Міністерство освіти і науки України

Харківська національна академія міського господарства

І.Л. Яковицький

Конспект лекцій з дисципліни

«Комп'ютерна техніка та програмне забезпечення»

(для студентів 2 курсу денної форми навчання освітньо-кваліфікаційного рівня бакалавр у галузі знань 0507 - «Електротехніка та електромеханіка» за напрямом підготовки 6.050701 – «Електротехніка та електротехнології», спеціальності 6.090600 - «Електротехнічні системи електроспоживання»)

Харків - 2009

УДК 681.3.06
«Комп'ютерна техніка та програмне забезпечення». Конспект лекцій для студентів 2 курсу денної форми навчання освітньо-кваліфікаційного рівня бакалавр у галузі знань 0507 - «Електротехніка та електромеханіка» за напрямом підготовки 6.050701 – «Електротехніка та електротехнології», спеціальності 6.090600 - «Електротехнічні системи електроспоживання» / І.Л. Яковицький – Харків: ХНАМГ, 2009. – 89 с.

Анотація:

Повсякденно людина займається обробкою інформації й прийняттям рішень. У виграші виявляється той, хто найбільше ефективно вміє обробляти інформацію: чітко розуміє завдання предметної області; вміє визначити вимоги до вихідної інформації; має коректні й швидкісні алгоритми (способи) вирішення завдань предметної області.

Обсяги інформації збільшуються, алгоритми рішення завдань ускладнюються, тому технічне забезпечення процесу обробки інформації стає вирішальним елементом. До нього в першу чергу належить обчислювальна техніка.

Технічні та технологічні досягнення, жорстка конкуренція тримають виробників обчислювальної техніки в постійній напрузі. Але більш важливим для зростання у галузі інформаційних технологій є усвідомлення значною кількістю людей ідеї процесу обробки інформації, а також відмінна якість людини - досягнення мети не зупиняє, а дозволяє визначити нові цілі.

Головна мета навчання: розвиток уявлень про процес обробки інформації та навичок інформаційного структурування завдань предметної області; вміння вибирати ефективні функціональні можливості програмного забезпечення для вирішення завдання предметної області.

У сучасній термінології завдання обробки інформації поєднують за назвою "Інформаційні технології". Варто підкреслити істотну відмінну рису цієї дисципліни від класичних курсів з вищої математики, теорії ймовірностей і математичній статистики. Якщо відволіктися від предмета дисципліни і покласти визначення алгоритму на діяльність людини як таку, то з'ясується, поведінку людини визначає набор правил; чинені дії, свідомо чи несвідомо, мають певний порядок (послідовність), а їх виконання залежить від деяких умов (подій). Визначні послідовності дій людина запам'ятовує і повторює їх знову і знову. Настає етап, коли ми обмірковуємо послідовність операцій і дій за для досягнення будь-якої мети, оцінюємо можливі наслідки й таке інше. Це наштовхує на думку - розвиток алгоритмічного як і математичного мислення є вкрай важливим для фахівців у будь-якій сфері діяльності.

Програмування це справжній полігон для розвитку алгоритмічного мислення. Є усі складові - формальна мова опису дій та пристрій, що беззаперечно виконує їх обчислювальна система.

Ми можемо навчитися будувати нашу поведінку, аналізувати її результати, моделювати можливі наслідки.

Рецензент: завідувач кафедрою прикладної математики та інформаційних технологій, проф. д.т.н. Самойленко М.І.

Затверджено на засіданні кафедри прикладної математики та інформаційних технологій, протокол №1 від 28 серпня 2008 р.

© Яковицький І.Л., ХНАМГ, 2009

Зміст

РОЗДІЛ 1. АПАРАТНЕ І ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ. ОСНОВНІ ПОНЯТТЯ І ВИЗНАЧЕННЯ

1.1. Вступ. Історична довідка

Об'єкти реального світу постійно обробляють інформацію або обмінюються нею.

Людина займає вищу сходинку розвитку за методами і способами обробки інформації. Завдяки інтелектуальній діяльності, знання людства (інформація) продукують нові знання (інформацію). Це унікальний процес, тому що його практично неможливо обмежити.

Діяльність людини це часте повторення схожих дій. І вона прагне оцінити порядок або послідовність їх виконання для досягнення результату. Наступний крок у цьому процесі це створення **автомата (пристрій, що працює за певними правилами)**. Один з перших автоматів «Осел, який витягує воду з колодязя».

У спробі зрозуміти картину навколишнього світу й усвідомлення себе в ньому, людина звернулася до астрономії. Основним інструментом у дослідженнях стала математика і її апарат. Обсяг однотипних обчислень забирав величезну кількість часу й сил дослідника.

У 1642 р. французький математик **Блез Паскаль** сконструював першу рахункову машину для обчислення двох арифметичних дій - додавання і віднімання.

Пізніше, в 1670 р. німецький математик **Готфрід Вільгельм Лейбніц**, творець диференціального числення, сконструював рахункову машину для виконання операцій додавання, віднімання, множення, ділення, зведення в ступінь і добування квадратного й кубічного коренів.

У 1834 р. англійський математик **Чарльз Беббідж** розробив аналітичну машину (табулятор), введення даних в яку і програмування операцій проводилися за допомогою перфокарт, результати видавалися на записуючий пристрій або на перфокарти.

Передумовою для створення електронної обчислювальної машини (ЕОМ) з'явилося створення електричного елемента, який може перебувати у двох стійких електричних станах. Цей пристрій – тригер.

Перша сучасна ЕОМ «UNIAC» була розроблена й побудована в 1949 р. у США під керівництвом американського математика **Джона фон Неймана**.

У нашій країні роботи зі створення ЕОМ проводив математик **Віктор Михайлович Глушков.** Перша ЕОМ «Стріла» була створена в Києві в 1951 р. Власне дослідження в області обчислювальної техніки проходили, в нашій країні й за кордоном, паралельно аж до початку 60-х років. «Гонка за лідером» була безкомпромісною. Результат у наявності, - створення потужної аерокосмічної галузі. Однак на якомусь етапі керівництво держави прийняло вкрай згубне рішення, що підірвало всі наші досягнення. Спецслужбам країни вдалося провести операцію придбання й доставки в країну ЕОМ загального призначення IBM/360. Машина була розібрана й у стислий термін створена її копія ЄС ЕОМ 1020. Із цього моменту ми були приречені на постійне, все більше відставання в галузі комп'ютерної техніки й інформаційних технологій.

Надалі розвиток техніки пішов шляхом мініатюризації завдяки досягненням фізики напівпровідників. У 1980 р. «Блакитний гігант» - фірма IBM випускає перший персональний комп'ютер для бізнесу.

Власне в цьому стані ми перебуваємо і сьогодні. Потужність обчислювальних машин збільшується, що дозволяє обробляти все більш різноманітну інформацію.

Визначення 1. ЕОМ, Персональний комп'ютер (ПК) – автомат для обробки інформації.

Визначення 2. Електронна обчислювальна машина (ЕОМ) – машина, що виконує задану послідовність команд.

1.2. Архітектура електронної обчислювальної машини

Про які команди може йти мова?

Окрема **команда** служить для виконання таких **операцій**, як, наприклад:
додавання чисел;
пересилання інформації з однієї комірки пам'яті в іншу;
логічні операції порівняння;
зміна послідовності виконання команд.

Хто виконує ці команди?

Визначення 3. **Процесор (CPU – central processor unit)** – елемент, частина ПК, пристрій, що виконує команди. Команди, що виконуються, і дані, для обробки розміщуються в спеціальні (іменованих) **гніздах** процесора - регістрах.

Процесор – основний пристрій для обробки й перетворення інформації, однак простір для її зберігання у нього вкрай обмежений, крім того, він дуже дорогий. При проектуванні обчислювального пристрою керуються принципом функціональної декомпозиції. Процесор – функціональний пристрій обробки інформації, і взаємодіє з іншим функціональним пристроєм - пристроєм зберігання інформації.

Де зберігається інформація, яку обробляє процесор?

Інформація, яку обробляє процесор, розміщується в пам'яті обчислювального пристрою. Пам'ять обчислювального пристрою функціонально й фізично розділяється на постійну пам'ять (ROM – read only memory / пам'ять тільки для читання) і оперативну пам'ять (RAM – random access memory / пам’ять вільного доступу, обновлювану, доступну для запису й зчитування інформації). Не можна однозначно сказати, яка з них важливіше, всі елементи з погляду обчислювальної системи необхідні. Але щодо користувача, оперативна пам'ять виражає обчислювальну потужність системи в цілому.

Визначення 4. **Оперативна пам'ять (RAM – random access memory)** - елемент, частина ПК, пристрій, в якому розміщається інформація для обробки процесором.

Яка організація пам'яті?

Оперативна пам'ять умовно «розбита» на **комірки,** кожна з яких має власну **адресу**.

Визначення 5. **Комірка** – найменша логічна одиниця ОП. Коміркою пам'яті сучасного ПК є байт.

Визначення 6. **Адреса -** унікальний номер (число), за допомогою якого виконують звертання до інформації.

Визначення 7. **Байт (Byte) –** одиниця виміру обсягу пам'яті, найменша логічна одиниця ОП.

Визначення 8. **Біт (Bit) –** мінімальна одиниця виміру *ІНФОРМАЦІЇ*.

Достоїнства ОП:

висока швидкість обміну й доступу до інформації.

Недоліки ОП:

зміст оперативної пам'яті, тобто власне ІНФОРМАЦІЯ втрачається після відключення живлення;
обмежений обсяг;
висока вартість.

Як усувають недоліки ОП?

Для усунення недоліків оперативної пам'яті й розширення можливостей зберігання інформації використовують магнітні й оптичні носії інформації, які обслуговуються відповідними пристроями. У деяких випадках власне носій інформації є складовою частиною пристрою, який його обслуговує.

> ***Визначення 9.*** Магнітні й оптичні пристрої (носії) зберігання інформації – призначені для тривалого зберігання інформації.

Як інформація передається між пристроями ПК – як відбувається обмін інформацією?

Для передачі інформації від одного пристрою іншому використовується **шина даних.**

> ***Визначення 10.*** **Шина даних –** електричні лінії зв'язку, по яких пристрої обмінюються інформацією між собою.

В основі обміну інформацією між пристроями лежить:

паралельне підключення (приєднання) пристроїв до шини даних;
адресація пристроїв;
механізм переривань.

> ***Визначення 11.*** **Адреса пристрою –** унікальний номер (число), призначуваний системою пристрою, і використовуваний для звертання до нього**.**
>
> ***Визначення 12.*** **Переривання –** стан обчислювальної машини, при якому, по шині даних передається широкомовний сигнал «Увага».

1.3. Поняття про інформацію

Людина безперервно обробляє інформацію і приймає різні рішення. У виграші виявляється той, хто може ефективно обробляти інформацію:

чітко розуміє зміст завдання предметної області;
уміє визначити вимоги до вихідної інформації;
має коректні й швидкісні способи (алгоритми) розв'язання завдань предметної області.

Визначення 13. **Інформація** - сукупність символів (образів), що несуть смислове навантаження.

Визначення 14. **Інформація** - сукупність відомостей (даних), що сприймається з навколишнього середовища (вхідна інформація), видається в навколишнє середовище (вихідна інформація) або зберігається в якому-небудь вигляді для наступного використання.

Визначення 15. **Інформація** (від латинського *informatio* – роз'яснення, виклад) - загальнонаукове поняття, що включає обмін відомостями між людьми, людиною і автоматом, автоматом і автоматом; обмін сигналами у тваринному й рослинному світі; передачу ознак від клітки до клітки, від організму до організму. (Енциклопедія «Кирила і Мефодія»).

Інформацію можна розглядати з трьох точок зору:

з поведінкової точки зору. Створення порції інформації здійснюється з деякої причини, а одержання цієї інформації може привести до деякого результату (спостережуваній дії або розумової операції);

з математико-лінгвістичної точки зору. Порція інформації може бути описана шляхом співвіднесення її з іншою інформацією, вказівки її змісту й структури;

з фізико-технічної точки зору. Розглядаються фізичні аспекти прояву інформації - її матеріальний носій, розв’язна здатність і точність, з якими вона фіксується, кількість інформації, яка виконується, передається і т.д.

Визначення 16. **Форми інформації**. Інформація існує в різноманітних формах:

- у формі світлових, звукових або радіохвиль;
- електричного струму або напруги;
- магнітних полів;
- знаків на паперовому носії:
- документів;
- креслень,
- рисунків.

Визначення 17. Властивості інформації:

- об'єктивність;
- повнота;
- вірогідність;
- адекватність;
- актуальність;

- доступність.

Визначення 18. Операції. Інформацію можна

- створювати,
- передавати,
- ухвалювати,
- запам'ятовувати,
- шукати,
- копіювати (у тієї або іншій формі),
- обробляти,
- руйнувати.

Визначення 19. Дані - складова частина інформації, що являє собою зареєстровані сигнали.

Визначення 20. Операції обробки даних:

- збір даних – нагромадження інформації для забезпечення повноти при ухваленні рішення,
- формалізація даних – приведення різнорідних даних до єдиної форми,
- фільтрація даних – усунення «зайвих» даних,
- сортування – упорядкування за відповідною ознакою,
- архівація – збереження у зручній і доступній формі,
- захист – комплекс заходів для запобіганню втратам і руйнуванню даних,
- транспортування – приймання, передача між користувачами інформаційного процесу,
- перетворення – зміна форми, структури, типу носія.

Як організується обмін інформацією між користувачем і ПК?

Для обміну інформацією між користувачем і ПК існують спеціалізовані пристрої, що реалізують операції перетворення відповідного типу інформації у строго певний вид.

Ця операція називається **кодуванням.**

Зворотна операція називається **декодуванням.**

На початковому етапі своєї історії ПК використовувалися для обробки текстової і числової інформації. Набір знаків (символів), за допомогою якого записувалися тексти відповідною мовою, звичайно, обмежений.

1.4. Системи числення

Визначення 21. **Система числення –** спосіб записи чисел у вигляді послідовності цифр і символів певного алфавіту.

Визначення 22. **Позиційна система числення –** система, в якій при записі числа, значущість (інтерпретація) символу залежить від його місця розташування.

Десяткова система числення (позиційна арабська система)

Цифри системи (алфавіт) – 0,1,2,3,4,5,6,7,8,9

Основа системи – 10

2 1 0

$1\ 5\ 3 = 1*10^2+5*10^1+3*10^0$

Двійкова система числення (позиційна)

Цифри системи (алфавіт) – 0,1

Основа системи - 2

4 3 2 1 0

$1\ 0\ 1\ 1\ 0_{<2>} = 1*2^4+0*2^3+1*2^2+1*2^1+0*2^0 = 22_{<10>}$

Шістнадцятирична система числення (позиційна)

Цифри й символи системи (алфавіт) – 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Основа системи - 16

1 0

$F\ 3_{<16>} = 15*16^1+3*16^0 = 243_{<10>}$

$F\ 3_{<16>} = 11110011_{<2>}$

Восьмирична система числення (позиційна)

Цифри системи – 0,1,2,3,4,5,6,7

Основа системи - 8

1 0

$5\ 6_{<8>} = 5*8^1+6*8^0 = 46_{<10>}$

№	Назва	Тип	Алфавіт	Основа

1	Десяткова	позиційна	0,1,2,3,4,5,6,7,8,9	10
2	Двійкова	позиційна	0,1	2
3	Шістнадцятирична	позиційна	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F	16
4	Восьмирична	позиційна	0,1,2,3,4,5,6,7	8

Визначення 23. **Код ASCII** (American Standard Code Interleave Information – американський стандартний код для обміну інформацією) - стандартна система кодування знаків, уведена в 1963 р. Це семирозрядний двійковий код, що забезпечує 128 різних комбінацій. Для кодування інформації використовується розширений код ASCII. Розширений код ASCII - це восьми розрядний код, що забезпечує 256 різних кодів.

Таким чином, можна сказати, що байт - це обсяг інформації, необхідний для зберігання одного друкованого символу.

1.5. Класифікація математичного (програмного) забезпечення ПК

1.5.1. Базове програмне забезпечення

Базова система введення виводу: BIOS

Операційна система:Сімейство Windows: 98,NT,Me,2000,XP,Unix,Linux,Freebsd

1.5.2. Засоби обслуговування інформації

Файловий менеджер Total Commander, FAR manager

Архіватор Winzip, Winrar, 7-zip

Антивірусний пакет: AVP (лабораторія Касперського), Avast, Nod32

1.5.3. Пакет офісних додатків

Офісний пакет Microsoft Office,Openoffice, Staroffice

Word - текстовий процесор

Excel - процесор електронних таблиць

Access - система керування базою даних

Powerpoint - процесор підготовки презентацій

1.5.4. Засоби розробки програмного забезпечення

Системи і мови програмування - C/C++/C#, Java, Delphi

1.5.5. Спеціалізоване програмне забезпечення

Засоби обробки фотографій - Photoshop

Засоби розробки графічних матеріалів - Coreldraw

Засоби анімації і мультимедіа

1.5.6. Засоби автоматизованого проектування

Autocad, Archicad, 3D Studio Max, Maya,

1.5.7. Ігрові й розважальні програми

1.5.8. Базова система введення виводу

Визначення 24. **BIOS(Base Input Output System - базова система введення виводу)** ПК – набір зв'язаних програмних модулів, який забезпечує керування апаратними ресурсами обчислювальної системи на «низькому рівні» (рівні електричних сигналів).

1.6. Операційна система

Визначення 25. **Операційна система** – набір взаємозалежних програмних модулів, який забезпечує:

- керування апаратними й програмними ресурсами обчислювальної системи;
- інтерпретацію і виконання запитів користувача до інформації, що зберігається.

1.6.1. Файлова система.

Визначення 26. **Файлова система** - сукупність файлів (документів) і набір операцій роботи з ними. Файлова система є складовою частиною операційної системи.

Визначення 27. **Файл** - логічно завершена (ціла) частина інформації, записана на дисковому пристрої й доступна для багаторазового використання.

Незалежно від внутрішньої організації інформації, файл має атрибути:

Основний атрибут	<ім'я файлу>	{ NAME }
Інформаційні	<обсяг>	{ SIZE }
	<дата створення>	{ DATE }
	<час створення>	{ TIME }
Атрибути володіння		
Атрибути доступу		

За організацією інформації файли різняться

ТЕКСТОВІ	ДВІЙКОВІ
Командні	Програми
Файли даних	Файли даних
Вихідні тексти програм	Спеціальні файли
Тексти	

Ім'я файлу.

Визначення 28. **Ім'я файла** є основним його атрибутом й використовується для звертання до інформації, що зберігається у файлі при виконанні різних операцій над нею.

Правила формування імені файла.

<filename>[.<extension>]

<filename>	- ***основне ім'я файла*** - послідовність літер і/або цифр від 1 до 8(32, 256) символів
<extension>	- ***розширення імені*** - послідовність літер і/або цифр від 1 до 3 символів

Визначення 29. Розділяє ім'я файла й розширення імені символ « . » (крапка).

Визначення 30. Ім'я файла й розширення імені не можуть містити наступні символи: « . » крапка / « , » кома / « \ » зворотна похила риска / « / » похила риска / « : » двокрапка / « ; » крапка з комою.

