сайт створено, його можна переглянути на власному комп'ютері. Для цього достатньо клацнути на піктограмі файлу index.html - головна сторінка сайту відкриється браузером. Щоб сайт був доступним для інших користувачів мережі, його потрібно розташувати на сервері. Цей процес називається публікацією сайту, або ***хостингом.***

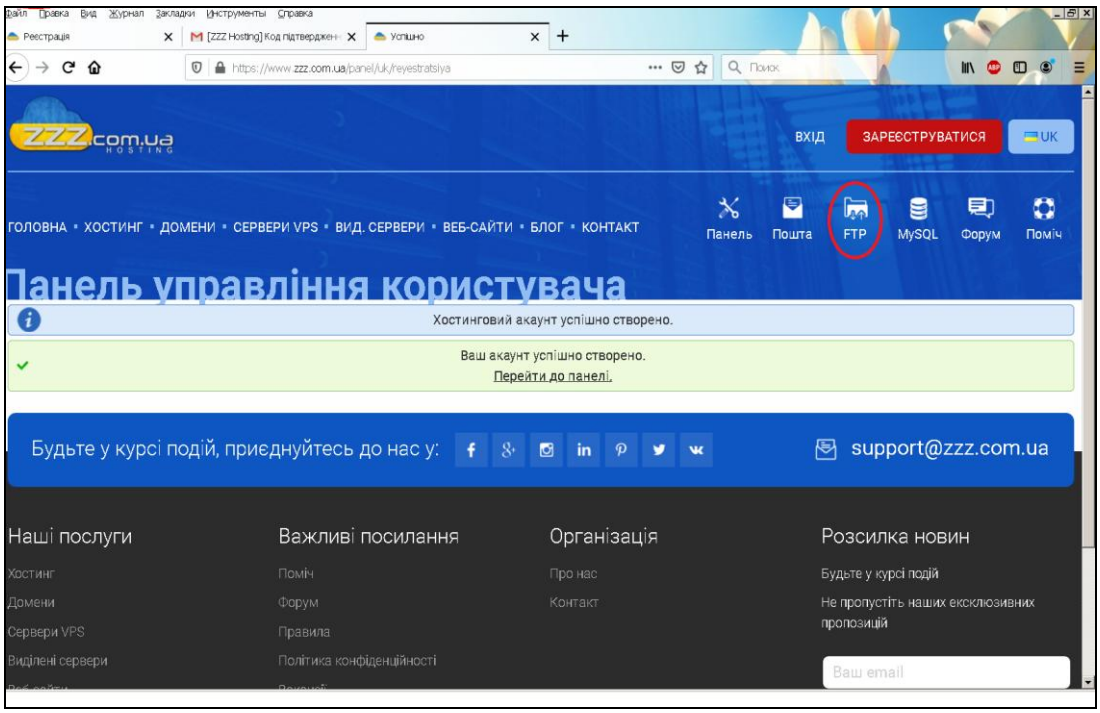


Рис.8.3. **Приклад FTP-з'єднання**

Під час публікування користувач пересилає html-файли з власного комп'ютера на сервер у режимі FTP-з'єднання. Для цього використовують деяку програму класу FTP-клієнт: Windows Commander, Far Manager тощо. Публікацію можна здійснити автоматично, користуючись FTP-сервісом, вбудованим у програми MS FrontPage, Macromedia DreamWeaver, у середовищі Windows, якщо доступний Майстер Web-видань а також за допомогою засобів автоматизації процесу публікації, які надають сервери.

