МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПРИКАРПАТСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАСИЛЯ СТЕФАНИКА

Кафедра інформаційних технологій

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни

«Основи Інтернет»

для студентів напряму підготовки

«Прикладна математика»

ІВАНО-ФРАНКІВСЬК – 2015 РІК

УДК 004.43
ББК 32.973.2-018
Л17

Рекомендовано до друку вченою радою факультету математики та інформатики Прикарпатського національного університету імені Василя Стефаника, (протокол №__ від 08 ггрудня 2015 р.)

Рецензенти:

Козленко М.І. – кандидат технічних наук, доцент кафедри інформаційних технологій факультету математики та інформатики Прикарпатського національного університету імені Василя Стефаника

Ткачук В.М. – кандидат фізико-математичних наук, доцент кафедри інформаційних технологій факультету математики та інформатики Прикарпатського національного університету імені Василя Стефаника

Лазарович І. М.

Л17 Конспект лекцій з дисципліни «Основи Інтернет» для студентів напряму підготовки «Прикладна математика» / І. М. Лазарович – Івано-Франківськ: Видавництво Прикарпатського національного університету імені Василя Стефаника, 2015. – 87 с.

Наведено основні поняття організації мережі Інтернет, розглянуто методологію створення веб-сторінок засобами мови розмітки гіпертекстових документів HTML та таблиць стилів CSS. Представлено синтаксис основних тегів і відомості та рекомендації щодо застосуваняння блокової та фреймової верстки документів. В якості програмування на стороні клієнта розглядається мова JavaScript, що використовується у складі HTML-сторінок для надання їм інтерактивності за рахунок обробки подій. Наведено опис основних операторів та функцій, що супроводжується прикладами їх практичного застосування.

Конспект лекцій розроблено на основі вимог навчальної програми з дисципліни «Основи Інтернет» для підготовки до фахівців з напряму «Прикладна математика». Наведений матеріал також може бути використаний для студентів інших напрямів підготовки, які вивчають веб-програмування і мають базові навики і знання з інформаційних технологій.

УДК 004.43
ББК 32.973.2-018

© Лазарович І. М., 2015
© Видавництво Прикарпатського національного університету імені Василя Стефаника, 2015

ЗМІСТ

ПЕРЕДМОВА

Інтернет (від англ. Internet) - всесвітня система взаємосполучених комп'ютерних мереж і призначена для обіну інформацією. Інтернет складається з мільйонів локальних і глобальних приватних, публічних, академічних, ділових і урядових мереж, пов'язаних між собою з використанням різноманітних дротових, оптичних і бездротових технологій. Інтернет становить фізичну основу для розміщення величезної кількості інформаційних ресурсів і послуг.

Ресурси мережі інтернет забезпечують ряд функцій – спілкування, розваги, навчання, шопінг, комерція. З іншого боку для розміщення інформації в інтернет потрібно володіти знаннями по створенню, дизайну і обслуговуванню веб ресурсів.

В межах даної роботи розглянуто основи функціонування всесвітньої мережі, принципи адресації, технологію веб-дизайну з використанням мови розмітки гіпертекстових документів HTML, а також таблиць стилів CSS. Також коротко розглянуто основи програмування на стороні клієнта з використанням мови JavaScript, що використовується у складі HTML-сторінок для збільшення їх функціональності та можливостей взаємодії з користувачами. Подані теоретичні відомості супроводжуються прикладами застосування тегів та команд, що дозволяє більш ефективно засвоїти наведений матеріал.

Конспект лекцій призначений для підготовки до виконання лабораторних робіт з дисципліни «Основи Інтернет» студентами напряму підготовки «Прикладна математика». Також наведений матеріал може бути використаний студентами інших напрямів підготовки, які вивчають веб-дизайн, для розуміння та засвоєння поданих відомостей потрібно володіти базовими навиками програмування.

Тема 1. Основні поняття Інтернет та гіпертекстових документів

1.1 Загальні відомості про мережу Інтернет

З моменту появи ЕОМ виникло питання про передачу даних між окремими комп'ютерами і раціональний розподіл ресурсів ЕОМ.

З розвитком апаратної і програмної бази комп'ютерів, удосконалювалися і мережеві технології. Спочатку були створені системи передачі даних спочатку в комерційних, військових і наукових цілях, потім сфера застосування мереж розширилася.

І сьогодні використання комп'ютерних мереж є невід'ємною частиною нашого життя; сфера їх застосування охоплює всі сфери людської діяльності.

Під комп'ютерною мережею розумітимемо будь-яку множину ЕОМ, зв'язаних між собою засобами передачі даних (засобами телекомунікацій).

Історія мережі Internet

У1961 роціDefence Advanced research Agensy (DArPA) за завданням міністерства оборони США приступило до проекту зі створення експериментальної мережі передачі пакетів. Ця мережа, названа ARPANET, була призначена спочатку для вивчення методів забезпечення надійного зв'язку між комп'ютерами різних типів.

З метою підвищення загальної надійності мережі її управління було децентралізовано, тобто всі її вузли абсолютно рівноправні. Це означає, що кожен вузол мережі забезпечує як прийом/передачу своїх повідомлень, так і переадресацію (маршрутизацію) повідомлень, що приходять від інших вузлів.

Як завжди, самі передові технології передбачалося використовувати у військових цілях. Принципи, закладені в знов створену мережу, за задумом творців дозволяли використовувати нову Мережу для управління країною після ядерного удару.

Недаремно головна вимога, що пред'являлася до Мережі, — надійна передача повідомлень за будь-яких змін умов передачі. Мережа повинна залишатися працездатною, навіть якщо велика частина її вузлів виведена з ладу. Для цього використовувалася нова технологія принципу комутації пакетів. Достоїнство даного методу у високій швидкості, надійності і гнучкості передачі.

Багато методів передачі даних через модеми було розроблено в ARPANET. Тоді ж були розроблені і протоколи передачі даних в мережі — TCP/IP.

TCP/IP — це безліч комунікаційних протоколів, які визначають, як комп'ютери різних типів можуть спілкуватися між собою.

Експеримент з ARPANET був настільки успішний, що багато організацій захотіли увійти до неї з метою використання для щоденної

передачі даних. В1975 році ARPANET перетворилася з експериментальної мережі в робочу. Відповідальність за адміністрування мережі узяло на себе Defence Communication Agency (DCA), в даний час зване Defence Information Systems Agency (DISA). Але розвиток ARPANET на цьому не зупинився. Протоколи TCP/IP продовжували розвиватися і удосконалюватися.

У1983 році вийшов перший стандарт для протоколів TCP/IP, що увійшов до Military Standarts (MIL STD), тобто у військові стандарти, і всі, хто працював в мережі, зобов'язано було перейти до цих нових протоколів. Для полегшення цього переходу DArPA звернулася із пропозицією до керівників фірми Berkley Software Design — упровадити протоколи TCP/IP до Berkeley (BSD) UNIX. З цього і почався союз UNIX і TCP/IP. Через деякий час TCP/IP був адаптований в звичайний, тобто в загальнодоступний стандарт, і термін Internet увійшов до загального вживання.

У1983 році з ARPANET виділилася MILNET, яка почала відноситися до Defence Data Network (DDN) міністерства оборони США. Термін Internet почав використовуватися для позначення єдиної мережі: MILNET плюс ARPANET. І хоча в 1991 році ARPANET припинила своє існування, мережа Internet існує, її розміри набагато перевищують первинні, оскільки вона об'єднала безліч мереж у всьому світі.

Число хостів, підключених до мережі Internet, з4 комп'ютерів в1969 році виросло до5 мільйонів у2001-у. Хостом в мережі Internet називають комп'ютери, що працюють в багатозадачній операційній системі (Unix, VMS), підтримують протоколи TCP\IP і надають користувачам які-небудь мережеві послуги.

Протоколи мережі Internet

Основне, що відрізняє Internet від інших мереж — це її протоколи — TCP/IP. Взагалі, термін TCP/IP зазвичай означає все, що пов'язане з протоколами взаємодіїміж комп'ютерами в Internet. Він охоплює ціле сімейство протоколів, прикладні програми, і навіть саму мережу. TCP/IP — це технологія міжмережевої взаємодії, технологія internet. Мережа, яка використовує технологію internet, називається "internet". Якщо мова йде про глобальній мережі, об'єднуючій безліч мереж з технологією internet, то її називають Internet.

Свою назву протокол TCP/IP отримав від двох комунікаційних протоколів(або протоколів зв'язку). Це Transmission Control Protocol (TCP) і Internet Protocol (IP).

Не дивлячись на те, що в мережі Internet використовується велике число інших протоколів, мережу Internet часто називають TCP/IP-мережею, оскільки ці два протоколи, безумовно, є найважливішими.

Міжнародною організацією стандартів затверджені певні вимоги до організації взаємодії між системами мережі. Ці вимоги отримали назву OSI (Open System Interconnection) — "еталонна модель взаємодії відкритих систем"(1984р).

Згідно вимогам еталонної моделі, кожна система мережі повинна здійснювати взаємодію шляхом передачі кадру даних. Згідно моделі OSI освіта і передача кадру здійснюється з допомогою 7-х послідовних дій, що отримали назву"рівень обробки".

Як і у всякій іншій мережі в Internet існує 7 рівнів взаємодії між комп'ютерами: фізичний, логічний, мережевий, транспортний, рівень сеансів зв'язку, представницький і прикладний рівень. Відповідно до кожного рівня взаємодії відповідає набір протоколів(тобто правил взаємодії).

Протоколи фізичного рівня визначають вигляд і характеристики ліній зв'язку між комп'ютерами. У Internet використовуються практично всі відомі в даний час способи зв'язку: від простого дроту(витаючи пара) до волоконно-оптичних ліній зв'язку.

Для кожного типу ліній зв'язку розроблений відповідний протокол логічного рівня, що займається управлінням передачі інформації по каналу. До протоколів логічного рівня для телефонних ліній відносяться протоколи SLIP (Serial Line Interface Protocol) і PPP (Point to Point Protocol). Для зв'язку по кабелю локальної мережі — це пакетні драйвери плат ЛОМ (локальні обчислювальні мережі).

Протоколи мережевого рівня відповідають за передачу даних між пристроями в різних мережах, тобто займаються маршрутизацією пакетів в мережі. До протоколів мережевого рівня належать IP (Internet Protocol) iARP (Address resolution Protocol).

Протоколи транспортного рівня управляють передачею даних з однієї програми в іншу. До протоколів транспортного рівня належать TCP (Transmission Control Protocol) і UDP (User Datagram Protocol).

Протоколи рівня сеансів зв'язку відповідають за установку, підтримку і знищення відповідних каналів. У Internet цим займаються вже згадані TCP і UDP протоколи, а також протокол UUCP (Unix to Unix Copy Protocol).

Протоколи представницького рівня займаються обслуговуванням прикладних програм. До таких програм належать програми, що запускаються, наприклад, на Unix-сервері, для надання різних послуг абонентам. До таких програм відносяться:

− telnet-сервер;
− FTP-сервер;
− Gopher-сервер;
− NFS-сервер;
− NNTP (Net News Transfer Protocol);
− SMTP (Simple Mail Transfer Protocol);
− POP2 iPOP3 (Post Office Protocol) і так далі.

До протоколів прикладного рівня відносяться мережеві послуги і програми їх надання.

1.2 Адресація в Internet

Internet— це мережа, що складається з рівноправних і незалежних вузлів, об'єднаних між собою каналами зв'язку.

Вузол Internet в широкому сенсі — будь-який обчислювальний пристрій, що включений в мережу і має свою унікальну IP-адресу. Так, вузлом стане і ваш персональний комп'ютер, після того, як він встановить зв'язок з провайдером.