Розширення імені файла може бути пропущене, але воно «свідчить» про призначення файла або його внутрішню організацію. Крім того, зареєстровані типи

файлів у рамках операційної системи зв'язують із програмами, які можуть обробити інформацію, що зберігається в них. Наведемо загальновідомі розширення імен файлів:

.c	- вихідний текст програми мовою програмування С (сі)
.cpp	- вихідний текст програми мовою програмування С++ (сі плюс плюс)
.cs	- вихідний текст програми мовою програмування С# (сі шарп)
.pas	- вихідний текст програми мовою програмування PASCAL
.vb	- вихідний текст програми мовою програмування VISUAL BASIC
.js	- вихідний текст програми мовою програмування Java
.txt	- текстовий файл
.doc	- текстовий документ, який сформований програмою Microsoft Word
.xls	- електронна таблиця, який сформований програмою Microsoft Excel
.mdb	- таблиця бази даних, яка сформована програмою Microsoft Access
.ppt	- презентація Microsoft Powerpoint
.sys	- файл настроювання
.cfg	- файл настроювання (конфігурації)
.hlp	- двійковий/текстовий файл, допомога по роботі з конкретною програмою
.zip	- двійковий файл, результат роботи програми ущільнення інформації
.rar	- двійковий файл, результат роботи програми ущільнення інформації
.lib	- бібліотека функцій
.dwg	- рисунок, виконаний у системі AUTOCAD

Спеціальні розширення імені файлу

Деякі розширення імені файла є спеціальними. Операційна система ПК, обробляючи ці файли, виконує строго певні дії.

Файли, що мають наступні розширення:

.bat	командний файл, який приймає управління ресурсами (пристроями) ПК для виконання послідовності команд
.exe	програма, яка підготовлена до виконання (додаток), реалізує алгоритм розв’язання завдання (обробки інформації)

.com	, яка підготовлена до виконання (додаток), реалізує алгоритм розв'язання завдання (обробки інформації)
.dll	динамічна бібліотека функцій додатка

Папки (Каталоги)

Логічно зв'язану інформацію користувачі групують за допомогою папок (каталогів).

> ***Визначення 31.*** **Папка/Folder (Каталог/Directory)** - загальне ім'я для набору (групи) логічно зв'язаних файлів.
>
> ***Визначення 32.*** **Ім'я каталогу(папки)** - послідовність літер і/або цифр від 1 до 8 (32, 256) символів.
>
> ***Визначення 33.*** Роздільники: ".", ",", "/", ";", ":", " " в ім'я каталогу включати не можна.
>
> ***Визначення 34.*** **Підкаталог (Sub-Directory)** - каталог у каталозі (принцип російської матрьошки). Вкладеність каталогів практично не обмежена.
>
> ***Визначення 35.*** Кореневий каталог дискового пристрою – ім'я пристрою, з символом « \ »

Угода про іменах дисків

Для звертання до різних дискових пристроїв операційна система використовує букви латинського алфавіту зі знаком « : » (двокрапка), відповідно до угоди:

A: ; B: - імена пристроїв для роботи з дискетами;

C: . . Z: - імена пристроїв для обслуговування магнітних (твердих), оптичних дисків, (будь-те фізичних і логічних), змінних електронних носіїв інформації великої ємкості (flash disk)

Повне ім'я файла

> ***Визначення 36.*** Для звертання до файла з будь-якого місця файлової системи використовують його повне ім'я.

[d:\] [path\] filename[.extension]

d:\	- ім'я корневого каталогу пристрою для зберігання інформації
path\	- ШЛЯХ до файла (перерахування імен вкладених один в одний каталогів, розділених символом "\")

filename	- основне ім'я файла
extension	- розширення імені.

Визначення 37. ШЛЯХ до файла (перерахування імен вкладених один в одний каталогів, розділених "\".

Визначення 38. Файлова система ПК не містить двох файлів з однаковим повним іменем.

РОЗДІЛ 2. СВІТ ІНТЕРНЕТ

2.1. Що таке Всесвітня Павутина

Всесвітня Павутина - особа сучасної Мережі. Поява цієї служби зробила роботу з інформацією легкою і приємною, залучило до лав користувачів Мережі сотні мільйонів людей.

Щоб довідатися про новини, навчитися чого-небудь або просто розважитися, люди дивляться телевізор, слухають радіо, читають газети, журнали, книги. Всесвітня Павутина теж пропонує своїм користувачам радіомовлення, відеоінформацію, пресу, книги, тільки з тією різницею, що не треба вибирати в телевізійній програмі цікавий фільм або передачу, немає необхідності бігти в газетний кіоск за журналом або їхати за потрібною книгою в бібліотеку. Досить включити комп'ютер і вийти в Інтернет. Кілька клацань мишею перенесуть для вас потрібні відомості з одного континенту Землі на інший, а Ви цього й не помітите, з інтересом переглядаючи в електронній газеті статтю з подробицями про важливу подію або вивчаючи зміни курсів світових валют у фінансових новинах. Одночасно Ви слухаєте улюблену радіопередачу; одне клацання мишею - і перед Вами на екрані фотографія кита, зроблена п'ять хвилин назад екіпажем дослідницького корабля десь в Атлантиці. На вечір у вас запланована прогулянка у віртуальний світ - не можна упустити можливість у спокійній обстановці поспілкуватися з діловими партнерами із Китаю...

Суттєво не тільки те, що одержати доступ до інформації у Всесвітній Павутині легко й просто: тут це можна зробити тоді, коли треба. Напевно, у вашому житті бувало так, що в бібліотеці траплявся санітарний день саме тоді, коли Ви знайшли час, щоб взяти там потрібну книгу; або іноді виявлялося, що цікава передача по телевізору збігається за часом із сімейним обідом. У Всесвітній Павутині все по-іншому: тут Ви вирішуєте, коли і яку інформацію Ви прагнете одержати. Якщо Ви весь день працюєте, і не маєте часу для того, щоб ходити на лекції в університет, Інтернет допоможе Вам отримати освіту заочно. Прагнете більше довідатися про те, як користуватися комп'ютером, удосконалити знання іноземної мови? - Всесвітня Павутина знову прийде на допомогу. Тут Ви знайдете потрібну інформацію й зможете підібрати підходящий для вас курс навчання - віртуальний або реальний.

Розвиток Всесвітньої Павутини послужив початком триваючого дотепер буму Інтернет. Однак причина блискавичного поширення Мережі і її бурхливого зростання полягає не стільки в тому, що її служби пропонують зручні засоби для обміну інформацією й доступу до неї, скільки в тому, що в Мережі є потрібна інформація. Крім того, часто виявляється, що цю інформацію не можна одержати в іншому місці. Тим часом, інформаційні ресурси, ті десятки мільйонів документів, яким Всесвітня Павутина зобов'язана своєю привабливістю для користувачів, створюють самі користувачі. Чим більше користувачів - тим більше інформації, і навпаки: зі швидкістю гірської лавини цивілізація переходить у нову інформаційну еру. А річ у тому, що для публікації інформації у Всесвітній Павутині користувачеві потрібний тільки комп'ютер, підключений до Інтернет, і мінімум навичок роботи з ним. Кілька хвилин роботи - і повідомлення або документ стає доступним усім користувачам Мережі.

Працюючи із Всесвітньою Павутиною, Ви маєте доступ до маси найрізноманітніших за формою й змістом документів, творцями яких можуть виявитися й житель далекого Еквадору, і житель Грузії, і Ваш сусід по під'їзду. При роботі з мережею взагалі, і із Всесвітньою Павутиною, зокрема, пам’ятайте, що за кожним ресурсом стоять конкретні люди, з якими Ви при бажанні легко можете вступити в контакт - до взаємної вигоди обох сторін. Про цю особливість, пов'язану з публікаціями в Мережі, часто говорять у контексті організації зворотного зв'язку з користувачем при створенні ресурсів, але вона, безсумнівно, буде вам цікава і як користувачеві.

Не треба також забувати, що, крім власне інформаційного навантаження, Всесвітня Павутина й Інтернет несуть важливу соціальну функцію. Тут зустрічаються один з одним люди із самих різних куточків планети, формується нова культурна спільність, не зв'язана державними кордонами й національною приналежністю. Унікальна загальнодоступність інформації й засобів її поширення, характерні для Всесвітньої Павутини, створюють своєрідну електронну демократію. Окремий уряд, організація або людина не можуть повністю контролювати технічну інфраструктуру, усі додатки або все інформаційне наповнення Мережі. Правда, уже сьогодні багато говорять про перевагу, яку користувачі Мережі мають стосовно тим, хто позбавлений підключення до Інтернет і навичок роботи з Мережею, і про нерівність, що виникає звідси, різних можливостей. Докладають значні зусилля, щоб забезпечити вільний доступ до Мережі всім бажаючим: багато бібліотек безкоштовно надають своїм

читачам можливість працювати в Мережі; на вулицях міст з'являються так звані Інтернет-Кіоски, де за невелику плату можна попрацювати із Всесвітньою Павутиною.

2.2. Що таке гіпертекст

Термін «Всесвітня Павутина» - дослівний переклад англійської назви World Wide Web. Набагато частіше вживаються англійські скорочення від World Wide Web - просто Web або WWW.

З технічної точки зору, World Wide Web складається з безлічі серверів, що надають користувачам доступ до спеціально складених, так званих гіпертекстових документів. Гіпертекст відрізняється від звичайного тексту тим, що включає гіпертекстові посилання (гіперпосилання), що з'єднують контекст, у якому вони перебувають, з іншим контекстом у рамках того або іншого тексту. Різновид гіпертексту, використовуваного на WWW, дає авторові документів додаткову свободу за рахунок можливості включати в текст документа ілюстрації, аудіо- і відеоінформацію, а також посилання на будь-які ресурси, що перебувають на довільному сервері Мережі. Назва «Всесвітня Павутина» дуже влучно передає образ зв'язків, що переплітаються, складаються у вигадливу структуру, яка охоплює безліч серверів по всій Земній кулі. Через можливість включати в документи мультимедійну інформацію, тобто звук, зображення й відео, застосовують термін «гіпермедіа».

При підготовці гіпертекстових документів для WWW текст спеціально розмічають за допомогою мови розмітки HTML (Hypertext Markup Language, що в перекладі й означає «мова розмітки гіпертекстів»).

Як і більшість інших служб, Інтернет, Всесвітня Павутина працює в рамках моделі клієнт-сервер. У якості сервера виступає постійно підключений до Мережі комп'ютер, на якому працює спеціальна програма – сервер(веб-сервер). Клієнт – будь-який комп'ютер, що працює в режимі постійного або сеансового підключення до Інтернет, на якому працює програма перегляду WWW - або браузер (англ. browser, від дієслова to browse - перегортати, листати). Браузер обробляє документ-HTML і відображає його на екрані. Обмін інформацією між веб-сервером і браузером здійснюється з використанням протоколу HTTP.

2.3. Браузер як основна програма доступу до служб Мережі

Протокол HTTP був запропонований швейцарським фізиком Тімом Бернерсом-Чи (Tim Berners-Lee) в 1989 р. У 1992 р. була створена перша програма-браузер. Але

справжня популярність прийшла до WWW у вересні 1993 р., коли в США, в університеті Штату Іллінойс був розроблений і випущений браузер Mosaic. Його успіх був обумовлений тим, що в ньому вперше була реалізована навігація за допомогою миші, і запропоноване розширення мови розмітки HTML, що дозволяє вставляти в гіпертекст зображення.

У квітні 1994 р. один з розроблювачів Mosaic Марко Андрессен (Marc Andreessen) заснував корпорацію Netscape Communications, а в грудні 1994 р. вийшла перша версія браузера Netscape Navigator, який завоював величезну популярність.

У 1996 р. в Netscape Navigator з'явився серйозний конкурент, Internet Explorer, браузер корпорації Microsoft, також заснований на браузері Mosaic.

Протокол HTTP, мова розмітки гіпертекстів HTML, браузери й веб-сервери безупинно удосконалюють, відкриваючи перед користувачами нові можливості.

Сьогодні браузер являє собою складне програмне забезпечення й поєднує легкість у використанні й багатство можливостей. Робота з браузером починається з вказівки в адресному рядку адреси (URL) того ресурсу, до якого користувач прагне одержати доступ. Браузер надсилає запит на зазначений сервер Мережі. У міру того, як із сервера приходять елементи зазначеної користувачем веб-сторінки, вона поступово з'являється в робочому вікні браузера. Процес одержання елементів сторінки із сервера відображається у «статусному» рядку браузера.

Текстові гіперпосилання, що містяться в отриманій веб-сторінці, як правило, дають кольором, відмінним від кольору іншого тексту документа, і підкреслюються. Посилання на ресурси, які користувач ще не переглядав, і посилання на вже знайомі ресурси звичайно мають різний колір. Зображення також можуть функціонувати як гіперпосилання. Незалежно від того, текстове посилання або графічна, якщо навести на неї курсор миші, його форма зміниться. Одночасно в статусному рядку браузера з'явиться адреса, на яку вказує посилання.

При натисканні на гіперпосилання браузер відкриває в робочому вікні ресурс, на який вона вказує, при цьому попередній ресурс із нього вивантажується. Дотримуючись вигадливих зв'язків, що ведуть від гіпертексту до гіпертексту і від одного сервера до іншого, користувач часом заходить в «нетрі» непотрібної інформації. Повернутися на правильну дорогу йому допомагає браузер, який веде список сторінок, що проглядаються. Якщо натиснути лівою клавішею миші на кнопку «Назад» («Back») у меню браузера, він повернеться до сторінки, яку Ви переглядали

до того, як відкрили поточний документ. Щоразу, коли Ви будете натискати на цю кнопку, браузер буде повертатися на один документ назад у списку переглянутих документів. Якщо Ви повернулися занадто далеко назад, скористайтеся кнопкою «Уперед» («Forward») меню браузера. Вона допоможе вам переміститися вперед за списком документів. Кнопка «Стоп» («Stop») зупинить завантаження документа. Кнопка «Обновити» («Reload») дає можливість перезавантажити документ із сервера.

Браузер дозволяє роздрукувати завантажену веб-сторінку за допомогою кнопки «Печатка» («Print»). Зберегти документ на жорсткий диск можна за допомогою команди «Зберегти» («Save» меню «File»). Для пошуку фрагмента тексту в сторінці, що цікавить, використовують команду «Знайти» («Find...» меню «Edit»). Побачити вихідний текст сторінки, що проглядається, мовою HTML можна за допомогою команди «Код HTML» («Page Source» меню «View»).

Після завантаження веб-сторінки браузер, тимчасово вміщає її елементи (картинки, анімації, звукові файли) у спеціальний буфер у пам'яті комп'ютера. Завдяки цьому, при повторному звертанні, вона завантажується дуже швидко. На окремому комп'ютері область в оперативній пам'яті й на жорсткому диску комп'ютера, використовувана для зберігання переглянутих веб-сторінок, називається «кеш» (англ. cache). У локальній мережі організації організують загальний буфер, у який попадають веб-сторінки, що проглядаються всіма комп'ютерами в локальній мережі. Такий загальний буфер управляється спеціальною програмою. Її називають « прокси-сервер» (англ. proxy-server). Прокси-сервер забезпечує високу швидкість завантаження популярних сторінок і скорочує обсяг інформації, одержуваної з Інтернет у локальну мережу. Це знижує вартість підключення організації до Мережі. Сучасні прокси-сервери додатково обмінюються один з одним інформацією, за рахунок цього ефективність їх використання суттєво зростає.

Зручним інструментів браузерів є інструмент закладок (за аналогією із закладками, що відзначають цікаві місця книги). Після появи нової закладки в списку закладок можна завантажити сторінку, на яку закладка вказує, без уведення адреси вручну. Список закладок можна редагувати й організовувати в папки.

Для обміну інформацією між користувачем і веб-сервером використовуються HTML-форми. Користувач заповнює необхідні поля форми й відсилає її на сервер за допомогою кнопки «Відправити» («Submit»).

Браузер не тільки забезпечує доступ до інформаційних ресурсів Всесвітньої Павутини. Він також працює з іншими службами Мережі, такими як: FTP, Gopher, WAIS. FTP (File Transfer Protocol - протокол передачі файлів) - служба, призначена для передачі файлів. Разом з браузером на комп'ютер звичайно встановлюють програма для користування електронною поштою і читання новин (новини - спеціальний сервіс Мережі).

Браузер є основною програмою для доступу до служб Мережі. З його допомогою можна одержати доступ практично до будь-якої служби Інтернет, навіть якщо браузер не підтримує роботу із цією службою. Для цього використовуються спеціальним образом запрограмовані веб-сервери, які зв'язують Всесвітню Павутину з даною службою Мережі. Приклад такого роду веб-серверів - численні безкоштовні поштові сервери з веб-інтерфейсом (приклад - http://www.online.ua/). У браузерах передбачений механізм для підключення додаткових можливостей незалежними виробниками, які називають plug-in.

Браузери працюють на комп'ютерах під керуванням різних операційних систем. Це говорить про незалежність Всесвітньої Павутини від типу застосовуваного користувачем комп'ютера й операційної системи.

Крім браузерів з графічним інтерфейсом існує спеціальний браузер, що працює в текстовому режимі. Він називається Lynx (http://www.lynx.org/).

Браузер стрімко перетворюється в універсальну багатоцільову програму для роботи в Мережі. Sun Microsystems, одна із провідних корпорацій в області розробки програмного забезпечення, бачить сучасний комп'ютер як органічну частину Мережі. «Мережа - це комп'ютер» - таке кредо Sun. Логічним підсумком такого підходу може стати використання на комп'ютері однієї програми - браузера.

Sun протягом от уже декількох років пропонує великим компаніям мережні комп'ютери, які реалізують цю концепцію. Новітні стільникові телефони, що надають своїм власникам вихід в Інтернет, також оснащені всього однією програмою - спрощеною версією браузера.

РОЗДІЛ 3. АЛГОРИТМИ

3.1. Визначення алгоритму

Алгоритм - це кінцева послідовність точно певних операцій, виконання кожної з яких вимагає кінцевий обсяг оперативної пам'яті й кінцевий час. Алгоритм повинен працювати кінцевий час для будь-яких вхідних даних.

> *Якщо відволіктися від предмета дисципліни й перекласти визначення алгоритму на діяльність людини, то наша поведінка визначається набором правил. Дії, вчинені свідомо або несвідомо, мають певний порядок (послідовність) і виконуються залежно від яких-небудь умов (подій). Примітні послідовності дій запам'ятовуються, а потім повторюються знову й знову.*
>
> *Настає етап, коли ми продумуємо послідовність операцій і дій для досягнення якої-небудь мети, оцінюємо можливі результати й вибираємо послідовність дій кращу на наш погляд.*
>
> *Це наштовхує на думку – розвиток як алгоритмічного, так і математичного мислення надто важливо для фахівців у будь-якій сфері діяльності.*
>
> *Програмування є справжнім полігоном розвитку алгоритмічного мислення. Тут є всі складові – формальна мова опису поведінки й пристрій, що «беззаперечно виконує» задану послідовність дій – обчислювальна система.*
>
> *Ми можемо моделювати поведінку й аналізувати можливі наслідки.*

Ми не претендуємо на вичерпний виклад теорії алгоритмів, але прагнемо відзначити основні ідеї, які закладені у фундамент ефективного використання як мови програмування при створенні власних програм, так і програмного забезпечення, з яким Ви будете працювати надалі.