Сервер, на якому розміщують сайт, називається ***хостом,*** а сервіси, які надають його служби, – ***хостингом.*** Хостинг буває безкоштовний або платним. Безкоштовний хостинг передбачає надання поштової web-скриньки і деякого обмеженого дискового простору для розташування сайту, причому на такому сайті автоматично буде розташована чужа реклама у вигляді банерів. Часто тут надаються засоби для створення сайтів на базі готових шаблонів, де-не-де є шаблонні гостьові книги, форуми, чати тощо. Платний хостинг дає змогу не лише уникнути чужої реклами на власному сайті, але й організувати свою рекламу на інших сайтах, використати додаткові сервіси

тощо. Якщо користувач є клієнтом місцевого web-провайдера, то доцільно сайт публікувати на його сервері. У мережі є велика кількість серверів, які надають безкоштовний хостинг. Увійшовши на деякий сервер, варто ознайомитися зі списком багатьох інших серверів, що надають безкоштовний хостинг, та умовами співпраці з ними. Для цього потрібно виконати пошук за словами „ безкоштовний хостинг " чи „free hosting" (рис.8.1) – отримаєте не тільки адреси серверів, але й корисну інформацію про обмеження і наявність чи відсутність тих чи інших сервісів на них.

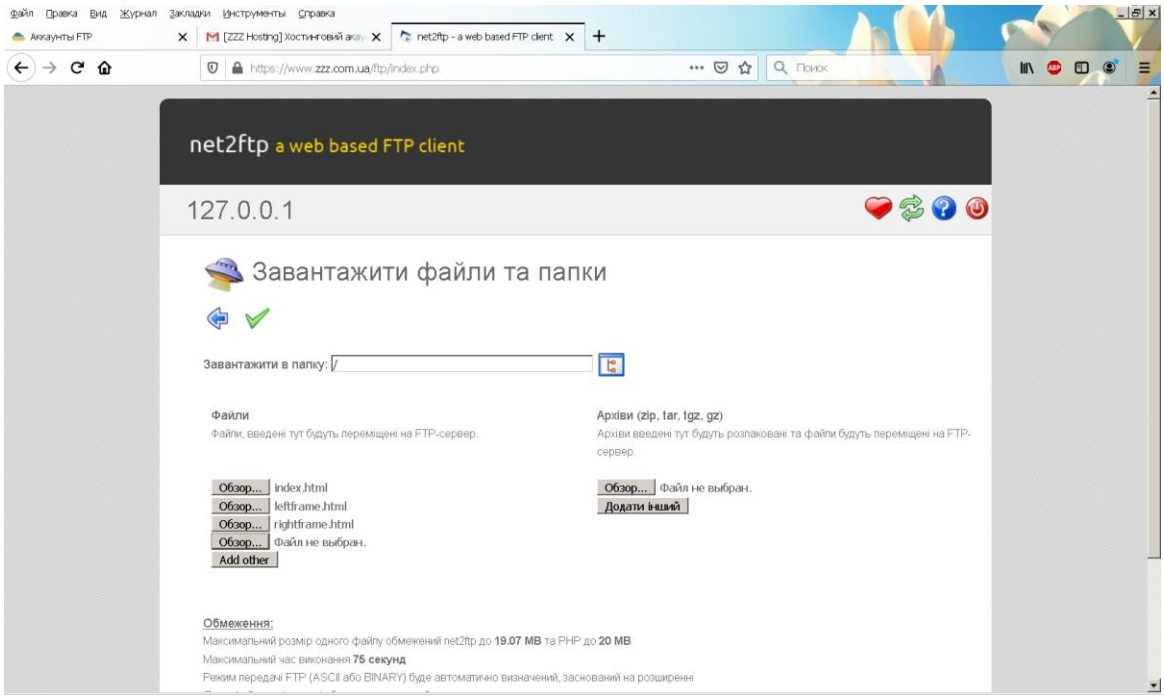


Рис.8.4. **Завантаження файлів при FTP-з'єднанні**

8.2. Реєстрація на сервері

Для користування платним чи безкоштовним хостингом користувач спочатку повинен зареєструватися на сервері за допомогою команди Реєстрація. На деяких серверах під час реєстрації потрібно зазначати власну електронну адресу, куди сервер відішле повідомлення про результати реєстрації, підтвердить адресу сайту, логін і пароль. Існує багато серверів, які надають безкоштовну електронну скриньку.