У більш вузькому сенсі вузол Internet — це могутній комп'ютер-сервер. Часто вузлом є достатньо крупна локальна мережа, до якої можуть бути включені десятки комп'ютерів. Вузол оснащений необхідним комунікаційним устаткуванням, яке дозволяє експлуатувати канали зв'язку. Як канали зв'язки можуть використовуватися звичайні і оптоволоконні кабелі, радіоканали і канали супутникового зв'язку.

Фізичні адреси

Internet утворює павутину, в якій між двома будь-якими вузлами є зв'язок або прямим каналом або через ряд проміжних вузлів.

Вузли обмінюються між собою повідомленнями. Будь-яке повідомлення розби-вають на пакети і відправляють доступними каналами зв'язку.

Щоб пакети(і повідомлення в цілому) могли дійти до мети, кожен з них містить

адресу призначення. Для цього будь-який вузолInternet має свою унікальнаIP-адресу(фізична адреса), яка складається з чотирьох чисел в діапазоні від0 до255.

Їх записують через крапку. Порядок цих чисел має значення. Наприклад:

193.125.5.38; 146.23.57.255. Цей протокол називають IP4.

Неважко порахувати, скільки ж всього може бути вузлів вInternet: зведемо 256 в четвертую ступінь і отримаємо більше4-х мільярдів адрес.

Багато це чи мало? Двадцять років тому це здавалося багато, навіть дуже, тому що ніхто не припускав, що до Internet захоче підключитися майже кожен цивілізований житель Землі. В даний час запас можливих адрес Internet стрімко скорочується. Вихід − в очікуваному ухваленні нового протоколу IP6. У ньому будуть використовувати адреси, що складаються не з чотирьох, а з шести чисел. Це збільшить кількість адрес в 65 тисяч разів. Залишається сподіватися, що цього запасу вистачить надовго.

Слід відзначити, що кожен вузол Internet має свою унікальну адресу, тобто в Internet немає двох вузлів, що мають однакову IP-адресу. Для цього існує координаційний центр INTERNIC (Internet Network Information Center).

Кожен вузол, бажаючи послати повідомлення іншого вузла, розбиває його на пакети фіксованої довжини. Всі пакети забезпечуються адресою

одержувача, а також адресою відправника. Підготовлені таким чином пакети прямують в доступні канали зв'язку до інших вузлів.

Щоб знати, в якому напрямі передавати"транзитний" пакет, на кожному вузлі є так звана таблиця маршрутизації, в якій для кожної адреси або групи адрес вказаний канал, яким слід передавати пакети.

При отриманні будь-якого пакету вузол аналізує адреса одержувача, і якщо він співпадає з його власною адресою, то пакет приймається, інакше — відправляється далі. Отримані пакети, що відносяться до одного і того ж повідомлення, накопичуються. Як тільки всі пакети одного повідомлення отримані, вони"склеюються" і одержувачеві доставляють все повідомлення.

Під адресою взагалі надалі розумітимемо як фізичну адресу (IP-адрес), так і символьну адресу(ім'я).

Символьні адреси

Символьні адреси замінюють фізичні для зручності користувачів. Використовуючи www або електронну пошту, фізичні адреси, тобто набір цифр, указувати украй скрутно. Для цього більш підходять адреси символічні, багато з яких легко запам'ятати.

Якщо фізична адреса — набір чисел, розділених крапкою, то символьна адреса — набір слів, також розділених крапкою. Такі імена читаються праворуч-ліворуч.

Кожне слово в символьному імені це так званий домен.

Найзагальніший домен— територіальний або домен верхнього рівня. Він указує на країну, в якій знаходиться вузол. Територіальний домен.ru закріплений за російськими вузлами, .ua — за українськими. Аналогічні домени закріплені і за рештою держав.

Розглянемо деякі приклади доменів верхнього рівня:

− com — комерційна організація;

− net — мережа;

− org — некомерційна організація;

− edu — освітня установа;

− gov — урядова установа.

Наступні за територіальним (праворуч-ліворуч) домени указують на вузол і його підмережі, а також окремі сервери.

DNS (Domain Name Service)

Звичайно, не існує якого-небудь строгого зв'язку між символічними і фізичними адресами вузлів Internet. Завдання перекладу одних на інших покладають на спеціальну службу — DNS.

DNS (Domain Name Service) — розподілена на вузлах Internet база даних про відповідність фізичних і символічних адрес. Таким чином, служба DNS — своєрідний довідник.

За забезпечення перетворення символічних імен у фізичних і навпаки відповідає провайдер, у якого є сервер DNS, тобто спеціальний комп'ютер

або комп'ютерна система, оброблювальна запити користувача. Для користувача робота DNS абсолютно прозора. Один раз настроївши правильно конфігурацію DNS на своєму комп'ютері, йому досить просто указувати символічні адреси серверів, з якими слід встановити зв'язок.

1.3 Сервіс WWW та основні поняття про веб-сторінки.

Інтернет — це глобальна комп'ютерна мережа, що об'єднує сотні мільйонів комп'ютерів в загальний інформаційний простір. Інтернет представляє свою інфраструктуру для прикладних сервісів різного призначення, найпопулярнішою з яких є Всесвітня Павутина — World Wide Web(www).

World Wide Web (www, web, русск.: веб, Всесвітня Павутина) — розподілена інформаційна система, що надає доступ до гіпертекстових документів по протоколу HTTP.

WWW — мережева технологія прикладного рівня стека TCP/IP, побудована на клієнт-серверній архітектурі і використовуюча інфраструктуру Інтернет для взаємодії між сервером і клієнтом (рисунок 1).

Сервери www (веб-сервери) — це сховища гіпертекстової (у загальному випадку) інформації, керовані спеціальним програмним забезпеченням.

Документи, представлені у вигляді гіпертексту називаються веб-сторінками. Декілька веб-сторінок, об'єднаних загальною тематикою, оформленням, пов'язаних гіпертекстовими посиланнями і таких, що зазвичай знаходяться на одному і тому ж веб-сервері, називаються веб-сайтом.

Для завантаження і перегляду інформації з веб-сайтів використовуються спеціальні програми — браузери, здатні обробляти гипертектовую розмітку і відображати вміст веб-сторінок.

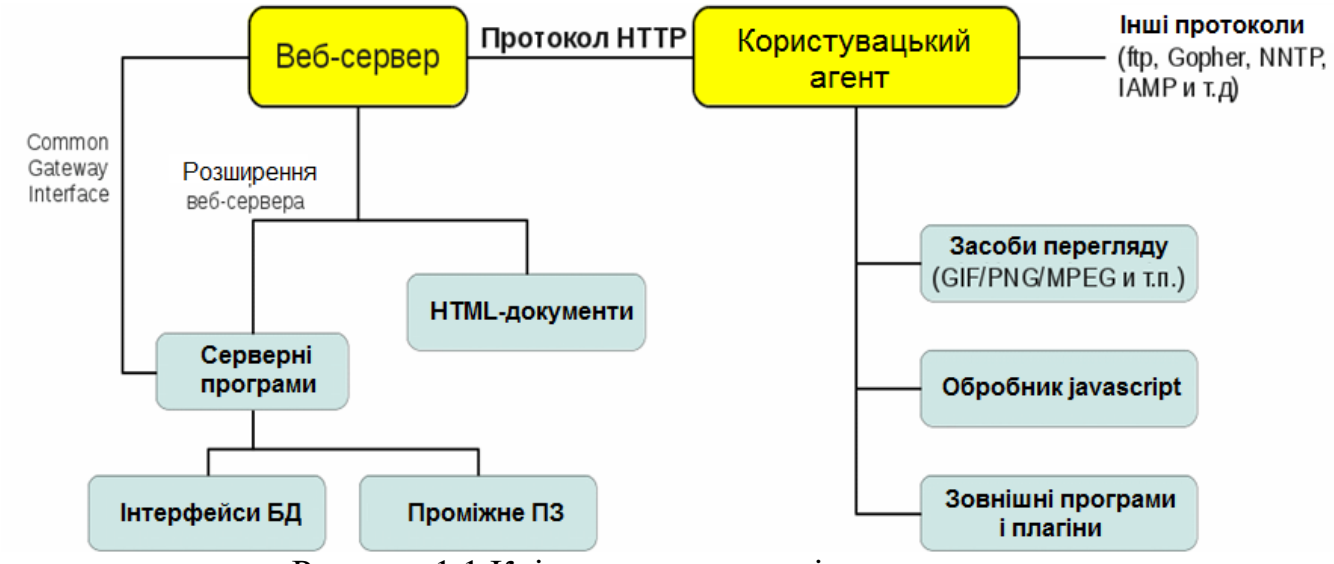


Рисунок 1.1 Клієнт-серверна архітектура www

У основі www — взаємодія між веб-сервером і браузерами по протоколу HTTP (HyperText Transfer Protocol).

Веб-сервер — це програма, запущена на мережевому комп'ютері і очікуюча клієнтські запити по протоколу HTTP. Веб-сервер приймає запити і повертає відповіді, зазвичай разом з HTML-сторінкою, зображенням, файлом, медіа-потоком або іншими даними. Веб-сервери — основа Всесвітньої павутини. З розширенням спектру мережевих сервісів веб-сервери все частіше використовуються як шлюзи для серверів додатків або самі представляють такі функції (наприклад, Apache Tomcat).

Браузер може звернутися до веб-сервера по доменному імені або по ip - адресу, передаючи в запиті ідентифікатор необхідного ресурсу. Отримавши запит від клієнта, сервер знаходить відповідний ресурс на локальному облаштуванні зберігання і відправляє його як відповідь. Браузер приймає відповідь і обробляє її відповідним чином, залежно від типу ресурсу (відображає гіпертекст, показує зображення, зберігає отримані файли і тому подібне).

Основний тип ресурсів Всесвітньої павутини — гіпертекстові сторінки. HTML (HyperText Markup Language) — мова розмітки гіпертексту, призначена для створення веб-сторінок.

HTML представляє прості правила оформлення і компактний набір структурних і семантичних елементів розмітки (тегів), які дозволяють створювати веб-сторінки різної складності. HTML дозволяє описувати спосіб представлення логічних частин документу (заголовки, абзаци, списки і так далі). Спочатку мова HTML була задумана і створена як засіб структуризації і форматування документів без прив'язки до засобів відображення. У ідеалі, гіпертекстовий документ повинен однаково виглядати на різних пристроях (монітор ПЕВМ, екран ПДА або мобільного телефону, принтер, медіа-проектор і тому подібне).

HTML не є мовою програмування, але веб-сторінки можуть утримувати вбудовані або завантажувані програми на скриптових мовах (в першу чергу Javascript) і програми-аплети на мові Java. На сьогоднішній день актуальними версіями HTML є 4.01 та 5.

Загальна структура HTML-документу.

Документ HTML 4.01 складається з трьох частин (рисунок 1.2):

Декларація типу документу (англ. Document type declaration, Doctype), на качану документу, в якій визначається тип документу (DTD). Наявність секції DOCTYPE дозволяє вказати браузеру, який тип документу йому належить розбирати, т.е, які вимоги треба виконувати при обробці гіпертексту.

Приклади DOCTYPE :

- <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http ://www.w 3.org/TR/html4/frameset.dtd"> - гіпертекстовий документ у форматі HTML 4.01, що містить фрейми.
- <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN""http ://www.w 3.org/TR/html4/strict.dtd"> - гіпертекстовий документ у форматі

HTML 4.01 із строгим синтаксисом (тобто не використані застарілі і не рекомендовані теги).

- <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http ://www.w 3.org/TR/html4/loose.dtd"> - гіпертекстовий документ у форматі HTML 4.01 з нестрогим («перехідним») синтаксисом (тобто використані застарілі або не рекомендовані теги і атрибути).