3.2. Способи запису алгоритмів

Розглянемо просте завдання

> *Дано два цілі позитивні числа M і N, потрібно знайти їхній найбільший загальний дільник. (Визначення: найбільший загальний дільник – позитивне ціле число, яке ділить числа M і N націло.)*

Розв'язання цього арифметичного завдання відоме нам із предмета «Арифметика», а запропоноване Евклідом близько 300 р. до н.е.

> ***Алгоритм Евкліда: Вибрати більше із чисел і розділити його на менше. Якщо залишок від розподілу рівний 0, то відповідь знайдена - це менше число. Якщо залишок не рівний 0, виконувати розподіл дільника на залишок, поки не одержимо залишок 0.***

Викладений алгоритм *у такому вигляді* має право на життя, але спробуйте запропонувати його автомату?

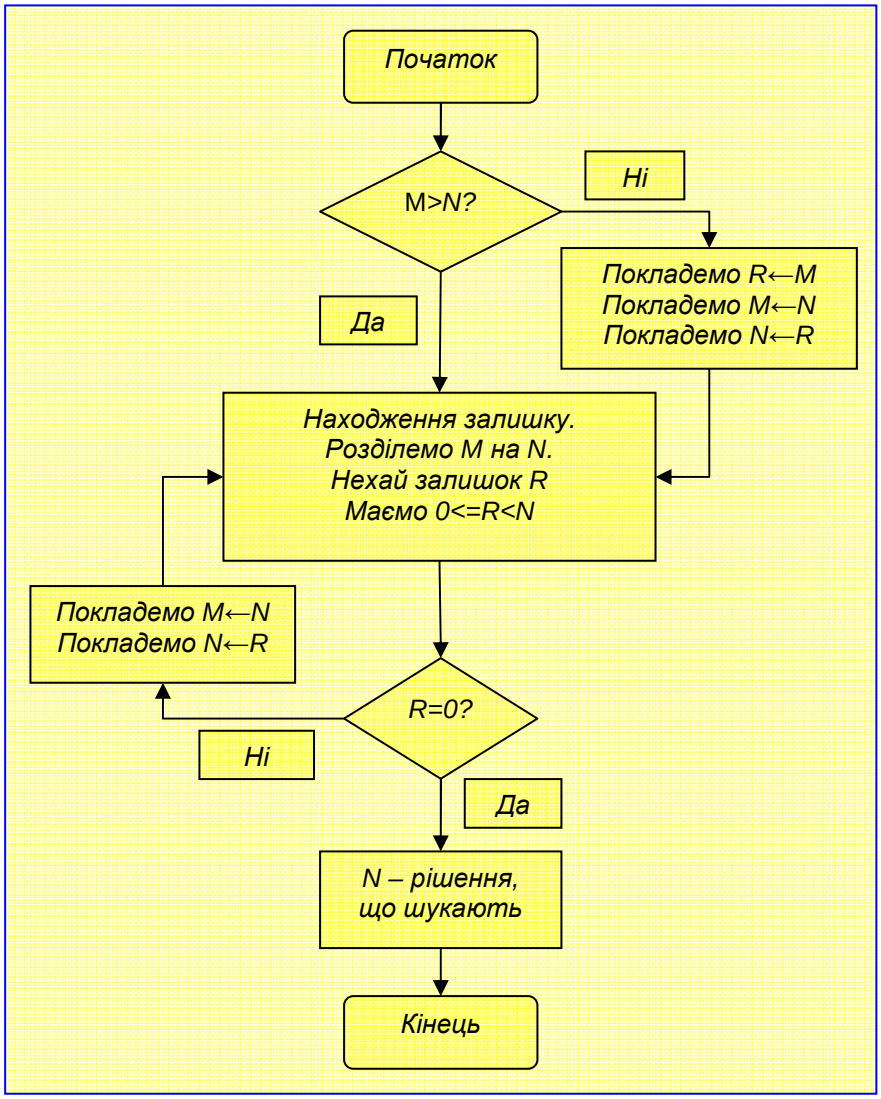
Для опису алгоритмів, як правило, використовують блок-схеми й діаграми. Такий опис алгоритмів дає уявлення про послідовність кроків, з їх коротким описом. Сучасні засоби створення програмного забезпечення дають набір інструментів візуального проектування структури алгоритмів і програмних систем і комплексів, що їх реалізують.

Сучасна парадигма програмування *об'єктно-орієнтоване програмування* (ООП) дозволяє глибше усвідомлювати предметну область, для якої розробляють програмне забезпечення, вводить абстракції і відношення між ними. Наближає групу розробників до розуміння внутрішніх і зовнішніх зв'язків, які обслуговують інформаційні потоки. Однак ця парадигма не звільняє розробників від аналізу видів операцій, їх послідовності й умов виконання. А це і є побудова алгоритмів.

Блок-схеми містять кілька основних елементів:

блок виконання дій (прямокутник – один вхід, один вихід);
блок аналізу умов (ромб – один вхід, два виходи);
напрямок переходу (стрілка – з початку в кінець).

3.2.1. Блок-схема алгоритму Евкліда



3.3. Основні етапи побудови алгоритму

3.3.1. Постановка завдання

Для розуміння завдання його слід точно сформулювати. Ця умова не є достатньою, але абсолютно необхідна. Для точного формулювання завдання слід відповісти на низку питань. Ось деякі з них:

Чи зрозуміла термінологія, використовувана в попередньому формулюванні?
Що дано?
Що потрібно знайти?
Як визначити рішення?
Яких даних не вистачає?
Чи є, якісь дані непотрібними?
Які зроблені допущення?

Перелік питань може бути розширений залежно від розв'язуваного завдання.

3.3.2. Побудова моделі

Побудова або вибір моделі - найбільш важливий крок у процесі розв'язання. Це крок суттєво впливає на інші етапи розв'язання. *(«Який фундамент такий і будинок»)*.

Вибір моделі - більшою мірою мистецтво, ніж наука, імовірно ця тенденція збережеться. Вивчення вдалих моделей – це найкращий спосіб набути досвід у моделюванні.

Починати розробку моделі слід з відповідей принаймні на два питання:

Чи існують розв'язання аналогічних завдань?

Які математичні (формальні) структури найбільше підходять для опису завдання?

Перше питання, можливо, саме корисне не тільки в науці, але й у повсякденному житті. Погодьтеся, ми часто керуємося як накопиченим власним досвідом, так і досвідом навколишніх.

3.3.3. Правильність алгоритму

Доведення правильності алгоритму – це один з найбільш важких, а іноді й особливо стомлюючих етапів створення алгоритму. Імовірно, найпоширеніша процедура доказу правильності алгоритму - це виконання (або *прогін*) програми, яка його реалізує, на різних тестах. Якщо видані програмою відповіді можуть бути підтверджені відомими або обчисленими вручну даними, виникає спокуса зробити висновок, що програма «працює», а алгоритм правильній. Але цей метод не виключає всі сумніви; може існувати випадок, в якому програма не спрацює.

3.3.4. Реалізація алгоритму

Наступний крок – реалізація алгоритму, тобто написання програми для обчислювальної машини. Іноді, особливо в починаючих програмістів, складається неправильне уявлення про те, що саме реалізація алгоритму і є програмуванням (насправді це кодування алгоритму у вигляді конструкцій мови програмування).

Програмування - це все-таки:

розробка й аналіз алгоритму розв'язання завдання;
проектування і розробка типів даних, які найбільш адекватно описують оброблювану інформацію;
вибір ефективних засобів мови програмування для реалізації алгоритму;

проектування й розробка типів даних, які найбільше адекватно описують оброблювану інформацію.

Труднощі, що чекають на етапі реалізації:

по-перше, часто трапляється, що окремо взятий крок алгоритму виражений у формі, яку важко перевести безпосередньо в конструкції мови програмування. Наприклад, один з кроків алгоритму може бути записаний у вигляді, що вимагає набору функцій для його реалізації;

по-друге, перед тим, як ми зможемо почати писати програму, ми повинні побудувати цілу систему структур даних для представлення важливих аспектів використовуваної моделі.

Як правило, на цьому етапі роботи слід відповісти на наступні запитання:

Які основні змінні, об'єкти, зв'язки, відношення?
Якого вони типу?
Скільки потрібно масивів і яка їхня розмірність?
Чи має сенс користуватися зв'язними списками?
Які потрібні функції (можливо, наявні власні або бібліотечні)?
Якою мовою програмування користуватися?

3.3.5. Аналіз алгоритму і його складнощі

Практична важливість цього етапу очевидна??? Ну, хоча б бажання одержати оцінки для обсягу необхідної пам'яті й часу роботи, який буде потрібний алгоритму для успішної обробки конкретних даних. У процесі аналізу алгоритму важливо встановити абсолютний критерій, коли можна вважати розв'язок завдання оптимальним. Тобто показати, що алгоритм настільки гарний, що неможливо (незалежно від наших розумових здібностей) значно поліпшити його.

3.3.6. Перевірка програми

Рідко можна гарантувати правильність програми, але все-таки слід провести досить повну перевірку випадків, які з більшою ймовірністю зустрінуться у практиці. Процес перевірки може бути тривалим, стомлюючим і складним і включає такі етапи:

підготовка варіантів вхідних даних, для яких можна встановити правильність висновків (результатів),
проведення прогонів програми,
аналіз їх результатів,
підготовка ручних розрахунків для перевірки і т.д.

Багато алгоритмів важко, а іноді неможливо досліджувати математично. У таких випадках особливо важливо перевірити алгоритми в дії, це єдина можливість оцінити їхню якість.

РОЗДІЛ 4. МОВА ГІПЕРТЕКСТОВОЇ РОЗМІТКИ ДОКУМЕНТУ – МОВА HTML

4.1. Основні поняття мови HTML. Теги, атрибути тегів, сутності

Опис мови HTML публікує спеціальна організація «World Wide Web Consortium (W3C)». Поточною версією мови є версія 4 (документ, який її описує, можна знайти за адресою http://www.w3.org/TR/REC-html40).

HTML є *мовою розмітки документа* (його назва - скорочення від Hypertext Markup Language - мова розмітки гіпертексту). Він визначає, *як документ повинен виглядати при його відображенні відповідним засобом* (у цьому випадку Web-браузером). Для оформлення елементів документа використовують *теги* (tags).

Тег - спеціальна послідовність символів, яка задає елементи для відображення, керування відображенням тексту або структурою документа.

Якщо не брати теги, документ у форматі HTML є звичайним *текстовим документом*.

Розрізняють два види тегів:

одиночні, які задають елементи для відображення. Такі теги цілком полягають усередину пари символів < > і зустрічаються в документі поодинці. Прикладом такого тега може служити тег <HR>, який задає горизонтальну лінія-роздільник.

парні, які керують відображенням тексту або визначають структуру документа. Такі теги складаються з двох частин: *відкриваючого тега*, укладеного всередину пари символів < > і *закриваючого тега*, укладеного всередину </ >. Між відкриваючим і закриваючим тегами може перебувати текст, інші теги (тобто допускається вкладеність тегів) і т.д. - усе, що там перебуває, зазнає форматуванню або іншому впливу - у відповідності зі специфікою тега. Найпростішим прикладом може служити пари тегів <H1> ... </H1>, що задає форматування всього, що перебуває між ними, як заголовка першого рівня. Існують такі парні теги, які служать контейнерами для інших тегів і без них не мають змісту, наприклад, тег <TABLE> ... </TABLE>, що задає Html-Таблицю, не має змісту без вкладених тегів, таких, як <TR> ... </TR> (рядок таблиці), <TD> ... </TD> (гніздо таблиці) і т.д.

Теги можуть мати *атрибути*, які задають додаткові характеристики. Загальний синтаксис запису тега:

```
<tag attribute1="value1" attribute2="value2" ...>
```

Приклад тега з атрибутами - тег <TABLE>, атрибутом якого є товщина лінії навколо гнізд – атрибут *border*:

```
<TABLE BORDER="1"> ... </TABLE>
```

Атрибути можна задавати не тільки для парних, але й для одиночних тегів (наприклад, тег <HR WIDTH="90%"> задає смугу-роздільник шириною в 0.9 поточної ширини вікна браузера).

Для відображення зарезервованих символів запропонована концепція Html-*підстановок* (HTML entities) - спеціальних послідовностей, що відображаються як спеціальні символи. Вони починаються із символу &, а закінчуються символом «крапка з комою». Ось таблиця відповідності символів і сутностей:

HTML-підстановки

Символ	Сутність
<	<
>	>
подвійні лапки	"
безпосередньо амперсанд	&
нерозривний пробіл (nonbreaking space)	
знак копірайту ©	©

Сутності можуть задавати й довільні коди. Для цього використовується такий запис: &#*код*;. *Код* задає десятковий ASCII-код символу.

4.2. Базова структура Html-документа

Будь-який Html-Документ має стандартну структуру:

1. документ починається зі спеціального тега <!DOCTYPE>, який задає версію мови HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
```

HTML- документ перебуває всередині парного тега <HTML> ... </HTML>.

документ має *заголовок.* Заголовок оформляється парним тегом <HEAD> ... </HEAD>. Інформація заголовка частково відображається у вікні браузера. Заголовок містить інформацію, яка визначає деякі характеристики документа:

- текст, відображуваний у заголовку вікна браузера (тег <TITLE> ... </TITLE>),
- додаткова інформація для браузера або інших додатків (тег <META>);

документ має *тіло.* Тіло документа оформляється парним тегом <BODY> ... </BODY>. Інформація тіла й відображається у вікні браузера.

Так виглядає структура (кістяк) Html-документа:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
        <HEAD>
                заголовок HTML - документа
                заголовок -
        </HEAD>
        <BODY>
                тіло HTML - документа
                тіло – це інформація, яка відображається у вікні браузера
        </BODY>
</HTML>
```

4.3. Основні теги HTML

Розглянемо загальні поняття для різних тегів.

4.3.1. Задання кольорів у HTML

Багато атрибутів приймають у якості значення колір (колір символів, колір фону і т.і.). Колір може бути заданий декількома способами (на практиці, для задання кольору Ви, скоріше за все, будете користуватися засобами редактора HTML):

1. чисельне задання у форматі RGB: #RRGGBB де RR, GG і BB - шістнадцятирічні цифри, що задають колірні складові. Багато продуктів генерують такий рядок при завданні кольору. Ось приклади: COLOR="#00FF00" - зелений колір, COLOR="#000000" - чорний, COLOR="#0000FF" - синій і т.і.

задання імені кольору. Багато кольорів мають визначені імена: white, red, green і т.д. Задаються так: COLOR="red" і т.і.

4.3.2. Задання розмірів у HTML

Розміри в HTML підрозділяються на *абсолютні* й *відносні*. Перші вказуються у фізичних одиницях ("ширина 200 пікселів"), другі - у відсотках.

За замовчуванням усі абсолютні розміри в HTML задаються в пікселах (крапках на екрані), тобто WIDTH="100" має на увазі 100 пікселів (це можна задати і явно, вказавши "100px"). Можна задавати ще інші одиниці виміру, наприклад WIDTH="2in" (дюйми), "2cm" (сантиметри), "2mm" (міліметри). Припустимі й специфічні видавничі одиниці, такі, як "точки" (2pt, для екрана збігаються з пікселями) та інші.

Відносні розміри відлічуються від об'ємного "батьківського" тега (наприклад, для гнізда таблиці WIDTH="90%" означає 90% ширини всієї таблиці) або, якщо такого немає - від поточної ширини вікна браузера. Елементи з відносними розмірами міняють розмір при зміні вікна браузера, з абсолютними - ні.

4.3.3. Задання URL

URL (Universal Resource Locator) ідентифікує документ в Internet або на локальному комп'ютері. URL використовують як можливі значення атрибутів таких тегів, як гіперпосилання <A> і інші. Розрізняють відносні й абсолютні (повні й неповні) URL, перші можуть посилатися тільки на документ на тій же машині в тому ж дереві каталогів, другі можуть задавати документ де завгодно (повні) і де завгодно на тій же машині (неповні).

Приклади відносних URL (зазначимо, що підкаталоги можуть розділятися як /, так і \, але для сумісності рекомендують використовувати /):

"logo.gif" (документ у тому ж каталозі),
"pic/logo.gif" (документ у підкаталозі поточного),
"../pic/logo.gif" (документ у підкаталозі каталогу-предка поточного).

Приклад абсолютного неповного URL (усі такі URL починаються з /):

"/pic/logo.gif" (документ у каталозі pic кореневого каталогу документів даного сервера, більш докладно про структуру каталогів сервера дізнаємося пізніше)

Приклади повних абсолютних URL (тут задають протокол, ім'я машини, може бути заданий порт, ім'я користувача, пароль і т.д):

"http://my_company.com/pic/logo.gif",
"http:// my_company.com:8000/pic/logo.gif" (заданий порт),
"ftp://user:password@my_company.com/pub/pic/logo.gif" (заданий протокол FTP, ім'я користувача й пароль).

URL можуть включати дані для серверної програми, що викликається при запиті по цьому URL. Такі дані звичайно відділяють від іншої інформації символом "?".

4.3.4. Теги структури документа

До цієї групи відносять три теги - <HTML>, <HEAD> і <BODY>. Розглянемо тег «тіла документа» - <BODY>.

Атрибути тега <BODY>

№	Ім'я атрибута	Функціональне призначення	Приклад використання	Змістовний опис прикладу
1	TEXT	Установлює колір символу	<BODY TEXT="#00FFFF">	Символи тексту зображувати кольором #00FFFF
2	BGCOLOR	Установлює колір фону	<BODY BGCOLOR="lightgray">	Задати колір тла сторінки – ясно-сірий.
3	LINK	Установлює колір для всіх гіперпосилань, які не відвідували з даної сторінки	<BODY LINK="black" >	Зображувати гіперпосилання, які не відвідували з даної сторінки, чорним кольором.
4	VLINK	Установлює колір для всіх гіперпосилань, які відвідували з даної сторінки	<BODY VLINK=”red”>	Зображувати гіперпосилання, які відвідували з даної сторінки червоним кольором.
5	ALINK	Установлює колір для всіх активних гіперпосилань	<BODY ALINK=”blue”>	Зображувати активні гіперпосилання синім кольором.
6	TOPMARGIN	Установлює ширину	<BODY TOPMARGIN="30px">	Установити відступ зверху

		відступу зверху		змісту документа від границі вікна в 30 пікселів
7	BOTTOMMARGIN	Установлює ширину нижнього відступу	<BODY BOTTOMMARGIN="25px">	Установити відступ знизу змісту документа від границі вікна в 25 пікселів
8	LEFTMARGIN	Установлює ширину відступу ліворуч	<BODY LEFTMARGIN="40px">	Установити відступ ліворуч змісту документа від границі вікна в 40 пікселів
9	RIGHTMARGIN	Установлює ширину відступу праворуч	<BODY RIGHTMARGIN="35px">	Установити відступ праворуч змісту документа від границі вікна в 35 пікселів
10	BACKGROUND	Указує ім'я файла з фоновим зображенням	<BODY BACKGROUND="img/bg.gif">>	Установити фонове зображення документа з файла *bg.gif* папки *img*
11	BGPROPERTIES	Установлює поведінку фонового зображення	<BODY BGPROPERTIES="fixed">>	фонове зображення не прокручувати
12	NOWRAP	Установлює скасування переносу слів	<BODY NOWRAP="true">> <BODY NOWRAP="false">>	перенос не робиться перенос виконується
13	SCROLL	Установлює	<BODY SCROLL="no">>	смуги не

		режим зображення смуг прокручування	<BODY SCROLL="yes">	відображаються смуги відображаються

4.3.5. Теги заголовка

Тег <TITLE>..</TITLE> використовують для задання інформації, яка відображується в заголовку вікна браузера. Вона багатьма додатками сприймається як заголовок документа.