Логін (логічне ім'я) і пароль вказує користувач, дотримуючись вимог серверу, наприклад, логін може містити лише латинські літери і цифри та має починатися з літери; пароль має містити не менше чотирьох символів і не може збігатися з логіном. Під час заповнення реєстраційної карти пароль вводять двічі. Якщо користувач забуде пароль чи логін, він втратить доступ до сервісів, тому ці дані рекомендується занотовувати і зберігати в таємниці. Особливо забудькуваті можуть скористатися системою контрольного запитання і відповіді, поля яких варто заповнити під час реєстрації. На підставі цієї інформації сервер ідентифікує користувача і надає можливість ввести новий пароль. Деякі сервери надсилають на електронну адресу користувача втрачені дані. Приклади контрольних запитань: дівоче прізвище матері, ваша улюблена страва, поштовий індекс батьків. Можна сподіватись, що ця інформація відома не всім. Під час роботи з реєстраційною формою стежте, щоб були заповнені всі поля, позначені зірочками. Після заповнення реєстраційної форми потрібно натиснути на кнопку Ok і зачекати підтвердження реєстрації. Якщо введений користувачем логін уже зайнятий чи пароль був уведений неправильно, чи залишились незаповненими обов'язкові поля, то реєстрацію доведеться повторити. Якщо реєстрація пройшла успішно, можна розпочати публікацію сайту. Насамперед корисно ознайомитися з розділом Угода з користувачем, де є важлива інформація про умови надання послуг на сервері і правила поведінки користувачів - правила мережевого етикету.

Для публікації сайту можна використати сервер zzz.com.ua. Якщо сайт потрібно модернізувати чи внести незначні зміни, користуються розділом Завантажити файлами та папки(рис.8.3). Відкриється вікно менеджера файлів, що дає змогу створювати, редагувати, копіювати, переміщувати і вилучати файли, а також стежити за наявністю вільного простору на сервері. Найчастіше користуються командою Вилучити з метою вилучення файлів старого сайту, оскільки сервіс для заміни файлів (overwrite) не надається.

8.3. Розміщення сайту за допомогою програми FAR

Перед розміщенням сайту необхідно одержати у власника Web-серверу адресу URL, логін (ім'я) і пароль. Далі дійте за таким сценарієм.

Підключіться до Інтернету і завантажте програму FAR.

Натисніть клавіші Alt+Fl (F2) і виберіть у меню, що з'явилось, опцію FTP.

На панелі програми FAR ви побачите список FTP-з'єднань, встановлених з даного комп'ютера. Якщо ви користуєтесь FTP уперше, то даний список буде порожнім. Натисніть клавіші Shift+F4, у результаті чого з'явиться діалог для надання FTP-адреси

У верхній рядок діалогу введіть відповідно до шаблона логін, пароль і URL сайту. Логін і пароль розділяються двокрапкою, а пароль і URL - значком @. Замість імені сервера ви можете використовувати його IP-адресу, якщо вона вам відома, наприклад, 195.230.142.115.

Після того як ви розмістили свій сайт на Web-сервері, ви можете редагувати його змісту (додавати, видаляти, обновлювати файли). Якщо ви користуєтеся безкоштовним хостингом, то редагувати сайт можна за допомогою передбаченого для цього інтерфейсу.

Для внесення змін у сайт підключіться до Інтернету і запустіть програму FAR. Потім відкрийте панель FTP, натиснувши клавіші Alt+Fl або Alt- вибравши опцію FTP. Якщо сайт, що редагується, є в наведеному списку укажіть на нього курсором і натисніть Enter. Якщо ж потрібного з'єднання в списку немає, натисніть Shift+F4 і в наступному діалозі наберіть дані необхідні для створення нового з'єднання. Після цього програма FTP виведе на свою панель вміст каталогу зазначеного вами сайта. Тепер ви можете працювати з каталогом сайту так, ніби він знаходиться на диску вашого комп'ютера. Користуйтеся звичайними прийомами роботи у Norton Commander і FAR щоб скопіювати або видалити будь-які файли і каталоги.