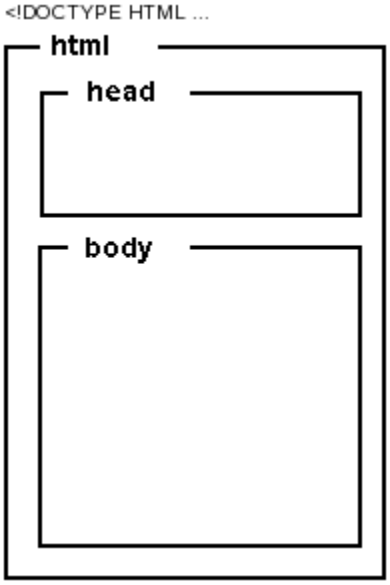


Рисунок 1.2 Загальна структура веб-сторінки

Шапка документу (знаходиться в межах елементу head), в якій записано загальні технічні відомості або додаткова інформація про документ, яка не відтворюється безпосередньо в браузері;

Тіло документу (може знаходитися в елементах body або frameset), в якому міститься основна інформація документа.

Нижче наведено приклад загальної структури HTML-документу:

```
<!doctype HTML public "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
   <head>
      <title>Мій перший HTML-документ</title>
   </head>
   <body>
      Hello world!
   </body>
</html>
```

Тег (html -тег, тег розмітки) — символьна послідовність, що управляє, яка задає спосіб відображення гіпертекстової інформації.

HTML -тег складається з імені, за яким може йти необов'язковий список атрибутів. Увесь тег (разом з атрибутами) полягає в кутові дужки <>:

<ім'я_тега [атрибути]>

Як правило, теги є парними і складаються з початкового і кінцевого тегів, між якими і поміщається інформація. Ім'я кінцевого тега співпадає з ім'ям початкового, але перед ім'ям кінцевого тега ставиться коса риска / (<html>...</html>). Кінцеві теги ніколи не містять атрибутів. Деякі теги не мають кінцевого елементу, наприклад тег <img>. Регістр символів для тегів не має значення.

Атрибути — це пари виду «властивість = значення», що уточнюють представлення відповідного тега :

<тег атрибут="значення">...</тег>

Атрибути вказують в початковому тегу, декілька атрибутів розділяють одним або декількома пропусками, табуляцією або символами кінця рядка. Значення атрибуту, якщо таке є, йде за знаком рівності, що стоїть після імені атрибуту. Порядок запису атрибутів в тегу не важливий. Якщо значення атрибуту — одне слово або число, то його можна просто вказати після знаку рівності, не виділяючи додатково. Усі інші значення необхідно брати в лапок, особливо якщо вони містять декілька розділених пропусками слів.

Примітка: Незважаючи на необов'язковість лапок, їх все ж варто завжди використати.

Атрибути можуть бути обов'язковими і не обов'язковими. Необов'язкові атрибути можуть бути опущені, тоді для тега застосовується значення цього атрибуту за умовчанням. Якщо не вказаний обов'язковий атрибут, то вміст тега швидше за все буде відображений неправильно.

Короткий список деяких часто використовуваних атрибутів і їх можливих значень :

style="опис_стилів" — локальні стилі

src="адреса" — адреса (URI) джерела даних (наприклад картинки або скрипта)

align="left|center|right|justify" — вирівнювання, за умовчанням left (по лівому краю)

width="число" — ширина елементу (у пікселях, списах, поинтах та ін.)

height="число" — висота елементу (у пікселях, списах, поинтах та ін.)

href="адреса" — гіперпосилання, адреса (URI) на який буде виконана перехід

name="ім'я" — ім'я елементу

id="ідентифікатор" — унікальний (в межах веб-сторінки) ідентифікатор елементу

size="число" — розмір елементу

class="ім'я_класу" — ім'я класу у вбудованій або пов'язаній таблиці стилів

title="рядок" — назва елементу

alt="рядок" — альтернативний текст

Контрольні питання до теми 1

1. Розкажіть історію появи мережі Internet.
2. Завдяки чому забезпечується коректна робота ресурсів Internet на різних пристроях, різних операційних системах?
3. Назвіть основні протоколи мережі Internet.
4. Що таке семирівнева модель OSI?
5. Які є види адресації в Internet?
6. Які недоліки адресації по IPv4?
7. В чому суть сервісу WWW?
8. Які складові сервісу WWW?
9. Що таке гіпертекст?
10. З яких частин складається HTML-документ?
11. На що впливає секція DOCTYPE?
12. Що таке тег і атрибут, різниця між цими поняттями?
13. Загальний формат тегів.
14. Назвіть основні атрибути тегів?

Тема 2. Форматування тексту і таблиць у Web-сторінках, робота з фреймами

2.1 Можливості загального форматування веб-сторінки

Тег <body> окрім своєї основної функції, також застосовується для загального форматування HTML -сторінки.

Ось основні атрибути тегу <body>:

- alink — Встановлює колір активного посилання.
- background — Задає фоновий рисунок на веб-сторінці.
- bgcolor — Колір фону веб-сторінки.
- bgproperties — Визначає, прокручувати фон спільно з текстом або ні.
- bottommargin — Відступ від нижнього краю вікна браузеру до контенту.
- leftmargin - Відступ по горизонталі від лівого краю вікна браузеру до контенту.
- link — Колір посилань на веб-сторінці.
- rightmargin - Відступ від правого краю вікна браузеру до контенту.
- scroll - Встановлює, відображати смуги прокрутки або ні.
- text — Колір тексту в документі.
- topmargin — Відступ від верхнього краю вікна браузеру до контенту.
- vlink — Колір відвіданих посилань.

```
<html>
<head>
<title>Мій перший сайт</title>
</head>
<body link="#ff9900" vlink="#336699"
    alink= "#ff6699">
    Привіт всім! А це мій перший
    <a href="http://google.com.ua">сайт</a>.
    З кольоровими <a href="#">посиланнями</a>
</body>
</html>
```

Розглянемо як це виглядає на прикладі.

Як ви помітили, колір вказується в незвичній для вас формі, в цьому прикладі колір вказаний в шістнадцятковому значенні. На початковому етапі замість використання таких страшних позначень кольору(#FF9900, #336699, #FFFFFF) можна обмежиться назвою кольору англійською мовою(black, brown, red, darkred, yellow, green, blue, white і так далі). Проте рано чи пізно вам доведеться впустити у своє життя позначення кольорів в HEX. А може вам більше сподобається CMYK або HSV. Для того, щоб визначиться що вам більше до душі пропоную заглянути в wiki. А я у свою чергу хочу поділитися

з вами програмою ColorPix, яка допоможе вам упізнати код будь-якого вподобаного вам кольору.

Якщо вам захочеться в якості фону використати зображення, то атрибут background вказує шлях до зображення:

```
<body background="bg.jpg" bgproperties="fixed">
```

2.2 Основні теги форматування тексту веб-сторінки

Параграфи.

Перш ніж вивчати теги форматування HTML, подивимося як введений текст відобразиться, якщо не будуть застосовані ніякі теги окрім тегів <html> і <body>. Наступний приклад демонструє такий документ HTML

```
<html>
<body>
Цей текст буде показаний у вікні браузеру.
</body>
</html>
```

Цей простий приклад документу HTML, який містить мінімальну кількість тегів HTML і демонструє, як текст усередині елементу body відображається у браузері.

Якщо ввести великий об'єм тексту у такий спосіб, то читати його буде дуже незручне. Логічніше розбити його на параграфи, як в книзі, які підвищують читабельність тексту, і крім того виділяють смислові блоки.

Наступний приклад показує, як відображаються параграфи

```
<html>
<body>
<p>Це параграф 1.</p>
<p>Це параграф 2.</p>
<p>Це параграф 3.</p>
</body>
</html>
```

Цей приклад демонструє, як у браузері виводиться текст усередині елементів параграфа. Можна бачити, що за умовчанням текст кожного параграфа виводиться у вигляді окремого блоку. Кожен з таких блоків відділяється від попередніх і подальших блоків сторінки порожнім рядком. Проте відображення параграфа браузером може бути легко змінене за допомогою таблиці стилів.

Можна помітити, що параграфи можна записувати без закриваючого тега , проте краще за це не робити, в наступній версії HTML усі теги треба буде закривати.

У різних браузерах на різних моніторах з різним дозволом сторінка відображатиметься по-різному, тому не варто форматувати за допомогою додавання порожніх рядків і пропусків. Будь-яке число пропусків замінюється одним.

Використання порожніх параграфів для вставки порожніх рядків є

поганим стилем, замість цього використайте тег <br/>.

Атрибут ALIGN

Атрибут ALIGN дозволяє вирівняти текст по лівому або правому краю, по центру або ширині. За умовчанням текст вирівнюється по лівому краю. Цей атрибут застосований також до графіки і таблиць.

Далі приведені можливі значення атрибуту ALIGN :

ALIGN=justify вирівнювання по лівому і правому краям. Реалізовано не в усіх програмах інтерпретації.

ALIGN=left вирівнювання по лівому краю. За умовчанням текст HTML вирівнюється по лівому краю і не вирівнюється по правому, тобто початок рядків знаходиться на одному рівні по вертикалі, а кінці — на різних. Найчастіше, текст, що виходить при цьому, з рівними проміжками між словами виглядає краще. Оскільки вирівнювання по лівому краю задається автоматично, атрибут ALIGN=left можна опустити.

ALIGN=right вирівнювання по правому краю. Текст, вирівняний по правому краю і не вирівняний по лівому, — кінці рядків знаходяться на одному рівні, а почало на різних, — часто використовується з метою створити оригінальний дизайн. Для цього задається атрибут ALIGN=right в звичайних тегах, наприклад в тегу <P>.

ALIGN=center центрування тексту і графіки. Є декілька способів відцентрувати текст або графіку. У специфікаціях HTML 3.0 пропонується користуватися тегом <ALIGN="center">. Проте цей тег застосований не до усіх об'єктів HTML-сторінки, тому розробники Netscape додали тег <CENTER>, який центрує будь-які об'єкти і підтримується браузерами Netscape Navigator 3.0, Microsoft Internet Explorer 3.0 і іншими. До тега <CENTER> треба відноситися з обережністю. Який-небудь браузер може його взагалі проігнорувати, і на сторінці виявиться текст, вирівняний по лівому краю.

Заголовки

Заголовки визначаються за допомогою тегів від <h1> до <h6>. <h1> визначає заголовок найбільшого розміру, а <h6> визначає заголовок найменшого розміру.

```
<h1>Це заголовок першого рівня</h1>
<h2>Це заголовок другого рівня</h2>
<h3>Це заголовок третього рівня</h3>
<h4>Це заголовок четвертого рівня</h4>
<h5>Це заголовок п'ятого рівня</h5>
<h6>Це заголовок шостого рівня</h6>
```

Заголовки автоматично відділяються додатковими проміжками від інших елементів документу.

Перенесення рядків.

Для перенесення усередині параграфа використовується тег <br>, який виконує примусове перенесення рядка.

```
<html>
<body>
<p>Це <br>пара<br>граф з перенесеннями рядків</p>
</body>
</html>
```

Тег
 не має закриваючого тега. Тому для сумісності з майбутніми версіями стандарту рекомендується наступне написання тега

Горизонтальна лінійка

<hr> — тег, який малює горизонатльную лінію. Цей тег відноситься до блокових елементів, лінія завжди розпочинається з нового рядка. Має 5 атрибутів:

◦align — Визначає вирівнювання лінії. Може приймати значення left, center, right.

◦color — Задає колір лінії.

◦noshade — Малює лінію без тривимірних ефектів.

◦size — Задає товщину лінії.