Задавайте значення тега <TITLE> завжди і робіть це осмислено.

Ще одним важливим тегом, який має сенс тільки всередині тега <HEAD> є тег <META>. Він передає сховану інформацію (метаінформацію) про документ серверу й клієнтові. Для нього припустимо три атрибути:

1. HTTP-EQUIV - задає інформацію, яку Web-сервер повинен включити в заголовок Http-відповіді. Значень, які приймає атрибут:

"refresh" (що змусить браузер перечитувати документ час від часу), у цьому випадку припустиме CONTENT="число-секунд" (час між перечитуваннями) або CONTENT="url" (документ, який буде перечитаний)

"Content-Type" (звичайно для задання кодування документа), у цьому випадку атрибут CONTENT виглядає, наприклад, так: CONTENT="text/html; charset=utf-8", при цьому до крапки з коми йде MIME-тип документа, після - кодування.

Тег <META> з атрибутом HTTP-EQUIV

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
```

NAME - ідентифікує інформацію, задану в атрибуті CONTENT. Може приймати такі значення:

"Description" - значення атрибута CONTENT описує документ. Може використовуватися пошуковими системами, які зберігають це значення щоб видавати його користувачеві як короткий опис знайденого документа.

"Generator" - значення атрибута CONTENT задає ім'я додатка, який створив документ.

"Keywords" - значення атрибута CONTENT складається зі списку ключових слів, що описують документ. Слова в списку розділяються комами. Використовується пошуковими системами для організації пошуку по ключовім слову.

"Robots" - значення атрибута CONTENT задає, чи повинен документ індексуватися пошуковими системами, які зорієнтовані на тег META. Якщо CONTENT="all" індексація припустима, CONTENT="noindex" – не припустима.

Теги <META> з описом документа

```
<META NAME="Description" CONTENT="Web Lecture # 1">
<META NAME="Keywords" CONTENT="web,html,tags,internet">
```

CONTENT - задає метаінформацію відповідно до значень атрибутів HTTP-EQUIV або NAME. Можливі значення атрибута зазначені вище.

Якщо потрібно задати декілька характеристик, в заголовок включають кілька тегів <META> - по одному на кожну характеристику.

4.3.6. Теги базового форматування документа

Розрізняють форматування *логічне*, коли задаються принципові характеристики елементів тексту, такі як "виділення", "заголовок першого рівня" і т.і., і *фізичне*, коли задають точні вимоги до відображення, такі як "курсив", "напівжирний шрифт", "колір" і т.і.

> Користуйтеся логічним форматуванням, де це можливо, тому що можуть існувати браузери, для яких деякі характеристики не мають змісту.

Логічне форматування на рівні абзацу

За винятком <HR>,
 і <WBR> усі теги тут є парними.

<P> ... </P> - задає логічний абзац (параграф) тексту. Закриваючий тег опускати не рекомендують. Без нього браузер часто сприймає тег <P> як команду «пропустити один рядок», що не коректно. Для тега припустимий атрибут ALIGN (вирівнювання – left (зліва), right (з права) ,center (по центру), justify (розподілений між границями)).

<H1>-<H6> - задає заголовки різного рівня. Браузер відображає напівжирним шрифтом різного розміру й виділенням в окремий рядок.

<PRE> - задає відображення ділянки документа зі збереженням вихідного форматування (пропуск, переведення рядка і т.і.)

Логічне форматування всередині абзацу

<EM> - виділення (emphasizing) тексту, реалізується *курсивом*.

<STRONG> - логічний акцент на тексті, реалізується **напівжирним** шрифтом.

<CODE> <KBD> <TT> - теги звичайно реалізуються однаково (шрифтом фіксованої ширини), але логічно різняться (наприклад <CODE> призначений для форматування програмного коду)

<BLOCKQUOTE> - задає відображення тексту з відступом від лівого краю й пропуском місця до й послу цього тексту

<HR> - задає горизонтальний роздільник. Для нього припустимі такі атрибути, як WIDTH (ширина), ALIGN (вирівнювання), NOSHADE (якщо є, то лінія не буде опуклою), COLOR.

Фізичне форматування

<I> - задає виділення *курсивом* <B> - задає **напівжирний** шрифт <U> - задає підкреслення

<CENTER> - центрує текст

<FONT> - задає різні характеристики шрифту.

Використання тега <FONT> не вітається. Низькорівневе форматування зручніше реалізувати за допомогою таблиць стилів.

Припустимі атрибути:

1. COLOR - задає колір шрифту

FACE - задає сімейство шрифту. Якщо не знайдений шрифт зазначеного сімейства, то система намагається знайти інший за списком і т.д. Приклад: <FONT FACE="Arial,Helvetica,Sans-Serif>

SIZE - задає розмір шрифту (допустимо абсолютний розмір, наприклад, 12pt, відносний (1-7), відносний до основного шрифту (+1, +2, -1 і т.д.) і заданий словами (xx-small, x-small, medium і т.д.)

 - задає переведення рядка. Це одиночний тег.

<NOBR> - задає, що текст всередині тега не можна переносити на новий рядок (якщо знадобиться такий перенос, використовують одиночний тег <WBR>)

<SUB> <SUP> - відображення тексту як $_{\text{підрядкового}}$ й ^{надрядкового} індексу

4.3.7. Теги для оформлення списків у документі

Нумеровані списки

Такі списки задаються двома тегами <OL> ... </OL>, між якими знаходиться набір елементів списку.

Елемент списку задається так: <LI>текст елемента</LI> (на практиці можна опускати закриваючий тег). Елементи в такому списку автоматично одержують номери. Можна явно задавати номера для елементів так: <LI VALUE="10">

Оформлення нумерованого списку:

```
<OL>
<LI>Елемент 1</LI>
<LI>Елемент 2<BR>ще текст</LI>
```

```
<LI VALUE=10>Елемент 10</LI>
</OL>
```

А ось як він відображається:

1. Елемент 1
2. Елемент 2
   ще текст

10. Елемент 10

Для «нумерації» можна використовувати інші позначення. Такі можливості надає атрибут TYPE.

Використання атрибута TYPE у нумерованих списках

Тег з атрибутом	Результат
<OL TYPE=a>	нумерація малими літерами
<OL TYPE=A>	нумерація прописними літерами
<OL TYPE=I>	нумерація римськими літерами

Ненумеровані списки

Елементи списків супроводжуються маркерами.

Списки задаються парою елементів `<UL> ... </UL>`, елементи задаються як для нумерованого списку. Можна задавати формат елемента так: `<UL TYPE="disc">` - суцільне коло (так за замовчуванням); є й інші значення для TYPE: `circle` - порожнє коло, `square` - квадрат.

Оформлення ненумерованого списку

```
<UL TYPE="circle">
<LI>Елемент 1</LI>
<LI>Елемент 2<BR>ще текст</LI>
<LI>Елемент 3</LI>
</UL>
```

А ось як він відображається:

- Елемент 1
- Елемент 2
  ще текст

- Елемент 3

Формат маркера елемента, задає атрибут TYPE.

Використання атрибута TYPE у ненумерованих списках

Тег з атрибутом	Результат
<UL TYPE=”disc”>	маркер – «суцільне коло»
<UL TYPE=”circle”>	маркер – «порожнє коло»
<OL TYPE=”square”>	маркер – «квадрат»

Списки визначень

Такий список знаходиться між тегами <DL> ... </DL>. Елементи списку складається з двох частин:

1. заголовок визначення, який знаходиться між тегами <DT>...</DT>;

текст визначення, який знаходиться між тегами <DD> ... </DD>.

Оформлення списку визначень.

```
<DL>
<DT>Заголовок 1</DT><DD>Текст визначення 1<br>Ще текст</DD>
<DT>Заголовок 2</DT><DD>Текст визначення 2<br>Ще текст</DD>
</DL>
```

А от як він відображається:

Заголовок 1

Текст визначення 1

Ще текст

Заголовок 2

Текст визначення 2

Ще текст

Списки можна вкладати один в один.

4.3.8. Гіперпосилання

Гіперпосилання - основа для Web-навігації, користувач вибирає гіперпосилання і переходить на зв'язаний з ним документ. Вони можуть указувати на

довільний URL (пізніше побачимо, як з їхньою допомогою можна передавати дані додаткам). От які теги реалізують гіперпосилання:

Тег <A> - anchor

Тег <A> . . . </A> задає гіперпосилання або місце в документі, на яке може посилатися гіперпосилання. Основний атрибут HREF задає, на який URL вказує гіперпосилання. Текст між тегами задає текст, який буде бачити користувач (цей текст звичайно якось виділений, за замовчуванням підкресленням і синім кольором).

Оформлення гіперпосилань

```
<A HREF="jac_eom/ot_am.html">Початок</A>
<A HREF="http://www.my_company.com/ jac_eom/ot_am.html">Повернутися до початку</A>
<A HREF="mailto:i.jac@ksame.kharkov.ua">Відправити почту</A>
```

Останній приклад демонструє спеціальний вид URL – так званий mailto URL.

Вибір користувачем такого посилання веде до запуску програми «Поштовий клієнт» в режимі формуваня електронного листа з полем To (Кому), яке заповнене відповідно до атрибута HREF.

Додаткові атрибути тега <A> такі:

NAME - задає ім'я адреси всередині документа для посилання. <A NAME="gl_3">. Такий атрибут задає точку всередині документа (анкер, anchor), на яку можна посилатися так

```
<A HREF="http://www. my_company.com/jac_eom/ot_am.html#gl_3">Розділ 3</A>
```

TARGET - звичайно використовують із фреймами, задаючи, в якому фреймі повинен відкритися документ, на який робиться посилання.

Тег <BASE>

Тег <BASE> задає URL, щодо якого будуть задаватися відносні URL у документі. Повинен перебувати в розділі заголовка.

Використання тега <BASE>

якщо задане

```
<HEAD><BASE HREF="http://www.my_company.com/jac_eom/"></HEAD>
```

те посилання виду

```
<A HREF=" ot_am.html">Page</A>
```

насправді буде посилатися на URL - http://www. my_company.com/jac_eom/ot_am.html

Тег <LINK>

Тег <LINK> - встановлює зв'язок між документом і додатковими даними. Атрибути:

REL - задає відношення між поточним документом і тим документом, на який указує посилання;
REV - те ж, але навпаки (відношення з боку документа, на який указує посилання).

Організація гіперпосилань за допомогою атрибута ідентифікації ID

Кожний тег може мати атрибут ідентифікації ID. За форматом він дуже схожий з атрибутом NAME. Якщо ми задамо для тега атрибут ID="*ім'я*", то він буде працювати так, ніби ми задали в цьому місці анкер.

Використання атрибута ID

якщо в тексті документа page.html заголовок заданий так:

```
<H1 ID="gl_3"> Розділ 3</H1>
```

на нього можна посилатися, так само, як посилаємося на анкер:

```
<A HREF="http://www.my_company.com/jac_eom/ot_am.html#gl_3"> Розділ 2</A>
```

4.3.9. Робота із зображеннями

Розміщення зображень в документі HTML виконують у формі посилань на файли зображень - графічні файли (GIF, JPEG або PNG). Для такого посилання використовують тег <IMG>. Синтаксис тега:

```
<IMG SRC="url">
```

URL файла зображення може бути як відносним, так і абсолютним. Тобто можна відображати на своїх сторінках файли з інших сайтів.

> , Не слід робити абсолютні посилання без попиту. Якщо виникає проблема з доступом до вилученого сайту завантаження сторінки з посиланням на нього теж піде із проблемами.

Посилання на зображення

```
<IMG SRC="pic/logo.gif">
```

Атрибути тега IMG

1. WIDTH - ширина зображення (звичайно в пікселах, якщо інше не задане);
2. HEIGHT - висота зображення.

Атрибути WIDTH і HEIGHT дуже важливі, вони дає браузеру можливість відображати текст на сторінці коректно, не чекаючи завантаження всіх картинок (на місці картинок виводяться рамки відповідного розміру)

Встановлюйте значення атрибутів WIDTH і HEIGHT тега IMG завжди, це прискорить завантаження Web-сторінок у користувача.

Технологія графічних розпорок.

Відзначимо, що коли реальні розміри зображення не збігаються з розмірами, заданими тегами HEIGHT і WIDTH, браузер виконає масштабування зображення до розмірів, зазначених в атрибутами. Це основа технології графічних розпорок. Вона дозволяє управляти відносним розташуванням елементів на сторінці, вміщуючи між ними теги <IMG>. Вони посилаються на прозоре зображення розміром в один піксел. При цьому HEIGHT і WIDTH задають того розміру, який потрібний для розпорки. У прозорому зображенні встановлюють так званий біт прозорості для одного кольору в зображенні. Тоді, що задане таким кольором, завжди буде приймати колір фона.

Задавати біт прозорості можна інструментами будь-якого графічного редактора.

От приклад такої розпірки: звідси до сюди
<img border=0 src="pic/pixels.gif" height=1 width=50>.

ALIGN - задає, як зображення розташовується щодо навколишніх елементів сторінки.

Значення атрибуту ALIGN тега IMG

Значення	Результат
absbottom	вирівнюється нижня лінія картинки по лінії самої нижньої точки тексту
absmiddle	вирівнюється середина об'єкта із серединою навколишнього тексту. Практично єдиний коректний спосіб вирівняти текст і картинку по вертикалі
baseline	вирівнюється нижня частина об'єкта з базовою лінією навколишнього тексту
bottom	вирівнюється нижня частина об'єкта з нижньою лінією навколишнього тексту (не самої нижньої, а розрахованої із середнього "хвоста" букви)
left	вирівнює картинку по лівій стороні навколишнього тексту, наступний текст обтікає картинку праворуч. Картинка ліворуч від тексту використовує align="left"
middle	вирівнює середину об'єкта по навколишньому тексту (виходить нерівно, використовують рідко)

right	вирівнює картинку по правій стороні навколишнього тексту, наступний текст обтікає картинку ліворуч. Картинка праворуч від тексту використовує align="right"
texttop	вирівнюється верх картинки по лінії самої верхньої точки тексту
top	вирівнюється верх картинки по верхній лінії тексту (розрахованої із середньої висоти букви).

На практиці найчастіше використовують значення absmiddle, left і right.

ALT - задає текстовий опис зображення. Опис зображення з'являється на місці зображення, якщо його пересилання зазнало невдачі або користувач відключив завантаження картинок. Крім того, опис з'явиться як спливаючий напис, якщо підвести мишу до зображення.

BORDER - ширина рамки навколо зображення.

HSPACE - горизонтальний відступ від зображення до тексту або інших елементів

VSPACE - вертикальний відступ від зображення до тексту або інших елементів

LOWSRC - задає альтернативне зображення (звичайно з низьким дозволом), яке завантажується паралельно з основним зображенням (заданим атрибутом SRC), відображується раніше його й залишається видимим, поки основне зображення не завантажиться повністю.

Зображення як гіперпосилання

Доволі цікаво, коли зображення використовують на Web-сторінці як гіперпосилання. Таку конструкцію створити нескладно. Потрібно вкласти тег <IMG> всередину тега <A>. Гіперпосилання у формі зображення

```
<A HREF="page.html"><IMG SRC="page.gif" BORDER=0></A>
```

Задавайте атрибут BORDER=0, інакше навколо зображення-гіперпосилання буде синя рамка.

Коментарі

Коментар в HTML має вигляд:

```
<!-- це коментар - браузер його не відобразить -->
```

Усі 4 символи відкриваючого тега й 3 - закриваючого є необхідними. Коментар сховає інформацію від користувача, який одержав Web-сторінку. Користувач завжди може подивитися HTML-код Web-сторінки за допомогою функціональних можливостей браузера.

4.3.10. Оформлення інформації у вигляді таблиці

Таблиці в HTML можуть використовуватися, принаймні, для двох цілей:

1. безпосередньо для відображення табличної інформації;

для візуального форматування сторінки.

Для оформлення таблиць використовують наступний набір парних тегів:

1. тег <TABLE>, у який вкладені всі інші - він обмежує код таблиці;

тег <TR>, що задає рядок таблиці. У теги <TR> вкладають теги для гнізд таблиці, що утворюють рядок;

теги <TH> і <TD>, гнізда що задають таблиці (гніздо-заголовок і звичайне гніздо). Інформація, яка відображається в таблиці, повинна перебувати всередині тегів <TH> і <TD>.

Найпростіша таблиця

HTML код	Вигляд у браузері
<TABLE BORDER=1> <TR><TH>Заголовок1</TH><TH>Заголовок2</TH></TR> <TR><TD>Текст 1</TD><TD>Текст 2</TD></TR> <TR><TD>Текст 3</TD><TD>Текст 4</TD></TR> </TABLE>	**Заголовок1** **Заголовок2** Текст 1 Текст 2 Текст 3 Текст 4

Відмінність тегів <TH> і <TD> тільки в тому, що заголовки таблиць виділяються напівжирним шрифтом і за замовчуванням центруються. Далі будемо розглядати таблиці без заголовків.

Атрибути тега TABLE

Для тега <TABLE> припустимі наступні атрибути:

1. BORDER - товщина рамки таблиці. На практиці найчастіше використовують значення BORDER=1 (більш товста рамка виглядає некрасиво). Якщо потрібна таблиця без рамки встановлюють BORDER=0.

CELLPADDING - відстань від рамки таблиці до тексту в гнізді

CELLSPACING - відстань між гніздами таблиці (якщо для таблиці задана ненульова рамка, то збільшення значення даного атрибута робить цю рамку товстіше, тобто додаткова відстань між гніздами заповнюється рамкою)

Використання атрибутів BORDER, CELLPADDING, CELLSPACING

HTML код	Вигляд у браузерії

<TABLE BORDER=1 CELLPADDING=0 CELLSPACING=5> <TR><TD>Текст 1</TD><TD>Текст 2</TD></TR> <TR><TD>Текст 3</TD><TD>Текст 4</TD></TR> </TABLE>	Текст 1 | Текст 2 Текст 3 | Текст 4

Використання атрибутів BORDER, CELLPADDING, CELLSPACING

HTML код	Вигляд у браузері
<TABLE BORDER=1 CELLPADDING=5 CELLSPACING=0> <TR><TD>Текст 1</TD><TD>Текст 2</TD></TR> <TR><TD>Текст 3</TD><TD>Текст 4</TD></TR> </TABLE>	Текст 1 | Текст 2 Текст 3 | Текст 4

Звичайно для таблиць із рамкою задають CELLPADDING=3 (або близько того), а CELLSPACING=0. Для таблиць без рамки звичайно обидва атрибути обнуляють.

Поєднання гнізд таблиці

Тепер зупинимося на атрибутах, які дозволяють поєднувати гнізда таблиць. Поєднання гнізд описують у межах тегів <TD> і <TH>. Поєднувати гнізда можна як по горизонталі – атрибут COLSPAN, так і по вертикалі – атрибут ROWSPAN.

<u>Поєднання гнізд по горизонталі</u>

Атрибут COLSPAN=n задає, що дане гніздо буде поширюватися по горизонталі на n колонок. У поточному рядку потрібно задавати на n-1 гнізд менше, ніж інші.

Поєднання гнізд по горизонталі

HTML код	Вигляд у браузері
<TABLE BORDER=1 CELLPADDING=3 CELLSPACING=0> <TR><TD COLSPAN=2>Текст 1</TD><TD>Далі</TD></TR> <TR><TD>Текст 3</TD><TD>Текст 4</TD><TD>Ще далі</TD></TR> </TABLE>	Текст 1 | Далі Текст 3 | Текст 4 | Ще далі

Як бачимо, в першому рядку задано два гнізда замість трьох.

<u>Поєднання гнізд по вертикалі</u>

Атрибут ROWSPAN=n задає, що дане гніздо буде поширюватися по вертикалі на n рядків. У n-1 наступних рядках потрібно задавати на 1 (або на декілька, якщо для гнізда задане ще й ROWSPAN) менше рядків.

Поєднання гнізд по вертикалі

HTML код	Вигляд у браузері
<TABLE BORDER=1 CELLPADDING=3 CELLSPACING=0> <TR><TD ROWSPAN=2>Текст 1</TD><TD>Текст 2</TD></TR> <TR><TD>Текст 4</TD></TR> <TR><TD>Ще текст</TD><TD>И ще текст</TD></TR > </TABLE>	Текст 1 / Текст 2 / Текст 4 / Ще текст / І ще текст

Розміри таблиць

Як для таблиць, так і для окремих гнізд можна задавати ширину й (теоретично) висоту. На практиці ширина для таблиць задається часто, а висота рідко, тому що найчастіше висота таблиці передбачається змінною. Ширина таблиці може задаватися як в абсолютних одиницях (за замовчуванням, у пікселах), так і у відсотках, в останньому випадку таблиця буде міняти свій розмір при зміні вікна браузера або об'ємної таблиці (як ми скоро побачимо, таблиці можна вкладати одна в іншу).

От таблиця шириною 100% (якщо змінити розміри вікна браузера, можна побачити, як вона міняє розмір слідом за ними):

Текст 1	Текст 2
Текст 3	Текст 4

Ось таблиця фіксованої ширини (200 пикселов):

Текст 1	Текст 2
Текст 3	Текст 4

Ось таблиця фіксованої висоти (100 пикселов):

Текст 1	Текст 2
Текст 3	Текст 4

Можна задавати такі атрибути і для гнізд таблиці. Відзначимо, що коли атрибут заданий у відсотках, то він відлічується від усієї таблиці.

Ось приклад таблиці з фіксованою шириною першої колонки (150 пікселей) і другої (100), а також висотою другого рядка (50).

Текст 1	Текст 2
Текст 2	Текст 4

Зазначимо, що коли встановлено ширину для одного гнізда, а в іншому гнізді тієї ж колонки буде більше даних, чим ця ширина, то значення атрибуту ширини ігнорується.

Ігнорування значення атрибуту ширини гнізда

```
<TABLE BORDER=1 CELLPADDING=3 CELLSPACING=0>
<TR><TD WIDTH=100>Текст 1</TD><TD WIDTH=100>Текст 2</TD></TR>
<TR><TD>Текст 3 ще й ще й ще й ще й ще</TD><TD>Текст 4</TD></TR>
</TABLE>
```

Текст 1	Текст 2
Текст 3 ще й ще й ще й ще й ще	Текст 4

Відзначимо, що коли в гнізді таблиці не міститься нічого, крім, можливо, пропусків, коментарів і т.і. браузер не рисує навколо нього рамку.

> Щоб рамки «порожнього» гнізда завжди відображалися вставте в гніздо, наприклад, елемент   (нерозривний пробіл).

Таблиця з порожніми гніздами

```
<TABLE BORDER=1 CELLPADDING=3 CELLSPACING=0>
<TR><TD WIDTH=40></TD><TD>Текст 2</TD></TR>
<TR><TD> </TD><TD>Текст 4</TD></TR>
</TABLE>
```

що буде виглядати так:

	Текст 2
	Текст 4

Цей момент потрібно враховувати, наприклад, при динамічній генерації таблиць на сервері - у разі відсутності даних для якогось гнізда в ній краще виводити нерозривний пропуск.

Атрибути оформлення гнізд таблиці

Для гнізд таблиць можна ще управляти:

кольором фону, задаючи атрибут BGCOLOR;
вирівнюванням по горизонталі й вертикалі, за допомогою, відповідно, атрибутів ALIGN і VALIGN, для яких припустимі значення LEFT, RIGHT і CENTER (для ALIGN), TOP, BOTTOM і CENTER (для VALIGN);
допустимістю переносу на границі слова (за допомогою атрибута NOWRAP) - це атрибут тега <TD>.

Використання атрибутів оформлення гнізд таблиці

```
<TABLE WIDTH=180 BORDER=1 CELLPADDING=3 CELLSPACING=0>
<TR><TD ALIGN=CENTER>Текст 1</TD><TD BGCOLOR=Yellow>Текст 2</TD></TR>
<TR BGCOLOR=Gray><TD NOWRAP>Текст ще текст</TD><TD>Текст ще текст</TD></TR>
<TR ALIGN=RIGHT><TD>Вправо</TD><TD>Ще вправо</TD></TR>
</TABLE>
```

Текст 1	Текст 2
Текст ще текст	Текст ще текст
Вправо	Ще вправо

4.3.11. Верстка сторінок за допомогою таблиць

Як ми вже говорили, за допомогою таблиць можна здійснювати верстку сторінок (до появи засобів точного позиціонування це був єдиний спосіб такої верстки).

> Для таких таблиць атрибути BORDER, CELLPADDING і CELLSPACING повинні бути встановлені в 0.

Відзначимо ряд моментів:

для таких таблиць звичайно використовують фіксовану ширину гнізд, у цьому випадку можна розташовувати елементи з точністю до піксела. Для всієї таблиці використовують або фіксовану ширину (звичайно 560 пікселей) або задають ширину у відсотках (т.зв. "гумовий" дизайн сторінки);
самим істотним недоліком верстки сторінок у такий спосіб є те, що браузер не відображає всю сторінку доти, поки не з'ясує низку питань щодо компонування таблиці, на якій вона заснована (наприклад, якщо забути атрибути HEIGHT або WIDTH для якогось зображення всередині такої таблиці, то відображення всієї сторінки буде затримано до одержання цього зображення, якщо в таблиці не фіксовані значення для ширини колонок, то відображення сторінки буде теж затримане й т.д.);
за допомогою таких таблиць можна вжити заходів для прискорення завантаження зображень - для цього необхідно розрізати зображення на кілька частин і вивести їх в окремих гніздах таблиці - браузер буде викачувати частини зображення паралельно. Необхідно стежити, щоб ніде між частинами

зображення не потрапили пропуски. У більшості випадків достатньо вивести частини зображення підряд без пропусків, а для переходу на новий рядок використовувати <BR>.

Ось як вивести текст у дві колонки за допомогою таблиці (зверніть увагу на третю колонку, яку ми використовуємо для організації проміжку між колонками).

Дві колонки тексту

```
<TABLE WIDTH=100% BORDER=0 CELLPADDING=0 CELLSPACING=0>
<TR>
 <TD> це в нас буде різноманітний і важливий для всіх без винятку  текст першої колонки</TD>
<TD WIDTH=20> </TD>
<TD> це в нас буде різноманітний і важливий для всіх без винятку текст другої колонки</TD>
 </TR>
</TABLE>
```

це в нас буде різноманітний і важливий для всіх без винятку текст першої колонки		це в нас буде різноманітний і важливий для всіх без винятку текст другої колонки

Приклад верстки Web-сторінки за допомогою таблиць

Приведемо типовий приклад верстки сторінки за допомогою таблиць. Припустимо, що нам необхідно створити набір сторінок для сайту, причому на кожній сторінці необхідно виділити такі області:

логотип ліворуч угорі сторінки;
набір посилань ліворуч сторінки (посилання на всі сторінки даного сайта, які важливі при переміщенні по ньому, т.зв. *панель навігації* (navigation bar));
заголовок праворуч угорі сторінки;
увесь інший простір відведений під основний зміст сторінки.

Цю інформацію треба повторювати на кожній сторінці набору. Ось як можна розв'язати це завдання (взагалі це один із стандартних підходів до розробки Web-сайтів):

на кожній сторінці створити таблицю з двох рядків і двох стовпців, ширину першого стовпця зробити фіксованою;
задати колір фону для першого стовпця відмінним від другого.

Верстка за допомогою таблиць

```
<TABLE WIDTH=100% BORDER=0 CELLPADDING=0 CELLSPACING=0>
<TR>
<TD BGCOLOR=black WIDTH=120><font color=white>Логотип</font></TD>
<TD BGCOLOR=lightgrey>Заголовок</TD>
</TR>
<TR>
<TD BGCOLOR=lightgrey WIDTH=120>Посилання</TD>
<TD>Зміст</TD>
</TR>
```

```
</TABLE>
```

Логотип	Заголовок
Посилання	Зміст

Відзначимо важливий момент. Як бачимо, все, що відноситься до службової інформації для сторінки (крім трьох закриваючих тегів </TD></TR></TABLE> знаходиться до змісту. Тим самим сторінка має такий вигляд:

1. заголовок (від початку до відкриваючого тега гнізда змісту включно) - повторюється (за винятком тексту в гнізді "заголовок");

зміст - для кожної сторінки свій;

закінчення (</TD></TR></TABLE>) – повторюється.

Можна полегшити роботу, якщо організувати *автоматичну вставку в усі сторінки сайта однієї й тієї ж інформації заголовка й закінчення сторінки*. При цьому, наприклад, у якості параметра можна передавати заголовок кожної сторінки. Звичайно це робиться на сервері, ми далі побачимо, як.

Достоїнства даного підходу:

1. простота в реалізації і особливо в автоматизації,

зрозумілість для користувача,

користувачеві завантажується за раз тільки одна сторінка, йому легко запам'ятовувати її адресу, зберігати в себе і т.д.

Недоліки даного підходу (крім затримки відрисовки до завантаження всієї таблиці):

1. якщо відбудеться обрив з'єднання, є шанс нічого не одержати на екрані зовсім (браузери поводяться не дуже добре, якщо вони відображали таблицю, а зв'язок обірвався)

при перегляді довгих документів навігаційна панель іде з екрана нагору

якщо документ занадто короткий, потрібно передбачити, щоб поле ліворуч екрана доходило до нижнього поля браузера (для цього можна вказувати для таблиці HEIGHT="100%" і не забувати для гнізд зі змістом VALIGN="top")

користувач із кожною сторінкою одержує ту саму інформацію заголовка, яка буває досить об'ємистої

Вкладені таблиці

Таблиці можна вкладати одна в одну (усередину гнізда). За допомогою вкладених таблиць можна виконувати верстку, коли звичайні таблиці не допомагають

(або виходить занадто складно). Відлік відносних розмірів ведуть від розміру гнізда, в яке вкладена таблиця.

> Використовуйте вкладені таблиці. Браузер надійніше працює з вкладеними таблицями, ніж таблицями, які ускладнені атрибутами COLSPAN, ROWSPAN і т.д. (найчастіше для розробника такий синтаксис зрозуміліше).

Вкладені таблиці, у верхньому ряді 3 гнізда, а в нижньому - два

```
<TABLE WIDTH=200 BORDER=0 CELLPADDING=0 CELLSPACING=0>
<TR><TD>
<TABLE WIDTH=100% BORDER=0 CELLPADDING=3 CELLSPACING=0>
<TR ALIGN=CENTER><TD BGCOLOR=gray>Text 1</TD>
<TD BGCOLOR=lightgrey>Text 2</TD>
<TD BGCOLOR=gray>Text 3</TD></TR>
</TABLE>
<TR><TD>
<TABLE WIDTH=100% BORDER=0 CELLPADDING=3 CELLSPACING=0>
<TR ALIGN=CENTER><TD BGCOLOR=darkgrey>Level 2</TD>
<TD BGCOLOR=black><font color=white>Level 3</font></TD></TR>
</TABLE>
</TD></TR>
</TABLE>
```

Text 1	Text 2	Text 3
Level 2	Level 3	

> Уважно слідкуйте за правильним використанням тегів <TR> і <TD>! Для звичайних таблиць браузери не дуже уважно ставляться до того, що теги <TR> і <TD> потрібно закривати (наступний відкриваючий вважається немовби закриваючим до попереднього), те у випадку вкладених таблиць у цьому разі можуть виникнути непередбачені помилки.

Додаткові теги роботи з таблицями

До цих тегів відносять:

1. <CAPTION> - задає опис для таблиці. Даний опис відображається по центру (так за замовчуванням, можна задати ALIGN) над таблицею (якщо атрибут VALIGN=top) або під таблицею (якщо VALIGN=bottom) поза рамкою таблиці.

Таблиця з тегом <CAPTION>

```
<TABLE BORDER=1 CELLPADDING=3 CELLSPACING=0>
<CAPTION VALIGN=top>Таблиця 1</CAPTION>
<TR>
<TD>Це текст у колонку 1</TD>
<TD>Це текст у колонку 2</TD>
<TR>
```

```
</TABLE>
```

Таблиця 1	
Це текст у колонку 1	Це текст у колонку 2

<THEAD>, <TBODY> і <TFOOT> групують рядки таблиці. <THEAD> задає рядки заголовку таблиці (і, наприклад, при друку будуть повторюватися вгорі кожної сторінки, якщо вона не вміщується на одній), <TFOOT> задає нижній колонтитул для таблиці (аналогічно заголовку, але друкується знизу), <TBODY> групує рядки для задання атрибутів для декількох рядків відразу (атрибути можуть задаватися й для <THEAD> і <TFOOT>). У таблиці можуть бути один <THEAD>, один <TFOOT> і кілька груп <TBODY>.

Таблиця з тегами <THEAD>, <TBODY> і <TFOOT>

```
<TABLE BORDER=1 CELLPADDING=3 CELLSPACING=0>
<THEAD BGCOLOR=Gray>
 <TR><TD>header</TD><TD>header</TD></TR>
</THEAD>
 <TBODY BGCOLOR=Yellow>
  <TR><TD>рядок 1</TD><TD>рядок 1</TD></TR>
  <TR><TD>рядок 2</TD><TD>рядок 2</TD><TR>
 </TBODY>
 <TBODY BGCOLOR=lightgrey>
  <TR><TD>рядок 3</TD><TD>рядок 3</TD><TR>
 </TBODY>
  <TFOOT BGCOLOR=Gray>
   <TR><TD>footer</TD><TD>footer</TD></TR>
  </TFOOT>
</TABLE>
```

header	Header
рядок 1	Рядок 1
рядок 2	Рядок 2
рядок 3	Рядок 3
footer	Footer

<COLGROUP> і <COL> використовують для організації угруповання колонок. <COL> задає характеристики для конкретної колонки, <COLGROUP> поєднує декілька колонок у групу (колонки задають тегом <COL> або атрибутом SPAN=n).

Таблиця з тегами <COLGROUP> і <COL>

```
<TABLE WIDTH=550 BORDER=1 CELLPADDING=3 CELLSPACING=0>
<COLGROUP BGCOLOR=gray><COL ALIGN=RIGHT><COL ALIGN=LEFT></COLGROUP>
<COL ALIGN=CENTER BGCOLOR=lightgrey>
<COLGROUP SPAN=2 BGCOLOR=yellow>
 <TR>
    <TD>рядок 1 рядок 1 рядок 1</TD>
    <TD>рядок 1 рядок 1 рядок 1</TD>
    <TD>рядок 1 рядок 1 рядок 1</TD>
    <TD>рядок 1 рядок 1 рядок 1</TD>
    <TD>рядок 1 рядок 1 рядок 1</TD>
   </TR>
   <TR>
    <TD>рядок 2 рядок 2 рядок 2</TD>
    <TD>рядок 2 рядок 2 рядок 2</TD>
    <TD>рядок 2 рядок 2 рядок 2</TD>
    <TD>рядок 2 рядок 2 рядок 2</TD>
    <TD>рядок 2 рядок 2 рядок 2</TD>
   <TR>
</TABLE>
```

рядок 1 рядок 1 рядок 1	рядок 1 рядок 1 рядок 1	рядок 1 рядок 1 рядок 1	рядок 1 рядок 1 рядок 1	рядок 1 рядок 1 рядок 1
рядок 2 рядок 2 рядок 2	рядок 2 рядок 2 рядок 2	рядок 2 рядок 2 рядок 2	рядок 2 рядок 2 рядок 2	рядок 2 рядок 2 рядок 2

4.3.12. Фрейми

Дотепер ми відображали у вікні браузера один Html-документ. Існує спосіб змусити відобразити відразу кілька документів так, що кожний з них буде займати певну область екрана. Технологія, яка це дозволяє, називається *технологією фреймів*.

При роботі з фреймами спочатку браузер завантажує спеціальний документ, т.зв. *кореневий документ набору фреймів*, який описує конфігурацію фреймів сторінки й те, які документи повинні завантажуватися на екран в який фрейм.

Кореневий документ, за одним винятком, не містить відображуваної інформації. Вона перебуває всередині документів, що завантажують у фрейми. Це

звичайні Html-документи, вони можуть бути відображені й без фреймів, для них тільки може дещо відрізнятися реалізація гіперпосилань.

Насамперед відзначимо, кореневий документ не містить тега <BODY>, замість нього присутній спеціальний тег <FRAMESET>.

Загальна структура кореневого документа

```
<HTML>
<HEAD><TITLE>Title</TITLE></HEAD>
<FRAMESET>
<!-- опис фреймів, див. далі -->
</FRAMESET>
<NOFRAMES>
Текст для тих, хто не підтримує фрейми
</NOFRAMES>
</HTML>
```

Відзначимо, що всередині тега <NOFRAMES> задають інформацію для таких браузерів, які не підтримують роботу з фреймами, це і є єдиний виняток з того, що кореневий документ відображуваної інформації не містить.

Основним атрибутом тега <FRAMESET> є один з двох взаємовиключних атрибутів ROWS і COLS. Вони задають, як у вікні браузера розташовані фрейми. Горизонтально, якщо задане ROWS, або вертикально, якщо задане COLS, і скільки місця займає кожний фрейм.