Після редагування сайту не відключайтеся від Інтернету, а завантажте програму браузера і зайдіть на цей сайт. Перегляньте уважно, як позначилися внесені вами зміни на зовнішньому вигляді і роботі Web-сайту.

8.4. Реєстрація сайту у пошукових системах

Коли сайт зареєстровано на сервері, бажано щоб про його існування дізналися зацікавлені у ньому користувачі мережі. З цією метою різними засобами популяризують адресу і зміст сайту. Дедалі частіше адреси на зразок www.name.server можна побачити на екранах телевізорів, сторінках книг і журналів. Для успішного просування бізнес-сайтів фірм і компаній, а також пізнавальних сайтів, присвячених актуальним питанням науки, освіти, медицини тощо, їх рекомендують реєструвати у пошукових системах (каталогах, індексаторах). Пошукові системи призначені для відшукання у мережі сайтів з потрібної тематики чи з наявними у них ключовими словами. Таких систем є декілька десятків і безпосередня реєстрація сайту у кожній зайняла би багато часу.

Під час реєстрації потрібно заповнити анкету-форму, де зазначають адресу сайту, його логічну назву, короткий зміст, ключові слова, дані про автора, зокрема адресу електронної пошти тощо. Деякі каталоги не приймають безкоштовно на реєстрацію сайту, розташовані на серверах віддалених локальних провайдерів. Тому варто скористатися спеціалізованими системами реєстрації, які отримавши інформацію про сайт, беруть на себе функції реєстрування сайту в пошукових системах. Не всі послуги, пов'язані з реєстрацією, є безкоштовні. Після безкоштовними реєстрації користувач може отримати лист з пропозиціями щодо дальшого платного просування сайту чи проханням розташувати на своєму сайті посилання на реєстратора.

Важливий засіб популяризації сайту — банерна реклама й участь у системах обміну банерами. *Банер* — це графічний об'єкт стандартного

розміру 468x60, 100x100,120x60, 88x31, 125x125 пікселів, часто з елементами анімації, який наочно передає зміст сайту і служить гіперпосиланням на нього. Власний банер можна створити засобами комп'ютерної графіки, зокрема, за допомогою програм CoralDraw, PhotoShop та інших або скориставшись конструктором банерів на базі колекції готових шаблонів-картинок.

8.5. Оплата послуг в Інтернет

Для отримання платних послуг (обслуговування в інтернет-магазинах, участь у інтернет-аукціонах тощо) потрібно завести солідну кредитну картку, наприклад Visa, MasterCard, American Express тощо, і вводити її номер на відповідний запит системи.

8.6. Реклама в Інтернет

Реклама сайтів з'явилася практично одночасно із самою глобальною мережею. Сьогодні найвідомішими й ефективними видами реклами є банерна, контекстна й пошукова реклама.

Банерна реклама є однією з перших у мережі Інтернет. Сьогодні існує два її види:

1. Звичайна банерна реклама.
2. Участь у банерообмінній мережі.

У випадку звичайної банерної реклами, Ви просто оплачуєте показ свого банера на одному або декількох сайтах. Оплата береться, як правило, розраховуючи на 1000 показів, і визначається власником веб-ресурса, на якому розміщається банер.