◦width — Задає ширину лінії.

```
<html>
<body>
<p>Цей параграф відобразиться згори горизонтальної
смуги.</p>
<hr>
<p>Цей параграф відобразиться знизу горизонтальної
смуги.</p>
</body>
</html>
```

Тег <hr> не має закриваючого тега. Тому для сумісності з майбутніми версіями стандарту рекомендується наступне написання тега <hr/>. Для цього тега визначений ряд атрибутів, але вони є застарілими. І хоча їх застосування можливе, але консорціум W3C їх використати не рекомендує. Замість них слід використати таблиці стилів.

Коментарі в HTML

Тег коментаря використовується для вставки коментарів в початковий код HTML. Коментарі будуть проігноровані браузером. Коментарі можна використати для пояснення коду, що може допомогти при редагуванні початкового коду в майбутньому.

```
<!-- Це коментар -->
```

Ось приклад:

```
<html>
<body>
Цей текст буде показаний у вікні браузеру.
```

```
<!-- Цей текст не буде показаний, це коментар. -->
</body>
</html>
```

Додаткові параграфи

Цей приклад демонструє деякі особливості поведінки за умовчанням елементів параграфа.

```
<html>
<body>
<p>
Цей параграф містить багато рядків
у початковому коді, але браузер
це ігнорує.
</p>
<p>
Цей параграф містить багато пропусків
у початковому    коді
але     браузер це ігнорує.
</p>

</body>
</html>
```

Перенесення рядків

Цей приклад демонструє використання перенесення рядків в документі HTML.

```
<html>
<body>

<p>
Щоб виконати перенесення<br>рядків<br>у
<br>параграфі<br>використайте тег br.</p>
</body>
</html>
```

Фоновий колір

Цей приклад демонструє використання кольорового фону на сторінці HTML. При виборі фону завжди перевіряйте, щоб текст був добре читаний!

```
<html>
<body bgcolor="yellow">
<h2>Дивися: Кольоровий фон!</h2>
</body>
</html>
```

Інші теги для роботи з текстом

 — встановлює жирне зображення шрифту.

 - встановлює курсивне зображення шрифту.

<u></u> - додає підкреслення до тексту.

 — призначений для акцентування тексту. Браузери відображають такий текст курсивним зображенням.

<strike></strike> — перекреслює текст. Цей тег виключений з HTML5 замість нього рекомендується використати <s></s>

<tt></tt> - відображає текст моноширинним текстом. Виключений з HTML5.

 — відображає шрифт у вигляді верхнього індексу. Шрифт при цьому відображається вище за базову лінію тексту і зменшеного розміру.

 — відображає шрифт у вигляді нижнього індексу. Текст при цьому розташовується нижче базової лінії інших символів рядка і зменшеного розміру.

 — призначений для акцентування тексту. Браузери відображають такий текст жирним зображенням.

<small></small> — зменшує розмір шрифту на одиницю в порівнянні із звичайним текстом. У HTML розмір шрифту вимірюється в умовних одиницях від 1 до 7, середній розмір тексту, використовуваний за умовчанням, прийнятий 3. Тег <small> зменшує текст на одну умовну одиницю. Допускається застосування вкладених тегів <small>, при цьому розмір шрифту буде менше на 1 з кожним вкладеним рівнем, але не може бути менший, ніж 1.

<big></big> — збільшує розмір шрифту на одиницю в порівнянні із звичайним текстом. У HTML розмір шрифту вимірюється в умовних одиницях від 1 до 7, середній розмір тексту, використовуваний за умовчанням, прийнятий 3. Таким чином, додавання тега <big> збільшує текст на одну умовну одиницю. Допускається застосування вкладених тегів <big>, при цьому розмір шрифту буде більше з кожним рівнем.

<q></q> - використовується для виділення в тексті цитат. Вміст контейнера автоматично відображається у браузері в лапках.

<blockquote></blockquote> — призначений для виділення довгих цитат усередині документу. Текст, позначений цим тегом, традиційно відображається як вирівняний блок з відступами ліворуч і справа(приблизно по 40 пікселів), а також з відбиттям згори і знизу.

<pre></pre> - визначає блок заздалегідь форматованого тексту. Такий текст відображається зазвичай моноширинним шрифтом і з усіма пропусками між словами. За умовчанням, будь-яка кількість пропусків що йдуть в коде підряд, на веб-сторінці показується як один. Тег <pre> дозволяє обійти цю особливість і відображати текст як потрібно розробникові.

 — є контейнер для зміни характеристик шрифту, таких як розмір, колір і гарнітура. Хоча цей тег досі підтримується усіма браузерами, він вважається застарілим і від його використання рекомендується

відмовитися на користь стилів. Тег має 3 атрибути: color — встановлює колір тексту, face — визначає гарнітуру шрифту, size — задає розмір шрифту в умовних одиницях.

<cite></cite> - позначає текст як цитату або виноску на інший матеріал. Таке виділення зручне для зміни стилю тексту через CSS, а також застосовується для розділення коду HTML на структурні елементи. Браузери зазвичай встановлюють текст усередині контейнера <cite> курсивом.

<abbr></abbr> — вказує, що послідовність символів є абревіатурою. За допомогою атрибуту title дається розшифровка скорочення, що дозволяє розуміти абревіатуру тим людям, які з нею не знайомі. Крім того, пошукові системи індексують повнотекстовий варіант скорочення, що може використовуватися для підвищення рейтингу документу.

За умовчанням, текст ув'язнений в контейнері <abbr> підкреслюється пунктирною лінією.

Таблиця найчастіше використовуваних символьних об'єктів			
Результат	Опис	Ім'я об'єкту	Номер об'єкту
	нерозривний пропуск		
<	менше	&lt;	&#60;
>	більше	&gt;	&#62;
&	амперсанд	&amp;	&#38;
"	подвійна лапка	&quot;	&#34;
'	апостроф	&apos;	&#39;
£	фунт стерлінгів	&pound;	&#163;
¥	йена	&yen;	&#165;
§	параграф	&sect;	&#167;
©	авторське право	&copy;	&#169;
®	зареєстрована торгова марка	&reg;	&#174;
x	множення	&times;	&#215;
÷	ділення	&divide;	&#247;

Елемент розмітки <NOBR>

Тег <NOBR> (No Break, без обриву) дає браузеру команду відображати увесь текст в одному рядку, не обриваючи її. Якщо текст, поміщений в <NOBR>, не поміститься на екрані, браузер додасть в нижній частині вікна документу горизонтальну смугу прокрутки. Якщо ви хочете обірвати рядок у визначеному місці, поставте там тег <BR>.

Створення списків в HTML

Списки є важливим засобом структуризації тексту і застосовуються в усіх мовах розмітки. У HTML є наступні види списків : ненумерований список(неврегульований) (Unordered Lists <UL>), нумерований

список(впорядкований) (Ordered Lists <OL>) і список визначень. Теги для ненумерованих і нумерованих списків — це основа HTML. HTML 3.2 додає декілька атрибутів до тегів списків для вибору різних типів маркерів в ненумерованих списках і різних схем нумерації в нумерованих. Можна включати такі атрибути і в самі теги елементів списку(List Item <LI>), щоб змінити тип маркера в середині списку. Після появи нового атрибуту усі подальші маркери в списку матимуть такий же вигляд.

Ненумеровані списки

Ненумерований список є списком елементів. Елементи списку маркіруються за допомогою спеціальних знаків(зазвичай невеликий чорний круг).

Ненумерований список розпочинається з тега <ul>. Кожен елемент списку розпочинається з тега <li>.

```
<html>
<body>
<h4>Ненумерований список:</h4>
<ul>
  <li>елемент 1</li>
  <li>елемент 2</li>
  <li>елемент 3</li>
</ul>
</body>
</html>
```

Усередині елементу списку можна поміщати параграфи, перенесення рядків, зображення, посилання, інші списки, і так далі

Впорядковані списки

Впорядкований список також є списком елементів. Елементи списку маркіруються за допомогою чисел або букв.

Впорядкований список розпочинається з тега <ol>. Кожен елемент списку розпочинається з тега <li>. У тега <ol> може бути три атрибути: start(визначає перше число, з якого розпочинається нумерація пунктів), `reversed — Нумерація в списку стає в зворотному порядку(3,2,1),` і type(визначає стиль нумерації пунктів). Може мати значення:

"A" - заголовні букви A, B, C ...

"a" - рядкові букви a, b, c ...

"I" - великі римські числа I, II, III ...

"i" - маленькі римські числа i, ii, iii ...

"1" - арабські числа 1, 2, 3 ...

Усередині елементу списку можна поміщати параграфи, перенесення рядків, зображення, посилання, інші списки, і так далі

Списки визначень

Список визначень не є списком елементів. Це список термінів і визначень термінів.Список визначень розпочинається з тега <dl>. Кожен

термін списку визначень розпочинається з тега <dt>. Кожне визначення списку розпочинається з тега <dd>.

```
<html>
<body>
<dl>
    <dt>елемент 1</dt>
    <dd>опис елементу 1</dd>
    <dt>елемент 2</dt>
    <dd>опис елементу 2</dd>
</dl>
</body>
</html>
```

Усередині визначення списку визначень(тег <dd>) можна поміщати параграфи, перенесення рядків, зображення, посилання, інші списки, і так далі

Вкладений список:

```
<html>
<body>
<h4>Вкладений список:</h4>
<ul>
  <li>елемент 1</li>
  <li>елемент 2</li>
    <ul>
    <li>елемент 2.1</li>
    <li>елемент 2.2</li>
    </ul>
  <li>елемент 3</li>
</ul>
```

Гіперпосилання

Посилання встановлюється за допомогою тега, що «закривається» <a>, у якого є 12 атрибутів:

accesskey — активує посилання за допомогою комбінації клавіш.

charset - вказує кодування тексту, на який веде посилання. *

coords — встановлює координати активної області. *

href - задає адресу документу, на який слід перейти.

hreflang — ідентифікує мову тексту по посиланню.

name — встановлює ім'я якоря усередині документу(у HTML5 слід використати id). *

rel — стосунки між засиланим і поточним документами.

rev — стосунки між поточним і засиланим документами. *

shape — задає форму активної області посилання для зображень. *

tabindex — визначає послідовність переходу між посиланнями при натисненні на кнопку <Tab>.

target - тип вікна, в якому браузер завантажуватиме документ. Може набувати значень

_blank — завантажує сторінку в новому вікні браузеру.

_self — завантажує сторінку в поточному вікні(значення за умовчанням).

_parent — завантажує сторінку у фреймі-батьку, якщо фреймів немає, то цей параметр працює як _ self.

_top — відміняє усі фрейми і завантажує сторінку в повному вікні браузеру, якщо фреймів немає, то цей параметр працює як _ self.

title — додає спливаючу підказку до тексту посилання.

Атрибут accesskey дозволяє активувати посилання за допомогою деякого поєднання клавіш із заданою в коді посилання буквою або цифрою. Браузери при цьому використовують різні комбінації клавіш. Наприклад, для accesskey="s" працюють наступні поєднання.

Internet Explorer : Alt + S

Chrome: Alt + S

Opera: Shift + Esc, S

Safari: Alt + S

Firefox: Shift + Alt + S

```
<p><a href="images/pic.jpg" accesskey="x">
```

Атрибут coords застосовується до посилань, які розташовуються усередині контейнера <object>. Спільно з атрибутом shape створює «гарячу область», яка служить посиланням.

За наявності атрибуту download браузер не переходить по посиланню, а запропонує викачати документ, вказаний в адресі посилання.