```
<FRAMESET COLS="ширина фрейму, ширина, ...">
<FRAMESET ROWS="висота фрейму, висота, ...">
```

Ширина визначається в пікселах, у відсотках або у вигляді "*", що означає "максимально можлива".

Задання вертикального розташування фреймів

```
<FRAMESET COLS="120, 40%, *">
```

Цей приклад задає три вертикальні фрейми, лівий - фіксованої ширини 120 пікселов, середній - відносної ширини 40% від ширини вікна браузера, правий повинен займати все місце, що залишилося по ширині.

Тег <FRAMESET> не відображає що-небудь, якщо всередині нього не задані теги <FRAME>. Формат тега наступний: `<FRAME SRC="url">`, де `url` задає адресу документа, який буде завантажений у фрейм. Тегів <FRAME> всередині <FRAMESET> повинно бути стільки, скільки перераховано в значенні атрибута ROWS або COLS.

Задання горизонтального розташування фреймів

```
<HTML><HEAD><TITLE>Gorizont Frames</TITLE></HEAD>
<FRAMESET ROWS="200, *">
<FRAME SRC="jac_01.html">
<FRAME SRC="jac_02.html">
</FRAMESET>
</HTML>
```

Атрибути фреймів

Для фреймів можна задавати такі атрибути:

1. SCROLLING (Yes/No/Auto) – визначає, чи виводити смуги прокручування для документів. За замовчуванням AUTO (виводити за необхідністю).

NORESIZE - якщо заданий, користувач не може сам міняти розміри фреймів за допомогою миші. Задається для <FRAME>.

FRAMEBORDER - 0 або 1, якщо 0, видимі границі фреймів не відображаються зовсім. Атрибут може бути заданий і для конкретного фрейму, і для <FRAMESET>.

BORDER - задається тільки для тега <FRAMESET> і задає товщину границь фреймів даного набору в пікселах.

MARGINHEIGHT - розмір верхнього й нижнього полів фрейму в пікселах.

MARGINWIDTH - розмір правого й лівого полів фрейму в пікселах.

NAME - рядок, який задає ім'я для фрейму, це ім'я потім буде використовуватися в атрибуті TARGET тега A (гіперпосилання) для завантаження документа по цьому посиланню у фрейм із заданим іменем.

Приклад. Документ, що використовує ці атрибути).

Тег <FRAME> з атрибутами MARGINHEIGHT та MARGINWIDTH

```
<HTML><HEAD><TITLE>Example Frames </TITLE></HEAD>
<FRAMESET ROWS="220,*" FRAMEBORDER=1 BORDER=6 BORDERCOLOR=black>
  <FRAME NAME="first" SRC="jac_01.html" MARGINWIDTH=25  NORESIZE>
  <FRAME NAME="second" SRC="jac_02.html" SCROLLING=No MARGINHEIGHT=25>
</FRAMESET>
</HTML>
```

Зверніть увагу, що коли заданий тільки один з параметрів MARGINHEIGHT або MARGINWIDTH, то другий з параметрів має значення 0, а не значення за замовчуванням, як було б, якби вони були задані разом.

Організація навігації на сайті за допомогою фреймів

Розв'яжемо знову завдання організації сайту, тільки цього разу звернемося до фреймів, а не до таблиць. Нам додатково знадобляться:

кореневий файл

файл із навігаційною панеллю

Теоретично можна було б задати ще й верхній фрейм із логотипом і заголовком, але на практиці більше двох фреймів на одній сторінці задавати рекомендується тільки в крайніх випадках (процес завантаження такого набору файлів досить стомлюючий).

Кореневий файл для сайту з фреймами

```
<HTML>
<HEAD><TITLE>Example Frames Site</TITLE></HEAD>
<FRAMESET COLS="140, *" FRAMEBORDER=0>
<FRAME NAME="index" SRC="jac_index.html" MARGINWIDTH=25 NORESIZE>
<FRAME NAME="content" SRC="jac_content.html" SCROLLING=No MARGINHEIGHT=25>
</FRAMESET>
</HTML>
```

Файл с навігаційною панеллю

```
<HTML><HEAD></HEAD>
<BODY BGCOLOR=grey><BR>
<!-- ПОСИЛАННЯ -->
 <UL>
  <LI><A TARGET="content" HREF="jac_01.html">Лекція 1</A></LI>
  <LI><A TARGET="content" HREF="jac_02.html">Лекція 2</A></LI>
  <LI><A HREF="navigation.html">Навігація</A></LI>
  <LI><A TARGET="_top" HREF="../jac_web.html#here6">повернення</A></LI>
 </UL>
</BODY>
</HTML>
```

У файлах зі контентом нічого додаткового вказувати не треба, за винятком того, що теги <A> можуть використовувати атрибут TARGET. Якщо атрибут заданий, то перехід за посиланням приведе до завантаження документа у фрейм, чиє ім'я вказане як значення атрибута TARGET.

Відзначимо спеціальний синтаксис `TARGET="_top"`, який означає, що при виборі посилання, для якого задане таке значення атрибута TARGET, документ займе весь екран браузера. Так можна реалізувати "вихід" із сайта із фреймами й гарний тон Web-дизайну вимагає, щоб усі посилання на зовнішні документи супроводжувалися таким атрибутом.

Слід зауважити, що, крім `_top`, існують ще інші визначені імена:

`_self` - поточний фрейм (за замовчуванням прийняте `TARGET="_self"`),
`_parent` - вікно на рівень вище,
`_blank` - нове вікно.

Якщо задати в TARGET ім'я неіснуючого фрейму, браузер теж відкриє нове вікно й завантажить у нього документ (аналогічно `_blank`).

Зазначимо, що коли для більшості гіперпосилань документа необхідно задавати атрибут TARGET з однаковим значенням, можна задати значення за замовчуванням за допомогою тега <BASE>.

Використання тега <BASE> для атрибута TARGET

```
<HTML>
<HEAD><BASE TARGET="content"></HEAD>
<BODY BGCOLOR=grey><BR>
<UL>
  <LI><A HREF="jac_01.html">Лекція 1</A></LI>
  <LI><A HREF="jac_02.html">Лекція 2</A></LI>
  <LI><A TARGET="_self"HREF="navigation.html">Навігація</A></LI>
  <LI><A TARGET="_top" HREF="../jac_web.html#here6">повернення</A></LI>
</UL>
</BODY>
</HTML>
```

Зверніть увагу, що коли ми прагнемо при цьому, щоб вибір деякого посилання однаково приводив до завантаження в поточний фрейм, ми зобов'язані використовувати `_self`.

Вкладені фрейми

Можна вкладати теги <FRAMESET> один в одний, при цьому отримують складні конфігурації фреймів.

Вкладені теги <FRAMESET>

```
<FRAMESET ROWS="100, *">
<FRAME NAME="the_top" SRC="the_top.html">
<FRAMESET COLS="120, *">
<FRAME NAME="the_navpane" SRC="the_navpane.html">
<FRAME NAME="the_content" SRC="the_content.html">
</FRAMESET>
</FRAMESET>
```

Зловживати більшою кількістю фреймів на одній сторінці не рекомендується.

Достоїнства й недоліки фреймів

Достоїнства фреймів:

навігація по сайту виглядає більш природною, навігаційна панель завжди залишається на екрані при прокручуванні іншої інформації (як у прикладі)

у загальному випадку заощаджується мережний трафік, тому що при переходах за посиланням всередині сайтів з фреймами завантажується не уся сторінка, а тільки один фрейм за раз (у прикладі - файл із змістом)

Недоліки фреймів, їх насправді дещо більше, ніж достоїнств:

при першому завантаженні сторінки із фреймами замість одного документа користувач завантажує декілька, що стомливо й займає більше часу;
збереження документів утруднене, хоча сучасні браузери дозволяють відразу зберігати весь набір документів у всіх фреймах, але мати справу з одним документом завжди простіше;
коли користувач перебуває на сайті із фреймами, у рядку адреси перебуває адреса кореневого документа. Це незручно для тих користувачів, яких цікавлять адреси різних документів всередині сайта. Крім того, легко забути у всіх посиланнях на зовнішні документи вказувати TARGET=_top, що веде до завантаження зовнішніх документів у поточний фрейм, а це незручно й дратує користувача;
установка закладок на конкретний фрейм із набору утруднена;
пошукові системи можуть погано індексувати документи всередині фреймів, точно так само не зовсім очевидно, що вони повинні повертати, якщо такий документ відповідає критерію пошуку.

Вкладання документів - плаваючі фрейми

Плаваючий фрейм (floating frames). - це фрейм, який не має постійного положення у вікні браузера і може перебувати в документі де завгодно. Плаваючий фрейм являє немовби вікно у яке може бути завантажений інший документ. Це дозволяє включити один документ всередину іншого.

Для визначення плаваючого фрейму використовують парний тег <IFRAME>. Документ, який завантажують, задають як значення атрибута SRC. Припустимі такі атрибути: HEIGHT, WIDTH, ALIGN, HSPACE, VSPACE, MARGINHEIGHT, MARGINWIDTH.

Визначення плаваючого фрейму

```
<IFRAME SRC="examples_html/ex_02_1.html" HEIGHT="200" WIDTH="100%"></IFRAME>
```

Зауважимо, що посилання працюють так само, як і для звичайних фреймів (нові документи завантажуються в той же фрейм, працює TARGET="_top").

Вкладання документів – використання тега OBJECT

Більш універсальний спосіб визначення вкладених документів - використання тега <OBJECT>. Тег має такі атрибути:

DATA="url" - задає місцезнаходження даних, які потрібно помістити на сторінку (у наведеному прикладі це адреса документа)

TYPE="тип" - задає тип даних (т.зв. MIME-тип, для Html-сторінок потрібно вказувати TYPE="text/html"
HEIGHT, WIDTH і т.д.

Вкладення документу за допомогою тега <OBJECT>

```
<OBJECT DATA="examples_html/ex_02_1.html" HEIGHT="200" WIDTH="100%">
Цей текст відобразиться на екрані, якщо файл завантажити не вдалося
</OBJECT>
```

4.3.13. Карти-Зображення (image maps)

Карти-Зображення (image map) - це зображення, всередині якого задані області різної форми, зв'язані з посиланнями, так що вибір області веде до переходу по відповідному посиланню.

Розрізняють два види карт: серверні й клієнтські.

Серверні карти (server-side image maps)

Дії користувача (клацання мишею) обробляє програма, яка знаходиться на сервері. При клацанні браузер ініціює перехід за посиланням, для якого формує URL. Частиною URL є дві піксельні координати (x і y) тієї точки, де відбулося клацання. Для задання такої карти потрібно помістити тег <IMG> у тег <A HREF="url"> (як для зображення-посилання) і задати для тега <IMG> значення атрибута ISMAP=true.

Визначення серверної карти-зображення

```
<A HREF="/cgi-bin/map.cgi"><IMG BORDER=0 SRC="map.gif" ISMAP=true></A>
```

Після цього клацання користувачем на зображенні в точці з координатами (10,20) буде сформований такий URL: `/cgi-bin/map.cgi?10,20` (передбачено, що серверна програма map.cgi може обробити такі дані).

Серверні карти мають два недоліки:

вимагають програми обробки на сервері;
браузер не знає нічого про те, яка область пов'язана з яким посиланням і не може нічого підказати користувачеві (наприклад, відобразити посилання в рядку статусу, показати спливаючу підказку-tooltip і т.д.), крім того, неясно, що робити з такою картою, якщо користувач відключив завантаження зображень.

Ці недоліки є настільки серйозними, що серверні карти використовують тепер рідко, замінивши альтернативною - *клієнтськими картами*.

Клієнтські карти (client-side image maps)

Технологія клієнтських карт не потребує серверної програми для обробки. Клієнтська карта описана в Html-документі. Її обробляє браузер. Щоб задати клієнтську карту треба:

задати парний тег <MAP> з атрибутом NAME, який задає ідентифікатор карти,
всередині тега <MAP> задати набір областей зображення – тег <AREA>,
для тега <IMG> карти-зображення, задати атрибут USEMAP="#идентификатор-карти"

Визначення клієнтської карти-зображення

```
<IMG SRC="map_jac.gif" BORDER=0 USEMAP="#jac_map">
<MAP NAME="jac_map">
теги AREA
</MAP>
```

Розглянемо, як задавати теги <AREA>. Тег має такі атрибути:

1. SHAPE - задає форму області.

Значення атрибуту SHAPE

Скорочене значення	Повне значення	Вигляд
rect	rectangle	прямокутник
poly	polygon	многокутник
circ	circle	коло

COORDS - задає координати області, формат якої залежить від значення атрибута SHAPE

Інтерпретація значень атрибута COORDS

Значення атрибута SHAPE	Формат атрибута COORDS	Зміст
rect	"x1,y1,x2,y2"	лівий верхній і правий нижній кути прямокутника
poly	"x1,y1,x2,y2,...,xn,yn"	пара координат описує вершину многокутника
circ	"x,y,r"	координати центру і радіус кола

Зазначимо, що для координат припустимі дробові значення (10.5 і т.д.), тому коли ми координати генеруємо (наприклад, на сервері), округляти не обов'язково.

HREF задає посилання за яким відбудеться перехід. Припустимі також атрибути TARGET і NOHREF, останній у випадку задання зі значенням true указує, що вибір даної області не веде до переходу по посиланню.

ALT задає текст, який описує область, і буде відображатися як спливаюча підказка, або замість зображення, якщо користувач відключить завантаження зображень.

Повне визначення клієнтської карти-зображення

```
<IMG SRC="img/map.gif" WIDTH="186" HEIGHT="150" USEMAP="#ex" BORDER="0" ALT="background">
<MAP NAME="ex">
<AREA SHAPE="rect" COORDS="19,22,46,54" NOHREF=true ALT="inactive">
<AREA SHAPE="rect" COORDS="14,15,157,61" HREF="ex/08-rect.html" ALT="rectangle">
<AREA SHAPE="circle" COORDS="50,105,29" HREF="ex/08-circ.html" ALT="circle">
<AREA SHAPE="poly" COORDS="114,73,153,79,158,117,123,135" HREF="ex/08-poly.html" ALT="polygon">
</MAP>
```

Увага! У разі перекриття областей, пріоритет має область, для якої тег <AREA> розташований у визначенні раніше. Наприклад, якби тег для неактивної області стояв після тега для прямокутника, ця область не сприймалася б як неактивна.

Клієнтські карти зручніше за все задавати в Html-редакторі, який надає такі можливості.

4.3.14. Форми

Форми використовують для організації обміну даними між Web-клієнтом і серверною програмою. Розглянемо теги й атрибути опису форм у документі HTML.

4.3.15. Тег <FORM>

Розробник документа задає набір елементів керування (поля введення, списки, кнопки і т.д.). Елементи керування форми розміщають всередині парного тега <FORM>.

Користувач вводить дані в елементи керування (*заповнює форму*) і віддає команду пересилання даних на сервер (*підтверджує передачу даних внесених в елементи керування форми*, звичайно, натискає кнопку типу submit). Після цього дані елементів керування потрапляють на сервер для обробки серверною програмою.

Атрибути тега <FORM>:

1. ACTION - задає URL-адресу, на яку будуть передані дані форми для обробки (звичайно активізація URL приводить до запуску серверної програми - оброблювача)

ENCTYPE - задає тип даних при пересиланні на сервер

METHOD - задає метод пересилання даних (можливі значення - "POST" і "GET")

NAME - задає ім'я форми (ім'я має сенс, коли на клієнтові використовують мову програмування Javascript або аналогічну)

TARGET – задає фрейм, в який будуть видані результати обробки форми (можна заповнити форму в одному фреймі, а результати одержати в іншому або, наприклад, в новому вікні).

Тег FORM

```
<FORM ACTION="/cgi-bin/process.cgi" METHOD="POST">
 елементи
</FORM>
```

В усіх прикладах для елементів керування включено тег <FORM>. Це підкреслює, що елементи керування використовують лише в контексті форм!!!

Елементи керування форми

Розглянемо, як задавати елементи керування у формах. Зазначимо:

1. елемент форми може передати дані, якщо має значення атрибута NAME (тобто NAME обов'язковий атрибут, якщо елемент буде передавати дані);

послідовність переходу між елементами форми визначає значення атрибута TABINDEX (переміщення йде від елементів з меншими значеннями до елементів з більшими);

елементу форми можна встановити атрибут READONLY ("тільки для читання");

елементу форми можна встановити атрибут DISABLED (заблокований, неактивний стан елемента).

Більшість елементів керування форми задають за допомогою тега <INPUT> з різними значеннями атрибута TYPE. Тег INPUT одиночний.

<u>Поля введення</u>

<INPUT TYPE="text"> задає звичайне поле введення. Можливі атрибути:

SIZE - ширина поля введення в символах (насправді ця ширина - один з найменш керованих параметрів в HTML, звичайно вона розраховується на підставі середньої ширини символу шрифту типу Courier New незалежно від того, який шрифт заданий насправді, наприклад, за допомогою таблиці стилів, так що число символів, які ми бачимо при введенні може зовсім не збігатися з SIZE)

MAXLENGTH - максимальне число символів, яке можна ввести в поле введення (якщо більше SIZE - буде організовано горизонтальне прокручування в міру потреби)

VALUE - дані для поля введення. Якщо задане заздалегідь, поле буде мати зміст відразу після завантаження сторінки.

Задання полів введення тексту

```
<FORM ACTION="/cgi-bin/process.cgi" METHOD="POST">
<INPUT TYPE=text VALUE="Ivanov" NAME="fname" SIZE=20>
<INPUT TYPE=text DISABLED VALUE="Ivanov" NAME="fname_dis" SIZE=20>
<INPUT TYPE=text READONLY VALUE="Ivanov" NAME="fname_ro" SIZE=20>
</FORM>
```

<INPUT TYPE="password"> задає поле введення пароля (текст при введенні відображається зірочками). При пересиланні пароль передається без шифрування. В іншому немає відмінності від звичайного поля введення. Атрибути ті ж.

Задання поля введення паролю

```
<form>
<INPUT TYPE=password VALUE="pwd" NAME="pwd" SIZE=20>
</form>
```

***

<INPUT TYPE="file"> задає поле введення імені файла для пересилання на сервер. При відображенні, таке поле супроводжується кнопкою Browse..., яка відкриває вікно вибору файла. Атрибут VALUE не задають (якщо задане, то ігнорується).

Задання поля введення імені файлу

```
<form>
<INPUT TYPE=file NAME="ifile" SIZE=20>
</form>
```

<TEXTAREA> - парний тег, який визначає поле для введення користувачем багаторядкового тексту. Текст, який перебуває всередині даного тега, відображається в поле введення при завантаженні (аналогічно значенню атрибута VALUE для звичайного поля введення). Можливі атрибути:

COLS - ширина поля в символах;

ROWS - число рядків у полі (тексту може бути й більше тоді з'являться смуги прокручування);

WRAP - управляє переносом тексту в поле по границі слова.

Значення атрибута WRAP

Значення	Зміст

Soft	дані відображуються з переносом, але переноси на сервер не передаються
Hard	дані і відображаються, і передаються з переносом
Off	дані відображаються так, як користувач їх набрав (щоб задати перенос явно треба нажати Ctrl-Enter, інакше переносу не буде)

Зазначимо, що коли заданий перехід на новий рядок у тексті всередині <TEXTAREA>, то перехід відобразиться незалежно від значення WRAP (у цьому змісті <TEXTAREA> схожий на <PRE>).

Зазначимо, що за замовчуванням <TEXTAREA> використовує шрифт фіксованої ширини (Courier New), а звичайні поля введення – змінної ширини.

Задання тега <TEXTAREA>