Банерообмінна мережа – це об'єднання множини сайтів, з метою обміну показами своїх банерів. Принцип роботи банерообмінної мережі досить простий – Ви надаєте місце на своєму веб-ресурсі для показу банерів інших учасників мережі. Залежно від того, як часто на Вашому сайті

показуються банери інших учасників, Ви заробляєте покази свого банера на їхній веб-ресурсах. Власник мережі встановлює комісію на показ, що становить від 10 до 15 відсотків показів. Однієї з найбільших банерообмінних мереж у російському просторі Інтернету є мережа RLE, у якій за різним даними, походить від 35 до 55 мільйонів показів. Основною характеристикою ефективності банерної реклами є CRT(click through ratio). Цей показник відбиває співвідношення клацання по банеру до числа його показів. Сьогодні ж банерна реклама як і раніше є досить ефективною, але її популярність з року в рік падає.

Контекстна реклама – це спеціальний вид реклами в мережі Інтернет, що полягає в розміщенні Ваших рекламних оголошень на сторінках веб-ресурсів, близьких по тематиці Вашому сайту. Контекстна реклама є досить ефективної, тому що дозволяє охопити тільки цільову аудиторію, тобто тих користувачів, які так чи інакше зацікавлені у Ваших товарах або послугах.

Пошукова реклама є частковим випадком контекстної. Особливість її полягає в тому, що текст рекламного оголошення розміщається не просто на сторінках веб-ресурсів, а на сторінках пошукових систем, відповідно до введеного користувачем запита. Пошукова реклама є найефективнішої, і відповідно, самою дорогою. На сьогоднішній день самі великі пошукові системи, такі як Google надають рекламодавцям безліч способів і можливостей розміщення пошукової реклами на свої сторінках. Крім цих перерахованих вище видів реклами існує й деякі інші. Найбільш традиційної є реклама веб-ресурсу на спеціалізованих електронних дошках оголошень або рекламних порталів. Будь-яка, навіть найефективніша й дорога реклама Вашого сайту, не дасть тих результатів, які можна одержати, якщо використати її разом із просуванням сайту. Просування охоплює всі аспекти, що впливають на популярність сайту серед користувачів, і на його позиції в пошукових систем. Реклама сайту допомагає ще більше підсилити

ефективність просування, однак необхідно звернути увагу на оптимізацію сайту.

Оптимізувати сайт під вимоги пошукових систем – значить привести його до вигляду, що необхідний для правильної індексації пошуковими машинами. Цю частину роботи реклами в Інтернет ще називають внутрішньою оптимізацією сайту, тому що вона виявляє технічні помилки на сайті й виправляє їх, а також поліпшує внутрішні параметри сайту (теги), необхідні для успішної індексації. Щодня тисячі користувачів Інтернет шукають інформацію в пошукових системах, у тому числі й про Ваш товар. Якщо Вашого сайту немає в результатах пошуку, то більшість Ваших потенційних клієнтів ідуть на сайт Ваших конкурентів. Перед проведенням внутрішньої оптимізації сайту обов'язково проводиться ретельний *аналіз и аудит сайту*, який дає можливість виявити помилки сайту, спланувати стратегію оптимізації й просування.

Просування сайту в Інтернет виконує функції довгострокової кампанії реклами в Інтернет, що ще називають зовнішньою оптимізацією сайту. Для цього проводиться підвищення авторитетності в тематичному співтоваристві Інтернет:

1. відновлення інформації на сторінках сайту, іншими словами – перший крок до просування сайту, його підтримці,
2. обмін статтями й посиланнями з тематичними сайтами,
3. реєстрація в каталогах фірм, товарів і послуг,
4. відновлення повідомлень про сайт у соціальних мережах,
5. моніторинг положення Вашого сайту в пошуковій видачі,
6. моніторинг просування сайтів конкурентів.

Які можливості дає Інтернет реклама.