Синтаксис

```
<a download>Посилання</a>
<p><a href="images/Pic.jpg" download>Викачати файл</a>
```

Атрибут href задає адресу документу, на який слід перейти. Оскільки як адреса посилання може використовуватися документ будь-якого типу, то результат переходу по посиланню залежить від кінцевого файлу. Так, архіви(файли з розширенням zip або rar) зберігатимуться на локальний диск. За умовчанням новий документ завантажується в поточне вікно браузеру, проте цю властивість можна змінити за допомогою атрибуту target.

```
<p><a href="example/knob.html">Відносне посилання</a></p>
<p><a href="www.google.com ">Абсолютне посилання</a></p>
```

Атрибут name використовується для визначення якоря усередині сторінки. Спочатку слід задати у відповідному місці закладку і встановити її ім'я за допомогою атрибуту name тега <a>. Ім'я посилання на закладку починається символом #, після чого йде назва закладки. Назва вибирається

будь-хто, що відповідає тематиці. Можна також робити посилання на закладку, що знаходиться в іншій веб-сторінці і навіть іншому сайті. Для цього в адресі посилання належить вказати її адресу і у кінці додати символ грат # і ім'я закладки.

Між тегами і текст писати не обов'язково, оскільки вимагається лише вказати місце розташування переходу по посиланню.

Синтаксис

```
<a name="закладка">...</a>
```

Атрибут tabindex визначає послідовність переходу між посиланнями при натисненні на кнопку Tab.

```
<p><a href="1.html" tabindex="1">Посилання 1</a></p>
<p><a href="3.html" tabindex="3">Посилання 3</a></p>
```

Атрибут target. За умовчанням, при переході по посиланню документ відкривається в поточному вікні або фреймі. При необхідності, ця умова може бути змінена атрибутом target тега <a>.

_blank - Завантажує сторінку в нове вікно браузеру.

_self - Завантажує сторінку в поточне вікно.

_parent - Завантажує сторінку у фрейм-батько, якщо фреймів немає, то це значення працює як _ self.

_top -Відміняє усі фрейми і завантажує сторінку в повному вікні браузеру, якщо фреймів немає, то це значення працює як _ self.

Атрибут title. Додає пояснюючий текст до посилання у вигляді спливаючої підказки. Така підказка відображається, коли курсор миші затримується на посиланні.

```
<p><a href="zoo.html" title="Рисунки різних тварин
і не лише.">  Рисунки   </a></p>
```

Атрибут type. Встановлює MIME- тип документу, на який вказує посилання. Цей атрибут носить рекомендаційний характер і може використовуватися для стилізації посилань із заданим типом документу. Атрибут type повинен додаватися тільки за наявності атрибуту href.

Використання зображень

Для того, щоб вставити зображення використовується тег <img>. Цей тег має єдиний обов'язковий атрибут src, який визначає адресу файлу з картинкою. Якщо необхідно, то рисунок можна зробити посиланням на інший файл, помістивши тег <img> у контейнер <a>. При цьому навколо зображення в деяких браузерах відображається рамка, яку можна прибрати, додавши атрибут border=″0″ в тег <img>. Також зображення можна маштабировать, вказавши в атрибутах width і height, бажаний розмір. Атрибут alt містить опис зображення, яке виводиться до завантаження

зображення. Для того, щоб задати позицію зображення(ліворуч, справа, по центру) використовується атрибут align, який може набувати п'ять значень: bottom, left, middle, right, top.

◦bottom — Вирівнювання нижньої межі зображення за навколишнім текстом.

◦left — Вирівнює зображення по лівому краю вікна.

◦middle — Вирівнювання середини зображення по базовій лінії поточного рядка.

◦right — Вирівнює зображення по правому краю вікна.

◦top — Верхня межа зображення вирівнюється по найвищому елементу поточного рядка.

Ось повний список атрибутів тега <img>:

◦align — Визначає як рисунок вирівнюватиметься по краю і спосіб обтікання текстом.

◦alt — Альтернативний текст для зображення.

◦border — Товщина рамки навколо зображення.

◦height — Висота зображення.

◦hspace — Горизонтальний відступ від зображення до навколишнього контенту.

◦ismap — Говорить браузеру, що картинка є серверною картою-зображенням.

◦longdesc — Вказує адреса документу, де міститься анотація до картинки.

◦src — Шлях до графічного файлу.

◦vspace — Вертикальний відступ від зображення до навколишнього контенту.

◦width — Ширина зображення.

◦usemap — Посилання на тег <map>, що містить координати для клієнтської карти-зображення.

2.3 Використання таблиць в HTML

Для опису таблиць використовується тег <TABLE>. Тег <TABLE>, як і багато інших, автоматично переводить рядок до і після таблиці.

Тег <TR> (Table Row, рядок таблиці) створює рядок таблиці. Увесь текст, інші теги і атрибути, які вимагається помістити в один рядок, повинні розміщуватися між тегами lt;TR></TR>.

Усередині рядка таблиці зазвичай розміщуються комірки з даними. Кожна комірка, що містить текст або зображення, має бути оточений тегами <TD></TD>. Число тегів <TD></TD> у рядку визначає число комірок

Заголовки для стовпців і рядків таблиці задаються за допомогою тега заголовка <TH></TH> (Table Header, заголовок таблиці). Ці теги подібні <TD></TD>. Відмінність полягає в тому, що текст, ув'язнений між тегами <TH></TH>, автоматично записується жирним шрифтом і по замовчуванню

розташовується пкоміркині комірки. Центрування можна відмінити і вирівняти текст по лівому або правому краю. Якщо скористатися <TD></TD> з тегом <B> і атрибутом <ALIGN=center>, текст теж виглядатиме як заголовок. Проте слід мати на увазі, що не усі браузери підтримують в таблицях жирний шрифт, тому краще задавати заголовки таблиць з допомогою <TH>.

Тег <CAPTION> дозволяє створювати заголовки таблиці. По замовчуванню заголовки центруються і розміщуються або над(<CAPTION ALIGN=top>), або під таблицею(<CAPTION ALIGN=bottom>). Заголовок може складатися з будь-якого тексту і зображень. Текст буде розбитий на рядки, що відповідають ширині таблиці. Іноді тег <CAPTION> використовується для підпису під рисунком. Для цього досить описати таблицю без меж.

```
<HTML>
<BODY>
  <TABLE BORDER>
  <CAPTION ALIGN=top>Заголовок над таблицею
</CAPTION>
    <TR>
      <TD>Текст або дані</TD>
      <TD>Текст або дані</TD>
      <TD>Текст або дані</TD>
      <TD>Текст або дані</TD>
    </TR>
  </TABLE>
  <TABLE BORDER>
  <CAPTION ALIGN=bottom>Заголовок під таблицею
  </CAPTION>
    <TR>
      <TD>Текст або дані</TD>
      <TD>Текст або дані</TD>
      <TD>Текст або дані</TD>
    </TR>
  </TABLE>
</BODY>
</HTML>
```

Зазвичай будь-який текст, що не поміщається в один рядок елементу таблиці, переходить на наступний рядок. Проте при використанні атрибуту NOWRAP з тегами <TH> чи <TD> довжина комірку розширюється настільки, щоб поміщений в неї текст помістився в один рядок.

Об’єдняння стовбців і рядків у таблицях

Теги <TD> і <TH> модифікуються за допомогою атрибуту COLSPAN(Column Span, з'єднання стовпців). Якщо ви хочете зробити який-

небудь комірка ширше, ніж верхня або нижня, можна скористатися атрибутом COLSPAN, щоб розтягнути її над будь-якою кількістю звичайних комірок.

```
<HTML>
<BODY>
  <CENTER>
  <TABLE BORDER=3>
    <TR>
      <TD>Якщо ви хочете зробити яку-небудь
          комірка ширше, ніж верхня або нижня
      </TD>
      <TD>можна скористатися атрибутом
          COLSPAN=2 </TD>
    </TR>
    <TR>
      <TD BGCOLOR=white COLSPAN=2>щоб
          розтягнути її над будь-якою кількістю
          звичайних комірок.</TD>
    </TR>
  </TABLE>
  </CENTER>
</BODY>
</HTML>
```

Атрибут ROWSPAN, використовуваний в тегах <TD> і <TH>, подібний до атрибуту COLSPAN=, тільки він задає число рядків, на які розтягується комірка. Якщо ви вказали в атрибуті ROWSPAN=s число, більше одиниці, то відповідна кількість рядків повинна знаходитися під розтягуваним комірком. Внизу таблиці її помістити не можна.

```
<table border = 1>
    <tr>
      <td>Комірка 1.1</td>
      <td rowspan = 2>Комірка 1+2.2</td>
      <td>Комірка 1.3</td>
    </tr>
    <tr>
      <td>Комірка 2.1</td>
      <td>Комірка 2.3</td>
    </tr>
   </table>
```

Атрибут WIDTH

Атрибут WIDTH застосовується в двох випадках. Можна помістити його в тег <TABLE>, щоб дати ширину усієї таблиці, а можна використати в тегах <TD> чи <TH>, щоб задати ширину комірку або групи комірок. Ширину можна вказувати в пікселах або у відсотках. Наприклад, якщо ви

задали в тегу <TABLE> WIDTH=250, ви отримаєте таблицю шириною 250 пікселів незалежно від розміру сторінки на моніторі. При завданні WIDTH=50% в тегу <TABLE> таблиця займатиме половину ширини сторінки при будь-якому розмірі зображення на екрані. Так що, вказуючи ширину таблиці у відсотках, майте на увазі, що якщо у користувача вузька область перегляду, ваша сторінка може виглядати декілька дивно. Якщо ви користуєтеся пікселами, і таблиця виявляється ширше за область перегляду, внизу з'явиться смуга прокрутки для переміщення управо і вліво по сторінці. Залежно від поставлених завдань і той, і інший спосіб завдання ширини таблиці може виявитися корисним.

Оформлення таблиці

Якщо комірка не містить даних, вона не матиме меж. Якщо вимагається, щоб у комірки були межі, але не було вмісту, необхідно помістити в неї щось, що не буде видно при перегляді. Можна скористатися порожнім рядком <BR>. Можна навіть задати порожні стовпці, визначивши їх ширину в пікселах або відносних одиницях і не ввівши в отримані комірки ніяких даних. Цей засіб може виявитися корисним при розміщенні на сторінці тексту і графіки.

Атрибут CELLPADDING визначає ширину порожнього простору між вмістом комірку і її межами, тобто задає поля усередині комірку.

```
<TABLE BORDER CELLPADDING=20>
```

Теги <TR>, <TD> і <TH> можна модифікувати за допомогою атрибутів ALIGN і VALIGN. Атрибут ALIGN визначає вирівнювання тексту і графіки по горизонталі, тобто по лівому або правому краю, або по центру. Горизонтальне вирівнювання може бути задане декількома способами :

ALIGN=bleedleft притискує вміст комірку впритул до лівого краю.

ALIGN=left вирівнює вміст комірку по лівому краю з урахуванням відступу, заданого атрибутом CELLPADDING.

ALIGN=center розташовує вміст комірку по центру.

ALIGN=right вирівнює вміст комірку по правому краю з урахуванням відступу, заданого атрибутом CELLPADDING.

Атрибут VALIGN здійснює вирівнювання тексту і графіки усередині комірку по вертикалі. Вертикальне вирівнювання може бути задане декількома способами :

VALIGN=top вирівнює вміст комірку по її верхній межі.

VALIGN=middle центрує вміст комірку по вертикалі.

VALIGN=bottom вирівнює вміст комірку по її нижній межі.

У тегу <TABLE> часто визначають, як виглядатимуть рамки, тобто лінії, що оточують елементи таблиці і саму таблицю. Якщо ви не задасте рамку, то отримаєте таблицю без ліній, але простір під них буде відведений. Того ж результату можна добитися, задавши <TABLE BORDER=0>. Іноді

хочеться зробити межу потовще, щоб вона краще виділялася. Можна для привертання уваги до рисунка або тексту задати виключно жирні межі. При створенні вкладених таблиць доводиться робити для різних таблиць межі різної товщини, щоб їх легше було розрізняти.