```
<form>
<TEXTAREA ROWS=3 COLS=20>Це перший рядок тексту
Це другий рядок тексту
Це третій рядок тексту
Це четвертий рядок тексту
</TEXTAREA>
</form>
```

Перемикачі

Перемикачі бувають двох видів - незалежні (check boxes) й залежні (radio buttons).

1. <INPUT TYPE="checkbox"> задає незалежний перемикач, який не об’єднаний в групу й може перебувати у двох станах "включене" (checked) і "виключене". Можливі атрибути:

CHECKED - задає, включений чи ні перемикач. Перемикач передає дані якщо для нього задане CHECKED (вибір користувачем перемикача аналогічний завданню CHECKED).

VALUE - задає, яке значення передається, якщо для перемикача задане CHECKED. За замовчуванням "on".

Незалежні перемикачі

```
<form>
<INPUT TYPE="checkbox" NAME="option" VALUE="opt">
<INPUT TYPE="checkbox" NAME="option2" CHECKED>
<INPUT TYPE="checkbox" NAME="option3" CHECKED DISABLED>
</form>
```

<INPUT TYPE="radio"> задає залежний перемикач. Такі перемикачі завжди працюють у групах, вибір одного перемикача групи приводить до

скасування вибору інших. Групу утворюють перемикачі з однаковим значенням атрибута NAME. Можливі атрибути:

CHECKED - задає, включений чи ні перемикач. Перемикач передає дані, тільки, якщо для нього задане CHECKED. У включеному стані може бути тільки один перемикач групи.

VALUE - задає, яке значення передається, якщо для перемикача задане CHECKED (або його вибрав користувач).

Залежні перемикачі

```
<form>
<INPUT TYPE="radio" NAME="color" VALUE="red">
<INPUT TYPE="radio" NAME="color" VALUE="green" DISABLED CHECKED>
<INPUT TYPE="radio" NAME="color" VALUE="blue">
<INPUT TYPE="radio" NAME="tone" VALUE="light">
<INPUT TYPE="radio" NAME="tone" VALUE="dark" CHECKED>
</form>
```

Списки

Для задання списків використовують парний тег <SELECT>. Всередині розташовані елементи списку, які задають тегом <OPTION>. Атрибути тега <SELECT>:

SIZE - задає кількість видимих елементів списку. Якщо SIZE="1" - створюється список, що випадає (combo box), якщо значення більше 1 – невипадаючий список (ширину списку встановлювати не можна),

MULTIPLE - якщо заданий, визначає, що в списку можна одночасно вибирати кілька елементів (Ctrl+клацання мишею). Допустимо, тільки якщо SIZE>1

Тег <OPTION> має такі атрибути:

VALUE - задає значення, яке буде передаватися формі, якщо обраний даний елемент списку,

SELECTED - означає, що даний елемент списку обраний

Текст, який буде відображатися в списку, повинен іти після тега <OPTION>.

Список, що випадає

```
<form>
<SELECT NAME="students" SIZE="1">
<OPTION VALUE="S1">Ivanov
<OPTION VALUE="S2">Petrov
<OPTION VALUE="S3" SELECTED>Sidorov
</SELECT>
</form>
```

Невипадаючий список

```
<form>
<SELECT NAME="students2" SIZE="3">
<OPTION VALUE="S1">Ivanov
<OPTION VALUE="S2">Petrov
<OPTION VALUE="S3" SELECTED>Sidorov
</SELECT>
</form>
```

Список, який дозволяє вибрати декілька елементів

```
<form>
<SELECT NAME="students3" SIZE="3" MULTIPLE>
<OPTION VALUE="S1" SELECTED>Ivanov
<OPTION VALUE="S2">Petrov
<OPTION VALUE="S3" SELECTED>Sidorov
</SELECT>
</form>
```

Кнопки

Кнопки використовують для підтвердження або скидання даних форми й для інших подібних дій.

Кнопка підтвердження передачі даних

<INPUT TYPE="submit"> задає кнопку підтвердження форми. ЇЇ вибір ініціює підтвердження форми й пересиланню даних на сервер.

Можна встановити кілька кнопок такого типу для однієї форми. В такому разі для кожної кнопки треба задати унікальне значення атрибута NAME. При виборі кнопки форма підтверджується, дані передаються тільки кнопки, яка була підтверджена, а серверна програма може визначити, яка із кнопок була обрана.

Атрибут VALUE визначає напис на кнопці й значення, яке буде передано разом з даними форми. Слід зазначити, що коли кнопка такого типу на формі одна, то натискання Enter на формі веде до підтвердження цієї форми (як немовби вона була натиснута).

Кнопка підтвердження передачі даних

```
<form>
<INPUT TYPE="submit" NAME="Next" VALUE="Go Next">
</form>
```

Go Next

Кнопка скидання форми

<INPUT TYPE="reset"> задає кнопку скидання форми, усі значення елементів форми вертає до первісних, ніякі дані не передає, атрибут NAME можна опустити.

Кнопка виклику засобів алгоритмічної обробки даних форми

<INPUT TYPE="button"> задає кнопку загального виду, за замовчуванням підтвердження форми не виконує, роботу з кнопкою забезпечує програма, що написана мовою програмування Javascript.

Кнопка-зображення

<INPUT TYPE="image"> задає зображення, що працює як кнопка. Така кнопка завжди підтверджує форму, при цьому разом з даними форми передаються два елементи: ім'я-кнопки.x і ім'я-кнопки.y, що задають x і y - координати місця, де клацнув мишею користувач. URL зображення задають атрибутом SRC, як і для тега <IMG>.

Кнопка-зображення

```
<form>
<INPUT TYPE="image" SRC="images/Nextarrow2.gif" NAME="Next">
</form>
```

Кнопка загального виду

<BUTTON> - парний тег, який створює кнопку, на якій відображається текст, що перебуває всередині тега (можливо, з форматуванням, зображеннями і т.і.). Для тега необхідний атрибут TYPE, який приймає значення "submit", "reset" або "button". Кнопка створена в такий спосіб є елементом керування, як кнопки відповідного типу створені тегом <INPUT> (підтверджують, скидають форму й т.д).

Кнопка загального виду

```
<form>
<BUTTON TYPE="submit" NAME="next">
<IMG SRC="images/Nextarrow2.gif" WIDTH="25" HEIGHT="25" ALIGN="absmiddle">
<B>Next</B>
</BUTTON>
</form>
```

На відміну від тега <INPUT TYPE="image"> вона відображується як кнопка.

Сховані елементи

Особливу роль відіграють сховані елементи керування визначені у формі. Дані схованих елементів не відображуються у вікні браузера, їх задають тільки при генерації сторінки або за допомогою засобів мови програмування сценаріїв Javascript. Сховані елементи дозволяють, наприклад, зберігати дані при переході між сторінками.

Для задання схованого елемента використовують тег <INPUT TYPE="hidden" NAME="ім'я" VALUE="значення">.

РОЗДІЛ 5. ОБ'ЄКТНА МОДЕЛЬ WEB-ДОКУМЕНТА

5.1. Парадигма об'єктно-орієнтованого підходу ООП

Сучасна індустрія інформаційного обміну спирається на парадигму об'єктно-орієнтованого підходу ООП. ООП – це методологія аналізу й синтезу в області інформаційних технологій, у фундаменті якої лежить поняття про інформаційний опис сутностей реального світу як об'єктів.

Розглянемо сутність реального світу - телефон. Аналіз цієї сутності дозволяє сформулювати його опис у наступній формі: телефон – технічний пристрій для реалізації інформаційного обміну між абонентами. Він виконує наступні дії (реалізує функції):

- надає кошти набору номера;
- надає кошти відображення;
- надає кошти перетворення електричного сигналу в звуковий;
- і т.д. (пропозиція з аудиторії).

Крім цього він має такі властивості:

- стаціонарний / мобільний,
- реалізує стандарт передачі даних,
- колір,
- форма,
- і т.д. (пропозиції з аудиторії).

Телефон може перебувати в стані:

- включений/виключений,
- у режимі розмови/у режимі очікування,
- і т.д. (пропозиції з аудиторії).

Усе в сукупності дозволяє описати конкретний телефон як об'єкт, а це комплекс «властивості/функції/стану».

В області інформаційних технологій алгоритмічна обробка інформації займає вирішальне місце, тому об'єктна модель елементів інформаційного обміну дуже важлива для розуміння можливостей користувача.

5.2. Модель структури Html-документа

Обмін інформацією в мережі Інтернет побудований на обробці файлів різного типу. Найбільш загальна назва цих файлів – Web-документи. Документ HTML є одним з видів Web-документа.

При обробці документа HTML браузер будує його об'єктну модель DOM (Document Object Model).

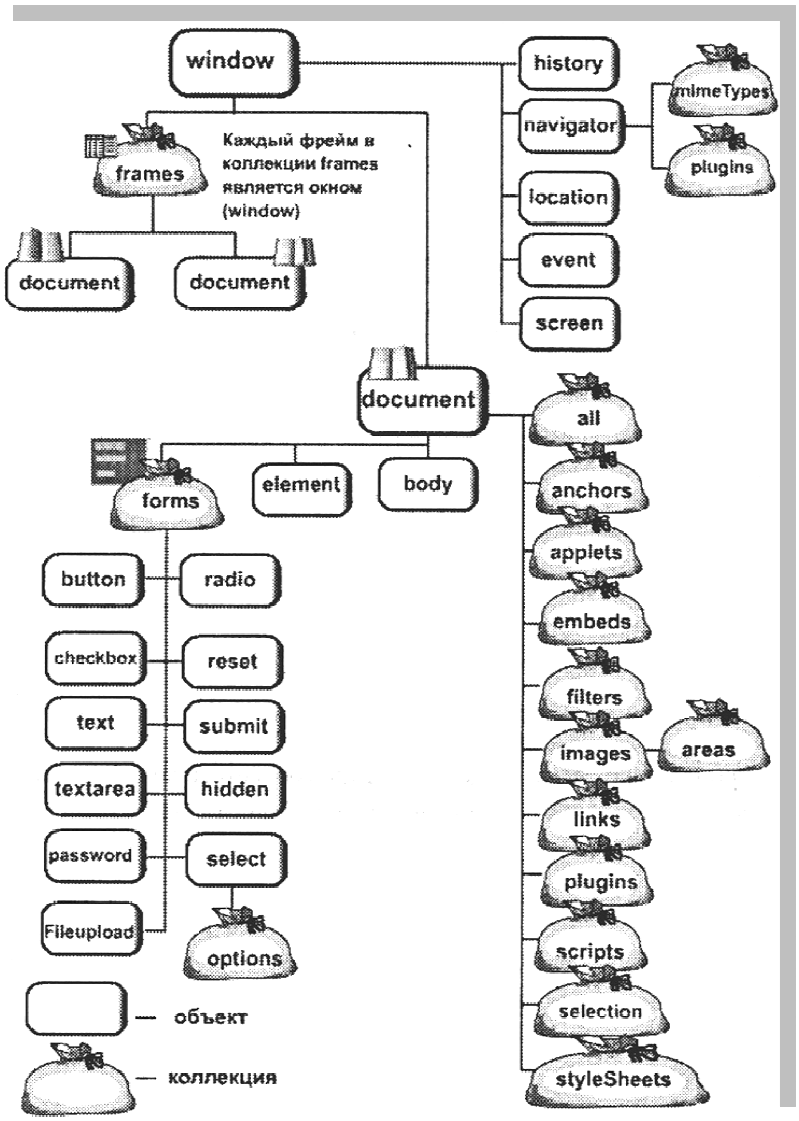
DOM – це ієрархічна (деревоподібна) структура моделі даних. Кожний вузол структури є конструкцією мови HTML (елемент, атрибут, коментар і т.п.). Надалі ми будемо вживати в якості синонімів такі поняття: ***«структура Html-документа»*** і ***«дерево Html-документа».***

Браузер надає розроблювачеві засоби для абсолютної й відносної адресації вузлів структури документа HTML.

За допомогою мови програмування можна обробляти структуру документа HTML: виконувати адресацію вузлів, пошук за різними критеріями, вибірку інформації і її зміну.

5.3. Об'єкт document і його колекції

Модель даних документа HTML описує об'єкт `document`, який є формальним представленням структури документа.



Властивості й методи об'єкта `document`, дозволяють одержати інформацію про документ у цілому, а також про кожну його частину. Елементи об'єкта document зібрані в колекції.

Колекцією називають набір об'єктів, які мають яку-небудь загальну властивість (ознаку). Очевидно, один об'єкт може бути членом декількох колекцій. Оскільки в будь-якій колекції може бути декілька елементів, їх можна порахувати, або пронумерувати. Такий номер члена колекції називають індексом, а саму операцію нумерації - ***індексацією***.

Таблиця 1 - Колекції об'єктів document.

Колекція	Склад
`All`	Усі елементи, що втримуються в документі

anchors	Об'єкти A, для яких визначено яке-небудь із властивостей name або id
applets	Усі об'єкти APPLET в документі
childnodes	Елементи й вузли текстового типу, що є безпосередніми нащадками поточного вузла
children	Об'єкти, що є безпосередніми нащадками поточного об'єкта
embeds	Усі об'єкти EMBED в документі
Forms	Усі об'єкти FORM в документі
frames	Усі об'єкти FRAME в документі
images	Усі об'єкти IMG в документі
Links	Об'єкти A, для яких визначена властивість href, а також усі об'єкти AREA
namespaces	Усі об'єкти namespace в документі
scripts	Усі об'єкти SCRIPT в документі
stylesheets	Усі об'єкти stylesheet в документі, представлені елементами LINK і STYLE

5.4. Адресація вузлів DOM

Для доступу до елементів документа HTML (вузлів документа) використовують два способи - ***абсолютну й відносну адресацію вузлів DOM***.

5.4.1. Абсолютна адресація вузлів

При абсолютній адресації використовують імена колекцій і індексацію їх членів. Суть механізму абсолютної адресації в тому, що адресуючи який-небудь вузол у документі, ми послідовно, починаючи з об'єкта document, перераховуємо всіх його предків, що є членами якої-небудь з колекцій. Наприклад:

```
document.all('my_div').item('name1').children(3)
```

Кожна колекція має властивість `length` - кількість елементів колекції. Властивість `length` дозволяє послідовно перебрати всі члени колекції від першого до останнього.

Крім властивості `length`, кожна колекція має метод `item(index)`. Метод дозволяє звернутися до будь-якого члена колекції за індексом (аргумент методу). У якості індексу можуть виступати:

порядковий номер члена колекції;

значення властивості `id;`

властивість `name`.

До потрібного члена колекції можна одержати доступ і за допомогою конструкції: `имяКоллекции(index_l, index_2),`

де `index_l` — обов'язковий аргумент, що являє собою первинний індекс члена колекції; `index_2` — необов'язковий аргумент, значенням якого є номер члена колекції у списку, який був отриманий відбором з колекції всіх її членів, властивість `id` яких збігається зі значенням `index_1`.

Для кожного вузла документа номер його позиції в колекції `all` можна одержати за допомогою властивості `sourceindex` об'єкта, відповідного цьому елементу.

Для одержання доступу до кореневого елементу документа HTML використовують властивість `documentelement` об'єкта document.

5.4.2. Відносна адресація вузлів

Відносну адресацію елементів HTML браузер реалізує за допомогою шести властивостей.

Таблиця 2 - Властивості, що реалізують функціонал для відносної адресації

Властивість	Адресований вузол
`firstchild`	Перший безпосередній нащадок поточного вузла
`Lastchild`	Останній безпосередній нащадок поточного вузла
`nextsibling`	Наступний відразу після поточного, що має з ним того самого

	батька
previoussibling	Що перебуває безпосередньо перед поточним, що й має з ним того самого батька
parentelement	Батько поточного вузла; якщо поточний вузол є кореневим, має значення null; властивість еквівалентна з parentnode
parentnode	Батько поточного вузла; якщо поточний вузол є кореневим, має значення null; властивість еквівалентна parentelement

5.5. Ідентифікація і характеристики вузла

Якщо для елемента визначений атрибут ID, то значення цього атрибута можна використовувати в сценаріях Javascript як ім'я об'єкта, відповідного цьому елементу. Унікальний ідентифікатор для об'єкта можна згенерувати за допомогою властивості uniqueid(). Ця властивість при багаторазовому виклику для того самого елемента буде давати ідентичний результат (однакове значення). Однак результати виклику uniqueid() для того самого елемента будуть різними при кожному новому завантаженні документа браузером.

Крім унікального ідентифікатора, вузли мають наступні властивості (характеристики).

Таблиця 3 - Властивості (характеристики) вузлів

Властивість	Значення
nodename	Ім'я вузла, яким є ім'я елемента або атрибута, або значення #text, якщо вузол текстовий; тільки для читання
nodetype	Тип вузла: 1 — елемент, 3 — текстовий вузол
nodevalue	Значення вузла, яким є текст вузла, якщо вузол текстовий, значення атрибута, якщо вузол є атрибутом, null, якщо вузол є елементом
tagname	Ім'я тега

Усі властивості в табл. 3 доступні тільки для читання.

Властивість nodevalue доступна як для читання, так і для запису.

Значення атрибута можна також одержати за допомогою методу `getatribute()`, аргументом якого повинне бути ім'я цього атрибута. Цей метод повинен бути викликаний для об'єкта, відповідного тому елементу, значення атрибута якого потрібно одержати.

5.6. Пошук вузла

Для реалізації пошуку в документі браузер використовує три критерії:

унікальний ідентифікатор `ID`;
значення атрибута `name`;
ім'я тега.

Нижче наведені методи, що реалізують пошук за цими критеріями:

`document.getelementbyid(sid)` — у всьому документі знаходить і повертає перший об'єкт, відповідний елементу, в якого атрибут `ID` має значення аргумента методу;

`document.getelementsbyname(sname)` - у всьому документі знаходить усі об'єкти, відповідні елементам, в яких значення або атрибута `ID`, або атрибута `name` збігається зі значенням аргумента `sname` методу й повертає колекцію, що складається із знайдених об'єктів;

`obj.getelementsbytagname(stagname)` — у всьому контенті елемента, представленого об'єктом `obj`, відшукує всі об'єкти, відповідні елементам, ім'я яких збігається зі значенням аргумента `stagname` методу, повертає колекцію, що складається із знайдених об'єктів.

5.7. Перевірка на наявність контенту й атрибутів

Як, ще один інструмент пошуку елементів можна назвати метод `contains()`, який повертає `true`, якщо об'єкт, заданий аргументом методу, перебуває в контенті елемента, відповідно об'єкту, для якого був викликаний метод, а якщо ні, то повертається `false`.

Далі описані інструменти, що допомагають визначити, чи є у елемента атрибути і контент, а також чи може елемент мати контент:

`attr.specified` — містить `true`, якщо атрибут, відповідний об'єкту `attr`, визначений у документі, і `false` — а якщо ні, то властивість доступна тільки для читання; атрибут може бути визначений безпосередньо в документі;

`obj.haschildnodes()` — повертає `true`, якщо елемент, відповідний об'єкту `obj`, містить елементи або текстові вузли, і `false,` якщо ні;

`obj.canhavechildren` — містить `true`, якщо елемент, відповідний об'єкту `obj`, може мати дочірні вузли, і `false,` а якщо ні;

`obj.canhavehtml` — містить `true`, якщо елемент, відповідний об'єкту `obj`, може містити розмітку HTML, і `false,` якщо ні.

5.8. Створення, видалення і модифікація вузлів

DOM надає методи для модифікації (зміни) дерева документа.

5.8.1. Створення вузлів

Можна створювати вузли двох типів — ***елемент*** і ***текстовий вузол***.

Для створення вузла типу ***елемент*** служить метод об'єкта `document` `createelement(stagname).` Значення обов'язкового аргумента `stagname` — ім'я елемента, а значення, що повертається, — об'єкт, що є представленням цього елемента.

Текстовий вузол створюється за допомогою методу `createtextnode(stext).` Значення необов'язкового аргумента `stext` — текст створюваного вузла, а значення, що повертається, — об'єкт, що є представленням створюваного вузла.

5.8.2. Клонування вузлів

Крім операцій створення вузлів можна використовувати метод клонування вузла `obj.clonenode(bclonechildren)`. Метод створює копію (клон) об'єкта `obj` з усіма його нащадками, якщо значенням необов'язкового аргумента `bclonenode` є `true`, і без нащадків, — а якщо ні, то значенням за замовчуванням для аргумента є `false`.

5.8.3. Копіювання вузлів

Для копіювання всієї сукупності атрибутів, що належать якому-небудь елементу, служить метод `obj.mergeattributes(osource, bid),` де `osource`, - копійований вузол в елемент представлений об'єктом `obj`. При цьому необов'язковий аргумент `bid` указує, чи потрібно копіювати значення атрибута `ID`. Якщо значенням аргумента є `true` (за замовчуванням), то значення атрибута `ID` скопійоване не буде.

5.8.4. Вставка елементів

Основним засобом вставки елемента в конкретне місце дерева є метод `obj.insertadjacentelement(swhere,oelement).`

Метод реалізує вставку елемента, представленого об'єктом `oelement`, у таке місце дерева, яке визначається щодо елемента, представленого об'єктом `obj`, залежно від значення аргумента `swhere`. Цими значеннями можуть бути наступні рядки:

- `beforebegin` — елемент вставляється безпосередньо перед відкриваючим тегом елемента, представленого об'єктом `obj`;
- `afterbegin` — елемент вставляється відразу після відкриваючого тега елемента, представленого об'єктом `obj`;
- `beforeend` — елемент вставляється безпосередньо перед закриваючим тегом елемента, представленого об'єктом `obj`;
- `afterend` — елемент вставляється відразу після закриваючого тега елемента, представленого об'єктом `obj`.

При спробі вставити елемент, представлений уже існуючим об'єктом, вставка зроблена не буде. Значення, що повертається, - вставлений об'єкт.

5.8.5. Вбудовування вузлів у дерево документа

Інструментом вбудовування ***елемента*** або ***текстового вузла*** в дерево документа HTML є метод `obj.appendchild(onode)` який додає вузол `onode` у кінець колекції `childnodes` об'єкта `obj`. При цьому значення, що повертається є `onode`.

Метод `obj.applyelement(oelement,swhere)` додає елемент, представлений об'єктом `oelement`, у дерево й робить його батьком елемента, представленого об'єктом `obj`, якщо значенням необов'язкового аргумента `swhere` є рядок `outside` (значення за замовчуванням), і безпосереднім нащадком, якщо цим значенням є `inside`. В останньому випадку елемент, представлений об'єктом `oelement`, буде єдиним безпосереднім нащадком елемента, представленого об'єктом `obj`, і буде містити всіх його нащадків. Значенням, що повертається, є вбудований у дерево елемент, контент якого після вбудовування в кожному разі буде відрізнятися від контенту елемента, представленого об'єктом `oelement.`

Метод вбудовування вузла в дерево документа `obj.insertbefore(onode,ochild).` Якщо не задане значення

необов'язкового аргумента `ochild`, то метод еквівалентний методу `appendchild()`. Якщо ж це значення задане, то вузол `onode` буде вставлений перед вузлом `ochild`, звідси й назва методу. Значенням, що повертається, є вставлений об'єкт.

5.8.6. Зміна місця розташування вузла

Для зміни місця двох вузлів дерева призначений метод `obj.swapnode(onode)` який міняє місцями вузли `obj` і `onode`. Значенням, що повертається, є об'єкт `obj`.

5.8.7. Створення і зміна атрибутів

Для створення або зміни значення атрибута визначений метод `obj.setattribute(sname,vvalue)`, що створює атрибут з іменем `sname` і значенням `vvalue` для елемента, представленого об'єктом `obj`. Тип значення `vvalue` може бути довільним (рядок, число, булеве значення, об'єкт). Якщо при створенні атрибута виявляється, що атрибут з іменем `sname` уже існує, то відбувається зміна старого його значення на `vvalue`.

5.8.8. Видалення вузлів з дерева документа

Метод для видалення вузлів `obj.removenode(bremovechildren)`. Метод видаляє з дерева елемент, представлений об'єктом `obj`. Якщо значенням необов'язкового аргумента `bremovechildren` є `true`, то видаляються й усі його нащадки. Якщо аргумент має значення `false` (за замовчуванням), то нащадки не видаляються, а стають нащадками його батька. Значенням, що повертається, є об'єкт, відповідний до видаленому вузлу.

Метод `obj.removechiid(onode)` еквівалентний попередньому. Він видаляє першого безпосереднього нащадка елемента, представленого об'єктом `obj`. Безпосередній нащадок при цьому представлений об'єктом `onode` є аргументом методу. Значенням, що повертається, є об'єкт, відповідний видаленому елементу. Якщо вузол є елементом, то весь його контент видаляється разом з ним.

Для видалення одного атрибута призначений метод `obj.removeattribute(sname)`. Він видаляє атрибут з іменем `sname` представленого об'єктом `obj`. При цьому метод повертає значення `true`, якщо видалення пройшло успішно, і `false` — якщо ні.

Для видалення всієї сукупності атрибутів елемента використовується метод `obj.clearattributes()`, де елемент представлений об'єктом `obj`.

5.8.9. Заміна вузлів з дерева документа

Для заміни вузлів типів "***елемент***" і "***текстовий вузол***" використовують наступні методи:

- `obj.replacenode(onewnode)` — заміняє вузол, представлений об'єктом `obj`, на вузол, представлений об'єктом `onewnode`, і повертає об'єкт `obj`;
- `obj.replacechild(onewnode,ochildnode)` — заміняє безпосереднього нащадка об'єкта `ochildnode`, що належить об'єкту `obj`, на вузол, представлений об'єктом `onewnode`. Значення, що повертається є об'єкт `ochildnode`.

5.9. Методи модифікації документа

Об'єкт `document` має два методи для модифікації документа: `write()` і `writeln()`. Ці методи записують у документ свій строковий аргумент. Метод `writeln()` відрізняється від `write()` тим, що додає після рядка свого аргумента символ «переклад на новий рядок».

Недоліком методів є те, що весь контент документа, створений без їхньої допомоги, при першому звертанні до документу повністю знищується.

Тому методи `write()` і `writeln()` використовують або спочатку документа (у контенті елемента SCRIPT, який є першим безпосереднім нащадком елемента BODY), або весь документ повинен бути генерований за допомогою цих методів.

5.10. Обробка контенту елемента

Контент елемента ділять на два види:

внутрішній контент - зміст елемента у звичайному значенні,

зовнішній контент - складається з внутрішнього контенту, а також тега елемента.

У табл. 4 описані властивості, що містять як внутрішній, так і зовнішній контент.

Таблиця 4 - Властивості, значенням яких є контент елемента

Властивість	Опис
innerhtml	Містить внутрішній контент елемента з розміткою HTML
innertext	Містить внутрішній контент елемента без розмітки HTML. Збігається зі значенням innerhtml, з якого вилучена розмітка
outerhtml	Містить внутрішній контент елемента з розміткою HTML. При присвоєнні властивості нового значення елемент, якому належить ця властивість, видаляється з документа й заміняється Html-кодом значенням, що є ним
outertext	Містить зовнішній контент елемента з розміткою HTML. Збігається зі значенням outerhtml, з якого вилучена розмітка. При присвоєнні властивості нового значення елемент, якому належить ця властивість, видаляється з документа й заміняється текстом значенням, що є ним

Щоб одержати значення текстового вузла, що перебуває в безпосередній близькості від закриваючого й відкриваючого тегів елемента, використовують obj.**getadjacenttext**(swhere). Метод повертає текст, місце розташування якого щодо елемента, представленого об'єктом obj, визначається за значенням текстового аргумента swhere. Аргумент може приймати такі значення:

- beforebegin — метод повертає значення текстового вузла, що перебуває безпосередньо перед відкриваючим тегом елемента;
- afterbegin — метод повертає значення текстового вузла, що перебуває відразу після відкриваючого тега елемента;
- beforeend — метод повертає значення текстового вузла, що перебуває безпосередньо перед закриваючим тегом елемента;
- afterend — метод повертає значення текстового вузла, що перебуває відразу після закриваючого тега елемента.

Наступні два методи служать для вставки відповідно Html-коду й тексту безпосередньо перед відкриваючим тегом елемента або відразу після закриваючого.

obj.**insertadjacenthtml**(swhere,shtml),

obj.**insertadjacenttext**(swhere,stext).

При цьому елемент представлений об'єктом obj, контент, що вставляється, — значенням другого аргумента, а місце вставки — значенням першого аргумента. Залежність місця вставки контенту від значення аргумента swhere у цих методах еквівалентна.

Для заміни значення текстового вузла на нове використовують метод obj.**replaceadjacenttext**(swhere,snewtext). Елемент - це об'єкт obj, текстовий вузол, значення якого потрібно змінити, визначає аргумент swhere, а текст для заміни — аргумент snewtext. Аргумент swhere приймає ті ж значення, як у методі getadjacenttext(), місце вставки контенту те саме.

Для «розбивки» текстового вузла на дві частини використовують метод otextnode.**splittext**(iindex). Об'єкт otextnode «розбивають» у такий спосіб:

Якщо необов'язковий аргумент iindex не заданий, метод повертає об'єкт, відповідний текстовому вузлу без тексту.

Якщо iindex заданий, то метод повертає об'єкт, відповідний текстовому вузлу, змістом якого є результат вирізання з текстового значення otextnode тексту, починаючи з позиції iindex і до кінця. Значення otextnode після розбивки є частина тексту, що залишилася, тобто результат вирізання тексту, починаючи з позиції 0, і до позиції, заданої iindex.

Нагадаємо, значення текстового вузла можна одержати за допомогою властивості nodevalue.

5.11. Обробка таблиць

Розглянемо методи й властивості для обробки таблиць, які створені за допомогою елемента TABLE. Методи й властивості належать об'єкту, відповідному елементам: TABLE, TBODY, THEAD, TFOOT, TD і TR. Властивості доступні тільки для читання.

Таблиця 5 - Властивості для обробки таблиць

Властивість	Значення
caption	Об'єкт CAPTION, що представляє заголовок таблиці
cellindex	Позиція гнізда в рядку (TR) таблиці
rowlndex	Позиція рядка в таблиці

sectionrowlndex	Позиція рядка в таблиці або в якій-небудь з трьох її секцій: TBODY, TFOOT, THEAD
tfoot	Об'єкт, що відповідає секції TFOOT таблиці
thead	Об'єкт, що відповідає секцій THEAD таблиці

Таблиця 6 - Методи для обробки таблиць

Метод	Опис
createcaption()	Створює заголовок (CAPTION) таблиці
createtfoot()	Створює секцію TFOOT таблиці
createthead()	Створює секцію THEAD таблиці
deletecaption()	Видаляє заголовок (CAPTION) таблиці
deletecell()	Видаляє гніздо таблиці
deleterow()	Видаляє рядок таблиці
deletetfoot()	Видаляє секцію TFOOT таблиці
deletethead()	Видаляє секцію THEAD таблиці
insertcell()	Вставляє гніздо в таблицю
insertrow()	Вставляє рядок у таблицю

5.12. Обробка URL

Обробці URL приділяється велика увага, тому що механізм практично реалізує основне достоїнство Інтернет як інформаційного сховища. Практично потужність HTML полягає в наявності тега <A> … </A>, параметром якого є HREF – гіперпосилання.

Таблиця 7 - Властивості для обробки URL

Властивість	Значення
Hash	Частина Url-адреси, що випливає після символу #

Host	Частина Url-адреси, відповідна імені комп'ютера й номеру порту
hostname	Частина Url-адреси, відповідна імені комп'ютера
Href	Для елемента BASE — базовий URL, для елементів A, AREA, LINK — URL адресованого ресурсу, для об'єкта location — URL відповідного до цього об'єкта документа, для об'єкта stylesheet — URL стильової таблиці
nameprop	Ім'я файла, що міститься у значеннях властивостей href або src елементів A або IMG
Port	Частина Url-адреси, відповідна до номера порту
protocol	Частина Url-адреси, відповідна до протоколу
referrer	URL комп'ютера, з якого був зроблений запит поточного документа; тільки для читання
search	Частина Url-адреси, що випливає після символу ?
URL	URL поточного документа
Urlunencoded	Декодований URL поточного документа (коди символів, неприпустимих в Url-адресі, наприклад пробілів, замінені самими символами); тільки для читання
A.urn	URN поточного документа

Властивості hash, host, hostname, port, protocol, search мають елементи A и AREA, а також об'єкт location, а властивостями referrer, URL і Urlunencoded — об'єкт document.

5.13. Обробка таблиць стилів CSS

Для оптимізації роботи при проектуванні й створенні Html-документів розроблена специфікація Cascading Style Sheets (каскадна таблиця стилів). Вона дозволяє зібрати воєдино всі установки з оформлення контенту документа, а в контент включати інструмент посилання на описаний стиль оформлення. Це дозволяє уніфікувати стилістичне оформлення комплексу документів HTML, об'єднаних єдиною тематикою.

Таблиця 8 - Методи для обробки таблиць стилів (CSS).

Метод	Опис
addimport(URL)	Додає до списку імпортованих таблиць стилів ще одну, URL якої представлений аргументом методу
addrule(Selector, property_value)	Для стильової таблиці створює правило, селектор якого представлений першим, а пари властивість-значення — другим аргументом методу
createstylesheet(URL)	Створює таблицю стилів для документа, підключену за допомогою елемента LINK або вбудовану в нього за допомогою елемента STYLE, залежно від значення аргумента методу
removerule(index)	Видаляє правило з номером, представленим аргументом методу

Методи addimport(), addrule() і removerule() належать об'єкту stylesheet, а метод createstylesheet() — об'єкту document.

5.14. Об'єкт event

Будь-які дії користувача в сеансі роботи породжують особливий вид інформації, який у системі інтерпретується як «***подія***». На рівні операційної системи реалізований так званий «***віконний інтерфейс***», коли для роботи додатку операційна система виділяє обчислювальні й апаратні ресурси. Для користувача ці ресурси відображуються у вигляді вікна додатку. У рамках об'єктно-орієнтованого підходу в системі визначено об'єкт «***вікно***» - window, а всі дії в системі формалізують об'єктом «***подія***» - event. Об'єкт window має властивість подія event, яка динамічно оновлюється. Ця властивість є об'єктом, що відповідає згенерированій події. Об'єкт event передається обробнику подій. Властивості об'єкта event містять дані про подію. Деякі властивості є загальними для всіх типів подій. До них ставляться координати курсору миші у вікні браузера. Кожний Html-елемент може бути джерелом багатьох подій. Розглянемо деякі події, які можуть генерувати всі елементи сторінки.

5.14.1. Події миші

У сценаріях надається можливість обробки дій користувача, які він виконує за допомогою маніпулятора «миша». Практично подій від маніпуляції мишею не так вже і багато, ось деякі з них:

`Mouseover` - наведення покажчика миші на об'єкт документа HTML;
`Click` - клацання мишею на певному об'єкті;
`Dblclick` - подвійне клацання;
`Mouseout` - видалення курсору з області об'єкта.

Подія `Mousemove` викликається при переміщенні миші. Для більш тонкого аналізу ситуації використовуються події:

`Mousedown` - натискання кнопки миші;
`Mouseup` - кнопка відпускається.

Властивість `srcelement` визначає джерело події, наприклад

```
var Елемент = window.event.srcelement
```

Далі наведено приклад документа HTML. Він містить сценарій (програмний код) мовою програмування Javascript. Сценарій оформлений за допомогою тегу <SCRIPT> і реалізує функцію «перетаскування зображення».

Сформуйте документ у файл *.html і відкрийте у вікні браузера. Натиснувши клавішу на будь-якій кредитній картці, можна перетягнути її в інше місце вікна, у тому числі й за його межі.

```
<HTML>
  <HEAD>
   <TITLE>Перетягування зображень</TITLE>
   <META http-equiv="Content-Type" content="text/html; charset=utf-8">
   <SCRIPT LANGUAGE="Javascript">
<!-- //
var Credut=""
function ButDown()                                  // визначення функції
{
  if (window.event.srcelement.tagname=="IMG")          // умовний оператор
  {
   Credut=window.event.srcelement;
     Credut.style.left=window.event.clientx-30;
     Credut.style.top=window.event.clienty-19;
  }
}
function ButUp()
{
  Credut=""
}
function Moving()
```

```
{
  if (Credut!="")
  {
    Credut.style.left=window.event.clientx-30;
    Credut.style.top=window.event.clienty-19;
  }
  event.cancelbubble=true;
  event.returnvalue=false;
}
//-->
   </SCRIPT>
  </HEAD>
  <BODY onmousedown="ButDown()" background="../pic/blue.gif" onmousemove="Moving()"
       onmouseup="ButUp()">
   <IMG src="../pic/ONTBAMX.gif" style="position:absolute; left: 50; top: 50">
   <IMG src="../pic/ONTBDSC.gif" style="position:absolute; left:150; top: 50">
   <IMG src="../pic/ONTBMC.gif"  style="position:absolute; left: 50; top:150">
   <IMG src="../pic/ONTBVSA.gif" style="position:absolute; left:150; top:150">
   <IMG src="../pic/ONTBJCB.gif" style="position:absolute; left:100; top:100">
  </BODY>
</HTML>
```

НАВЧАЛЬНЕ ВИДАННЯ

«Комп'ютерна техніка та програмне забезпечення». Конспект лекцій для студентів 2 курсу денної форми навчання освітньо-кваліфікаційного рівня бакалавр у галузі знань 0507 - «Електротехніка та електромеханіка» за напрямом підготовки 6.050701 – «Електротехніка та електротехнології», спеціальності 6.090600 - «Електротехнічні системи електроспоживання».

Укладачі: Ігор Леонідович Яковицький

Відповідальний за випуск: М.І. Самойленко

Редактор: М.З. Аляб'єв

Комп'ютерний набір і верстка: І.Л.Яковицький

План 2009, поз. 191Л

Підп. до друку 10.11.09	Формат 60х84 1 /16	Папір офісний
Друк на ризографі.	Умовн.-друк. арк.4,2	Обл.-вид. арк.4,5
Замовл №	Тираж 50 прим.	

61002, Харків, ХНАМГ, вул. Революції, 12

Сектор оперативної поліграфії ЦНІТ ХНАМГ
61002, Харків, вул. Революції, 12