У нашому розумінні Інтернет реклама - маркетинговий комплекс, що забезпечує досягнення мети. І залежно від мети - комплекс може бути зовсім різним. Властиво ми:

1. Вміємо збільшувати продажі, за рахунок розвинених консалтингових сервісів.
2. Підвищуємо впізнання брендів за рахунок забезпечення масштабної повторюваності марки.
3. Забезпечуємо грандіозне охоплення аудиторії за рахунок партнерства із сотнями тисяч рекламних площ.
4. Реалізуємо складні моделі таргетинга, за рахунок власної технологічної платформи.
5. Ефективніше керуємо рекламними бюджетами наших клієнтів, забезпечуючи ефективну вартість Інтернет реклами.

Ефективна Інтернет реклама починається з формування мети й критеріїв її досягнення й закінчується аналізом результативності проробленої роботи

Дуже ефективною рекламою в Інтернет є пошукове просування сайту, тобто розміщення вашого оголошення (у даному виді послуг воно називається сніппетом) у так званій природній (не рекламної) видачі пошукової системи:

Плюси пошукового просування:

1. Ви звертаєтеся тільки до цільового аудиторії, яка шукає інформацію за допомогою уведення ключових слів;
2. Ви одержуєте довгостроковий ефект від реклами, тому що вже тільки одна лише підготовка сайту до просування дає свої результати, які залишаються з вами назавжди;
3. Користувачі Інтернет не розглядають пошукове просування, як рекламу, а ви одержуєте рекламу без негативного "рекламного" ефекту;

4. Ціни на такий вид Інтернет реклами порівняно невисокі.

Однак при всіх плюсах пошукової реклами сайту є істотне обмеження, про яке необхідно замислюватися - час необхідний для досягнення високих результатів може досягати декількох місяців, що часто не підходить для короткочасних акцій.

II. Контекстна реклама сайту представляє собою розміщення в рекламній зоні видачі пошукової системи, та існує за іншими законами, ніж пошук

Плюси контекстної реклами сайту:

1. Ви можете дуже швидко змінювати вміст ваших рекламних оголошень;

2. Можна також швидко налаштувати рекламу на різні регіони й показувати в різний час, що часто дозволяє ефективно використати бюджет;

Однак ключовим обмеженням у ряді галузей є вартість контекстної Інтернет реклами й часто низька комунікабельність порівняно з пошуковим просуванням. Щоб отримати бажаного результату й бути конкурентоздатним в Інтернет необхідно ефективно використовувати інструменти Інтернет маркетингу в комплексі.

Комплексна Інтернет реклама, завжди краще рішення. Ціни на Інтернет рекламу формуються з декількох складових.

1. Створення Інтернет реклами.

2. Вартість розміщення Інтернет реклами звичайно є основною статтею витрат.

3. Вартість послуг консультантів становить відсоток від вартості розміщення.

Питання для самоконтролю

1. Що таке хостинг?
2. Яка послідовність дій користувача має бути при розміщенні сайту на безкоштовному Web-сервері?
3. Що розуміється під FTP-клієнтом і FTP-сервером?
4. Як розмістити сайт на сервері за допомогою FTP?
5. Як виконується редагування сайту в програмі FAR?
6. Які види реклами ви знаєте?

СПИСОК ЛІТЕРАТУРИ

1. Цеслів О.В.Інформатика: Навч. посіб. – К.,2011–330 с.
2. Цеслів О.В. WEB програмування: Навч. посіб. – К.,2011–298 с.
3. Microsoft Visual Basic 6.0 для профессионалов. Шаг за шагом: Практ.пособ./Пер. с англ.–М.: Издательство ЭКОМ,2005.–720 с.
4. Глинський Я.М. Visual Basic. Навч.посібник. – Львів:Деол, 2005.
5. Пономаренко В.С. Інформаційні системи і технології в економіці.–К.:–2002.–542с.
6. Глинський Я.М., Ряжська В.А. Internet.Сервіси, HTML і web-дизайн. Навч.посібн.-Львів,2003.-192 с.

Предметний покажчик

Навчальне видання

Цеслів Ольга Володимирівна

Основи програмування та веб-дизайн

Навчальний посібник