Атрибут CELLSPACING визначає ширину проміжків між комірками в пікселах. Якщо цей атрибут не вказаний, по замовчуванню задається величина, рівна двом пікселам. За допомогою атрибуту CELLSPACING= можна розміщувати текст і графіку там, де вам треба. Якщо ви хочете залишити порожнє місце, можна вписати в комірка пропуск.

```
<TABLE BORDER CELLSPACING=20>
```

Атрибут BGCOLOR дозволяє встановити колір фону. Залежно від того, з яким тегом(TABLE, TR, TD) він застосовується, колір фону може бути встановлений для усієї таблиці, для рядка або для окремого комірку. Значенням цього атрибуту є RGB- код або стандартна назва кольору.

```
<TABLE BORDER BGCOLOR=yellow>
```

Атрибут BACKGROUND задає фонове зображення для таблиць. Застосуємо до тегів TABLE і TD. Його значенням є URL файлу з фоновим зображенням. Застосування цього атрибуту розглядається нижче.

Інші атрибути тегу TABLE

frame - задає відображення меж навколо таблиці, може набувати значень:

void - не відображати межі;

border - відображати межу навколо таблиці;

above - провести межу тільки по верхньому краю таблиці;

below - провести межу тільки по нижньому краю таблиці;

hsides - відмалювати тільки горизонтальні межі;

vsides - відмалювати тільки вертикальні межі;

rhs - провести межу тільки по правій стороні таблиці;

lhs - провести межу тільки по лівій стороні таблиці;

height - задає висоту таблиці;

rules - задає відображення меж між комірками, може набувати значень:

all - відображати межі навколо кожного елементу таблиці;

groups - відображати межі між групами, що задаються тегами <thead>, <tfoot>, <tbody>, <colgroup>, <col>;

cols - відображати межі між колонками;

none - приховати усі межі;

rows - відображати межі між рядками таблиці;

summary - задає короткий опис таблиці;

width - задає ширину таблиці.

Усередині тега <table>, окрім <tr> і <td> допускається використання тегів :

<caption> - задає заголовок таблиці;

<col> - задає характеристики однієї, або декількох колонок таблиці;

<colgroup> - задає стиль для групи колонок таблиць;

<tbody> - зберігає одну, або декілька рядків таблиці, допустиме застосування тільки одного такого тега у рамках <table>;

<tfoot> - зберігає одну, або декілька рядків, що відображаються внизу таблиці;

<thead> - містить один, або декілька рядків, що відображаються вверху таблиці;

Використання таблиць в дизайні сторінки

Таблиці хороші тим, що за бажання можна зробити їх межі невидимими. Це дозволяє за допомогою тега <TABLE> красиво розміщувати на сторінці текст і графіку. Поки тег <TABLE> залишається єдиним потужним засобом форматування в HTML. Дизайнери Web- сторінок зараз мають практично ту ж свободу відносно використання «порожнього простору», що і творці друкарських сторінок. Таблиці краще всього допомагають відійти від ієрархічного розміщення тексту на Web- сторінках.

Якщо браузер підтримує таблиці, він зазвичай правильно відображає найцікавіші ефекти, отримані з їх допомогою

```
<HTML>
<BODY>
  <CENTER>
  <TABLE CELLPADDING="10" CELLSPACING="0"
         BORDER="16">
    <TR>
      <TD ALIGN="center">
        <H2>Прикарпатський національний
            університет імені В.Стефаника </H2>
        <H3>Ласкаво просимо!</H3>
        <TABLE BORDER WIDTH="100%">
          <TR>
            <TD ALIGN="center"><I>Навчальний курс
             "Основи Інтернет"</I></TD>
          </TR>
        </TABLE>
      </TD>
    </TR>
  </TABLE>
  </CENTER>
</BODY>
</HTML>
```

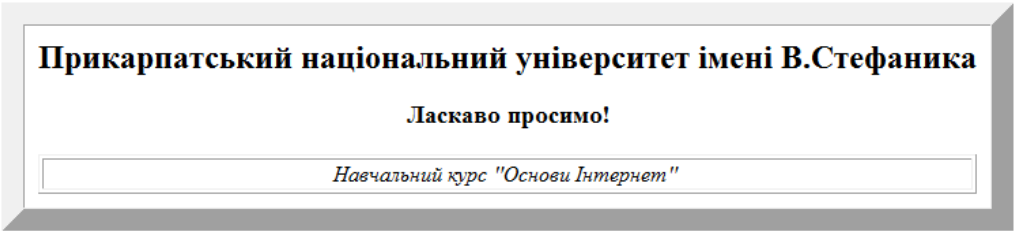


Рисунок 1.3 - Використання таблиць в дизайні

2.4 Робота з фреймами

У якомусь сенсі фрейм — це саме те, що означає це слово: рамка навколо картинки, віконце або сторінка. Вводячи тег <FRAME>, дизайнер HTML— сторінки розділяє екран браузеру на частини. В результаті чоловік, що переглядає сторінку, може вивчати тільки одну її частину, незалежно від іншого вмісту. Фактично браузер, що розпізнає фрейми, завантажує різні сторінки в різні секції, або фрейми, екрану. Наприклад, ви можете побудувати сторінку таким чином, що фірмовий знак буде зафіксований у верхній частині екрану, тоді як іншу частину сторінки користувач перегортує звичайним способом. Можна розташувати збоку кнопки навігація, яка не переміщається, коли читач клацає по них мишкою, так що змінюється тільки частина екрану, а сама смужка навігації залишається нерухомою.

Для створення фрейма використовується тег <frameset>, який замінює тег <body> у документі і застосовується для розділення екрану на області. Усередині цього тега знаходяться теги <frame>, які вказують на HTML-документ, призначений для завантаження в область.

У разі використання фреймів в першому рядку коду пишеться наступний тип документу

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">

Даний <!DOCTYPE> вказує браузеру, що він має справу з фреймами, цей рядок коду є обов'язковим. Контейнер <head> містить типову інформацію на зразок кодування сторінки і заголовка документу. Ось тільки врахуйте, що заголовок залишається незмінним, поки HTML- файли відкриваються усередині фреймів.

На перший погляд, фрейми — це щось складне, але їх легше зрозуміти, якщо провести аналогію з елементами таблиці. Розташування фреймів на екрані і комірок в таблиці задається майже однаково: теги і атрибути працюють так само, як їх табличні «родичі». Проте, хоча аналогія між одиничним фреймом на сторінці і елементом таблиці вірна, треба пам'ятати, що є і відмінності. Вміст комірки заданий в коде HTML— сторінки з таблицею. Текст або графіка, складові вміст таблиці, фактично вводяться на тій же сторінці HTML, що і тег або атрибут, що описують таблицю. Навпаки, екран з фреймами описується в HTML— сторінці, в контейнері FRAMESET. Вміст же фрейма — це окрема HTML— сторінка, яка може знаходитися де

завгодно : в іншому каталозі, на локальному сервері або на видаленому вузлі десь в мережі.

Фреймова структура визначає тільки спосіб організації екрану з фреймами і вказує, де знаходиться початковий вміст кожного фрейма. Для усіх фреймів задаються URL, що описують місцезнаходження їх даних. Як правило, на сторінці з фреймовою структурою вмісту фреймів немає. Така сторінка зазвичай невелика — вона описує тільки кадрову структуру екрану. Коли документ завантажується у фрейм, ви можете клацати мишкою на посиланні в цьому документі, щоб побачити пов'язані документи в інших кадрах, заданих у фреймовій структурі .

Створення простої сторінки з фреймами

Побудуємо сторінку з двома фреймами. Задамо ліворуч фрейм змісту із заголовками статей, а справа помістимо сторінку з самими статтями. Зробимо так, що коли користувач клацає мишкою на посиланні в тій частині екрану, де знаходиться зміст, сама стаття з'являється в правому фреймі. Це основний, найбільш поширений спосіб використання фреймів.

Спершу ми повинні уявити собі загальний вигляд сторінки - де розташувати фрейми і якого вони будуть розміру. Потім можна подумати про їх зміст. Нижче наводиться код простої фреймової структури з використанням тега <FRAMESET>. Зверніть увагу: сторінка з фреймовою структурою не містить тега <BODY>.

```
<HTML>
<HEAD>
<TITLE>Приклад фреймів</TITLE>
</HEAD>
<FRAMESET COLS="25%, 75%">
<FRAME SRC="menu.html">
<FRAME SRC="main.html" NAME="main">
</FRAMESET>
</HTML>
```

Ось і увесь код, необхідний для того, щоб задати фреймову структуру. В результаті ми отримали екран, розділений на два вікна. Ліве вікно займає 25% екрану і містить сторінку з назвою menu.html. Вікно справа займе 75% екрану і містить файл main.html. Поки у нас їх немає, так що ви побачите сторінку з двома порожніми фреймами. Перш ніж вона з'явиться, нам доведеться пару разів клацнути мишкою у відповідь на повідомлення про помилки, тому що браузер намагатиметься знайти неіснуючі сторінки. Зверніть увагу, що праву сторінку ми назвали «main»(<головна>) за допомогою рядка:

<FRAME SRC="main.html" NAME="main">

Це означає, що фрейм під ім'ям main міститиме сторінку main.html. Помітимо, що оскільки ми не збираємося показувати в лівому фреймі ніяких

сторінок, окрім menu.html, нам не треба його називати.

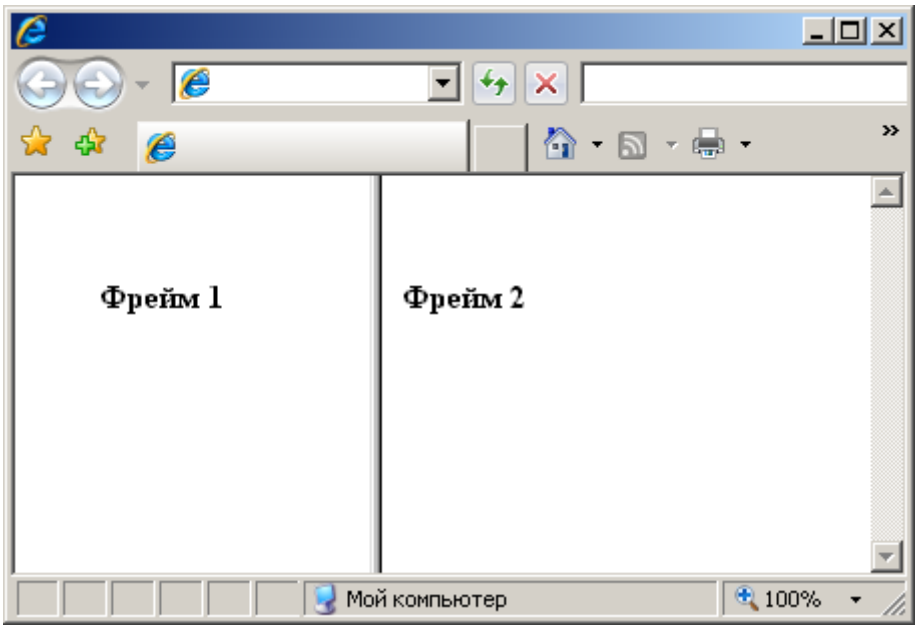


Рисунок 1.5 Сторінка з фреймовою структурою

Тепер завантажимо фрейми з вмістом. Задамо сторінку menu.html в лівому фреймі, де ми збираємося клацати мишею, перемикаючись між двома сторінками в правому фреймі. menu.html — це звичайна HTML— сторінка, побудована як зміст. Насправді ми можемо узяти готову сторінку зі змістом і використати її. Майте на увазі, що цей фрейм вузький і високий, так що сторінка, яка в нього завантажуватиметься, має бути спроектована відповідним чином. Тепер ми повинні визначити, де з'являтимуться інші сторінки при клацанні мишкою на посиланні. Оскільки ми хочемо, щоб вони відображалися в правому фреймі, додамо атрибут TARGET(TARGET=«main») в тег посилання. Це означає, що, коли користувач клацає на посиланні, сторінка, що викликається, з'являється у фреймі main. Ми відображаємо усі сторінки у фреймі main, тому давайте додамо атрибут TARGET=«main» в усі теги посилань в змісті. Якщо ми не визначимо атрибут TARGET, то сторінка з'явиться там, де ми клацнули мишкою, — в лівому фреймі.

Правий фрейм main міститиме самі HTML— сторінки. Ваше завдання — спроектувати їх так, щоб вони добре виглядали в меншому, ніж зазвичай, вікні, тому що частина екрану буде зайнята лівим кадром змісту. Але більше ці сторінки нічим не примітні.

У деяких користувачів ще залишилися браузери, що не уміють поводитися з фреймами. З цієї причини розумно надати доступ до версії ваших основних сторінок без фреймів. Якщо читач із застарілим браузером виявиться на вашій сторінці з фреймовою структурою, все, що знаходиться на ній між тегами <NOFRAMES> і </NOFRAMES>, виглядатиме відмінно — браузер просто проігнорує фрейми. Ось чому обов'язково треба використати теги <BODY> </BODY>. Можливо, екран без фреймів доведеться

організувати інакше.

Приклад сторінки з фреймовою структурою з доданим у кінці розділом <NOFRAMES>.

```
<HTML>
<HEAD>
<TITLE>Приклад фреймів</TITLE>
</HEAD>
<FRAMESET COLS="25%, 75%">
<FRAME SRC="menu.html">
<FRAME SRC="main.html" NAME="main">
<NOFRAMES>
Ви переглядаєте цю сторінку з допомогою
браузеру, що не підтримує фрейми.
</NOFRAMES>
</FRAMESET>
</HTML>
```

Майте на увазі, що підтримувальний фрейми браузер проігнорує все, що знаходиться між тегами <NOFRAMES> і </NOFRAMES>. І навпаки, не підтримувальний фрейми браузер проігнорує все, що знаходиться між тегами <FRAMESET> і </FRAMESET>. Код без фреймів можна помістити і в початок, і в кінець сторінки.

Макетування фреймів — тег <FRAMESET>

Теги <FRAMESET> обрамляють текст, що описує компонування фреймів. Тут розміщується інформація про число фреймів, їх розміри і орієнтацію(горизонтальною або вертикальною). У тега <FRAMESET> тільки два можливі атрибути: ROWS, задаючий число рядків, і COLS, задаючий число стовпців. Між тегами <FRAMESET> не вимагається вказувати тег <BODY>, але його можна помістити між тегами <NOFRAMES> у кінці фреймової структури. Між тегами <FRAMESET> не повинно бути ніяких тегів або атрибутів, які зазвичай використовуються між тегами <BODY>. Єдиними тегами, які можуть знаходитися між тегами <FRAMESET> і </FRAMESET>, являються теги <FRAME>, <FRAMESET> і <NOFRAME>. Це спрощує завдання. В основному все пов'язано з тегами <FRAME> і їх атрибутами. Якщо ж ви хочете поекспериментувати, можна створити вкладені один в одного теги <FRAMESET> аналогічно тегам <TABLE>.

Для кожного рядка і стовпця, згаданих в тегу <FRAMESET>, потрібний свій набір тегів <FRAME>.

Атрибут ROWS тега <FRAMESET> задає число і розмір рядків на сторінці. Кількість тегів <FRAME> повинно відповідати вказаному числу рядків. Праворуч від знаку «=» можна визначити розмір кожного рядка в пікселах, відсотках від висоти екрану або у відносних величинах(звичайна ця вказівка зайняти частину місця, що залишилася). Слід користуватися лапками і комами, а також залишати пропуски між значеннями атрибутів. Наприклад, наступний запис формує екран, що складається з трьох рядків :

висота верхньої — 20 пікселів, середньої — 80 пікселів, нижньої — 20 пікселів:

```
<FRAMESET ROWS="20, 80, 20">
```

Наступний тег — <FRAMESET> — створює екран, на якому верхній рядок займає 10% висоти екрану, середня — 60%, а нижня — решта 30% :

```
<FRAMESET ROWS="10%, 60%, 30%">
```

Можна задати відносні значення в комбінації з фіксованими, вираженими у відсотках або пікселах. Наприклад, наступний тег створює екран, на якому верхній рядок має висоту 20 пікселів, середня — 80 пікселів, а нижня займає місце, що все залишилося :

<FRAMESET ROWS="20, 80, *">

Стовпці задаються так само, як рядки. Для них застосовні ті ж атрибути.

Тег <FRAME> визначає зовнішній вигляд і поведінку фрейма. Цей тег не має закриваючого тега, оскільки в нім нічого не міститься. Уся суть тега <FRAME> у його атрибутах. Їх шість : NAME, MARGINWIDTH, MARGINHEIGHT, SCROLLING, NORESIZE і SRC.

Якщо ви хочете, щоб при клацанні мишею на посиланні відповідна сторінка відображалася в певному фреймі, необхідно вказати цей фрейм, щоб сторінка «знала», що куди завантажувати. У попередніх прикладах ми назвали великий правий фрейм main, і саме в нім з'являлися сторінки, вибрані зі змісту в лівому фреймі. Фрейм, в якому відображаються сторінки, називається цільовим(target). Фрейми, які не є цільовими, іменувати не обов'язково. Наприклад, можна записати такий рядок:

<FRAME SRC="my.html" NAME="main">

Імена цільових фреймів повинні розпочинатися з букви або цифри. Одні і ті ж імена дозволяється використати в декількох фреймових структурах. По клацанню миші відповідні сторінки відображатимуться в іменованому фреймі.

Атрибут MARGINWIDTH діє аналогічно атрибуту таблиць CELLPADDING. Він задає горизонтальний відступ між вмістом кадру і його межами. Найменше значення цього атрибуту дорівнює 1. Не можна вказати 0. Можна не привласнювати нічого — за умовчанням атрибут дорівнює 6.

Атрибут MARGINHEIGHT діє так само як і MARGINWIDTH. Він задає поля у верхній і нижній частинах фрейма.

Атрибут SCROLLING дає можливість користуватися прокруткою у фреймі. Можливі варіанти: SCROLLING=yes, SCROLLING=no, SCROLLING=auto. SCROLLING=yes означає, що у фреймі завжди будуть смуги прокрутки, навіть якщо це не треба. Якщо задати SCROLLING=no, смуг прокрутки не буде, навіть коли це необхідно. Якщо документ занадто великий, а ви задали режим без прокрутки, документ просто буде обрізаний. Атрибут SCROLLING=auto надає браузеру самому вирішувати, потрібно смуги прокрутки або ні. Якщо атрибут SCROLLING відсутній, результат буде таким же, як при використанні SCROLLING=auto.

Як правило, користувач може, переміщаючи межу фрейма мишкою, змінити його розмір. Це зручно, але не завжди. Іноді потрібно атрибут NORESIZE. Пам'ятайте: усі межі фрейма, для якого ви задали NORESIZE, стають нерухомими - відповідно, може виявитися так, що розміри сусідніх фреймів теж стануть фіксованими. Користуйтеся цим атрибутом з обережністю.

Атрибут SRC застосовується в тегу FRAME при розробці фреймової структури для того, щоб визначити, яка сторінка з'явиться в тому або іншому кадрі. Якщо ви задасте атрибут SRC не для усіх фреймів, у вас виникнуть проблеми. Навіть якщо сторінки, що відображаються у фреймі, вибираються в сусідньому фреймі, ви повинні принаймні задати для кожного фрейма початкову сторінку. Якщо ви не вкажете початкову сторінку і URL, фрейм виявиться порожнім, а результати можуть бути найнесподіванішими.

Щоб розібратися з атрибутом TARGET, необхідно повернутися до простого прикладу з кадром змісту. Коли користувач клацає мишкою на одному з посилань в лівому фреймі, відповідна сторінка повинна з'явитися в правому фреймі, а зміст залишається незмінним. Щоб цього добитися, треба визначити цільовий фрейм TARGET, в якому відображатиметься сторінка для кожного пункту змісту. Цільові фрейми задаються в посиланнях лівого фрейма. Ось навіщо усім кадрам у фреймовій структурі були присвоєні імена. Правий фрейм називається main, так що треба в кожному посиланні додати атрибут TARGET=«main», внаслідок чого відповідна сторінка з'явиться у фреймі main. Зверніть увагу: кожне посилання містить атрибут TARGET=«main», який по клацанню миші відображає сторінку у фреймі main.

Атрибут TARGET можна задавати для декількох різних тегів. При використанні в тегу <BASE> він направляє усі посилання в певний цільовий фрейм, якщо надалі не передбачене інше. Можна задати атрибут TARGET в тегу <AREA> у активному зображенні або в тегу <FORM>. Фрейми корисні для організації форм. Користувачі бачитимуть одночасно і форму, і результат свого вибору. Зазвичай при клацанні мишею кнопки Submit форма зникає, і з'являється сторінка з результатами вибору. Поєднання форм і фреймів може виявитися зручним способом навігації.

Вкладені і множинні кадрові структури

Вкладені фрейми не дуже сприяють навігації. Та все ж бувають випадки, коли виникає потреба розмістити одні фрейми усередині інших. Фрейми самі по собі — незвичайний засіб навігації, і немає чого ще більше ускладнювати свої сторінки. Але якщо вам все ж потрібні вкладені фрейми, то вони не викликають проблем.

В основному вкладені фрейми діють так само, як вкладені таблиці. Задайте кадрову структуру, а усередині якого-небудь фрейма в ній — ще одну структуру. Необхідно пам'ятати, що тег <FRAME> не має закриваючого тега. Ви, напевно, помітили, що при роботі з фреймами не використовуються

атрибути <COLSPAN> і <ROWSPAN>. Їх роль грають множинні, або вкладені, фрейми. Задавши усередині однієї охоплюючої фреймової структури дві незалежні **підструктури**, можна помістити в лівій частині екрану стовпець з двох, а в правій — з трьох фреймів.

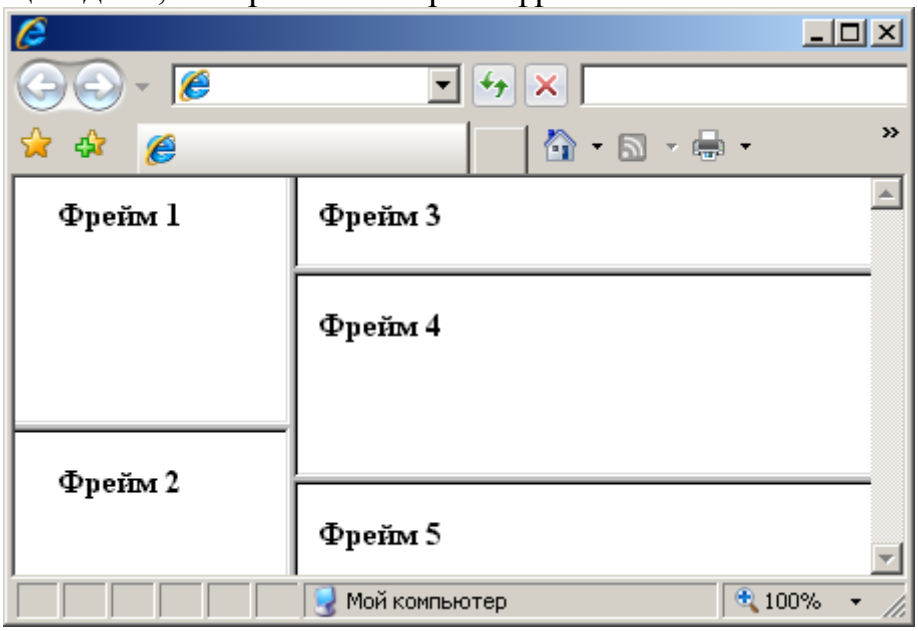


Рисунок 1.6 Вкладені фрейми

Плаваючі фрейми

Тег <iframe> створює плаваючий фрейм, який знаходиться усередині звичайного документу, він дозволяє завантажувати в область заданих розмірів будь-які інші незалежні документи.

Тег <iframe> є контейнером, зміст якого ігнорується браузерами, що не підтримують цей тег. Для таких браузерів можна вказати альтернативний текст, який побачать користувачі. Він повинен розташовуватися між елементами <iframe> і </iframe>.

Синтаксис

```
<iframe>...</iframe><iframe>...</iframe>
```

Атрибути:

align - визначає як фрейм вирівнюватиметься по краю, а також спосіб обтікання його текстом.

allowtransparency - встановлює прозорий фон фрейма, через який видно фон сторінки.

frameborder - встановлює, відображати межу навколо фрейма або ні.

height - висота фрейма.

hspace - Горизонтальний відступ від фрейма до навколишнього контенту.

marginheight - Відступ згори і знизу від змісту до межі фрейма.

marginwidth - Відступ ліворуч і праворуч від змісту до межі фрейма.

name - Ім'я фрейма.

sandbox - Дозволяє задати ряд обмежень на контент завантажуваний у фреймі.

scrolling - Спосіб відображення смуги прокрутки у фреймі.

seamless - Визначає, що вміст фрейма повинен відображатися так, немов воно є частиною документу.

src - Шлях до файлу, вміст якого завантажуватиметься у фрейм.

srcdoc - Зберігає вміст фрейма безпосередньо в атрибуті.

vspace - Вертикальний відступ від фрейма до навколишнього контенту.

width - Ширина фрейма.

Навігаційні карти

Активні зображення(image maps), або зображення, чутливі до клацань миші, дозволяють створити на вузлі графічні меню довільної форми. Активне зображення — це зображення з так званими активними областями(hot spots), які посилаються на URL інших сторінок або вузлів.

Є два методи формування активних зображень : на сервері і у клієнта. Зображення першого типу використовують сервер для того, щоб знайти той, що відповідає цій активній області URL і передати на браузер потрібну сторінку. Активні зображення, працюючі на клієнтській машині, задають інформацію про активну область на HTML- сторінці, так що браузер сам з'ясовує, які області є активними, і просить з сервера відповідну сторінку.

Активні зображення, працюючі у клієнта, мають декілька переваг. По-перше, сторінки з ними можна перенести на інший сервер. По-друге, серверу не доводиться виконувати зайву роботу(наприклад, переглядати усю інформацію про активні області), тобто навантаження на сервер зменшується. При використанні працюючих на сервері активних зображень в каталозі cgi - bin сервера має бути відповідний сценарій. З міркувань безпеки багато системних адміністраторів не записують сценарії в каталог cgi - bin. Тому детальніше ми розглянемо створення активних зображень у клієнта.

Створення активного зображення. Процес створення активного зображення складається з двох етапів. Спочатку необхідно визначити на картинці області, які треба зробити активними, а потім співвіднести їх з посиланнями на інші URL. Активні області задаються перерахуванням їх координат(у пікселах). Усе це можна зробити вручну, визначивши координати кутів активних областей, але набагато простіше скористатися якою-небудь програмою, наприклад MapEdit.

Визначити карту легко. Треба відкрити в MapEdit HTML- файл, що містить зображення, на якому вимагається створити активні області, після чого вибране зображення буде завантажено в робоче вікно. Потім слід вибрати тип активної області(квадрат, трикутник і круг), клацнути і потягнути мишкою, позначивши межу області. Програма автоматично робить запис в HTML- файл, що описує межі активної області. Потім цій області треба приписати URL. У будь-яких місцях зображення можна намалювати активні області і визначити для кожної з них URL. Важливо залишати між областями трохи місця, щоб при читанні бути упевненим, що активізується правильне посилання. Межі активних областей задаються координатами

кутів прямокутника і багатокутника або центру і радіусу круга. Якщо ви вирішили робити активне зображення у клієнта, Map Edit поставляє дані тільки для тегів <MAP>. Вам доведеться самим задати тег зображення з атрибутом USEMAP і помістити його після тега

</MAP>. Не забудьте перед ім'ям карти в

атрибуті USEMAP записати символ «#» таким чином:

```
<IMG SRC="mymap.gif" USEMAP="#sitemap">
```

Активні зображення у клієнта працюють незалежно від програмного забезпечення сервера і не перестануть функціонувати, навіть якщо файли будуть перенесені на інший сервер. Таким зображенням потрібно тільки дві речі: браузер, підтримувальний HTML 3.0, і інформація про карту, записана в HTML- файлі. Наведемо приклад активних зображень.

```
<IMG SRC="ball.gif" ALT="Навігаційна карта"
     WIDTH="200" HEIGHT="200" USEMAP="#imap">
<MAP NAME="imap">
<AREA SHAPE="rect" COORDS="0,0,100,200"
    HREF="frame.html">
<AREA SHAPE="rect" COORDS="150,0,200,200"
    HREF="Untitled1.html">
<AREA SHAPE="default" nohref>
</MAP>
```

Контрольні питання до теми 2

1. Який тег заголовної частини документа?
2. Який тег сповіщає броузер про розрив рядка?
3. Вкажіть тег, який створить текст такого стилю – підкреслений напівжирний текст.
4. Який тег створює абзац?
5. Що таке escape-послідовності? Коли вони використовуються?
6. Як задати цитату у web-документі?
7. Для чого використовується тег <TITLE>?
8. Який тег розміщує текст посередині рядка?
9. Який тег повинен бути першим у web-документі?
10. В рамках дії якого тегу записують вміст документа?
11. Коли використовують рівні заголовків <Hx>?
12. Вкажіть послідовність вкладення тегів у web-документі
13. Який тег додає картинку у документ?
14. Які обов'язкові параметрами тегу створення зображення?
15. Які необов'язкові параметрами тегу створення зображення?
16. Як створити ненумерований список? Вкажіть параметри такого списку.
17. Коли використовують список визначень?
18. Які є види фреймів, різниця між нимим?

Тема 3. Основні поняття CSS. Форматування тексту

3.1 Призначення та способи застосування CSS

CSS це мова стилів, що визначає відображення HTML-документів. CSS, на сьогоднішній день, підтримується всіма браузерами (програмами перегляду). HTML використовується для структурування вмісту сторінки. CSS використовується для форматування цього структурованого вмісту

Специфікація CSS(Cascading Style Sheets) дозволяє залишитися у рамках декларативного характеру розмітки сторінки і повністю контролювати форму представлення елементів HTML- розмітки.

Каскадні таблиці стилів покликані вирішити протиріччя між точністю визначення розмірів картинок і додатків, з одного боку, і точністю визначення розмірів блоків тексту і його зображення — з іншою.

Таблиці стилів також дозволяють визначити колір і зображення текстового фрагмента, змінювати ці параметри усередині текстового блоку, виконувати вирівнювання текстового блоку відносно інших блоків і компонентів сторінки.

Конкретні переваги CSS:

• управління відображенням безлічі документів за допомогою однієї таблиці стилів;

• більш точний контроль над зовнішнім виглядом сторінок;

• поданням для різних носіїв інформації (екран, друк, і т. д.);

• складна і пророблена техніка дизайну.

Наявність подібних можливостей дозволяє говорити про CSS як про засіб розділення логічної структури документу і форми його представлення. Логічна структура документу визначається елементами HTML- розмітки, тоді як форма представлення кожного з цих елементів задається CSS- описувачем елементу.

Способи застосування CSS

Під способами застосування CSS ми в цьому розділі розуміємо форму декларування стилю на HTML-сторінці і форму зв'язування опису стилю відображення елементу розмітки з самим елементом. Йдеться про те, де і в якій формі автор сторінки(чи дизайнер) описує стиль, і як і в якій формі на нього посилається.

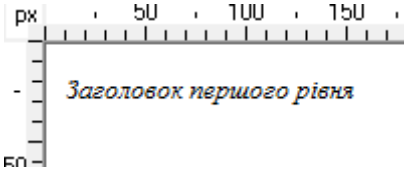
Розрізняють чотири способи застосування стилів :

- перевизначення стилю в елементі розмітки ;
- розміщення опису стилю в заголовку документу в елементі STYLE ;
- розміщення посилання на зовнішній опис через елемент LINK ;
- імпорт опису стилю в документ.

Перевизначення стилю:

```
<H1 STYLE="font-weight: normal;
```

```
font-style: italic;
font-size: 10pt;">
Заголовок першого рівня
</H1>
```



Атрибут style можна застосувати усередині будь-якого елементу розмітки. Наприклад, ми можемо через style визначити ширину і вирівнювання елементу hr(горизонтальна лінія) :

```
<HR STYLE="width: 100px;">
```

Очевидно, що не усі параметри стилю можна встановити для конкретного елементу розмітки. Про типи елементів і відповідні параметри стилів ми поговоримо в розділі "Поняття блочного і строкового елементів".

Тут же треба відмітити наступне: стилі розроблені в першу чергу для управління відображенням тексту. Не слід захоплюватися стилями при управлінні відображенням нетекстових елементів HTML- розмітки.

Розміщення опису стилю в заголовку документу в елементі STYLE

Застосування елементу STYLE — це основний спосіб впровадження каскадних таблиць стилів в тіло HTML— документу. Окрім управління відображенням елементів розмітки, елемент STYLE дозволяє описувати стильові властивості елементів, які можна змінювати при програмуванні на JavaScript.

Елемент STYLE дає можливість визначити стиль відображення :

- стандартних елементів HTML- розмітки ;
- довільних класів(селектор CLASS);
- HTML-об'єктів(селектор ID).

На жаль, робота з селекторами у браузерах різних виробників може підносити різного роду сюрпризи, для уникнення яких слід дотримуватися стандартів W3C.

Поняття селектора, застосування селекторів і формальний синтаксис CSS ми обговоримо в розділах "Синтаксис" і " Наслідування і перевизначення ".

Стандартні елементи розмітки описуються в елементі STYLE таким чином:

```
<HEAD>
<STYLE>
p { color: darkred;text-align: justify;
    font-size: 8pt; }
</STYLE>
</HEAD>
<BODY>
<P> Цей параграф ми використовуємо
```

```
як приклад застосування опису стилю для
стандартного елементу HTML- розмітки.</P>
</BODY>
```

Тепер усі параграфи документу відображатимуться стилем з елементу STYLE, якщо тільки стиль не буде яким-небудь чином перевизначений. У STYLE можна визначити стиль будь-якого елементу розмітки.

Розміщення посилання на зовнішній опис

Посилання на опис стилю, розташованого за межами документу, здійснюється за допомогою елементу LINK, який розміщують в елементі HEAD . Зовнішній опис може бути файлом, що містить опис стилів. Опис стилів в цьому файлі по синтаксису в точності співпадатиме зі змістом елементу STYLE.

Нижче наведений приклад посилання на зовнішній опис стилів :

```
<LINK TYPE="text/css" REL="stylesheet"
      HREF="http://mysite.at.ua/my_css.css">
```

Тут важливі значення атрибутів REL і TYPE. Атрибут REL повинен мати значення stylesheet. Type може набувати значень text/css або text/javascript. Другий тип опису стилів введений Netscape. Його ми в цьому учбовому курсі не обговорюємо.

Атрибут HREF задає універсальний локатор ресурсу(URL) для зовнішнього файлу опису стилів. Це може бути посилання на файл з будь-яким ім'ям, а не тільки на файл з розширенням *.css.

Імпорт опису стилю в документ

Імпорт описувачів стилів в деякому розумінні складає конкуренцію представленій вище вказівці на зовнішній описувач стилю.

Імпортувати стиль можна або всередину елементу STYLE, або всередину зовнішнього файлу, який є описувачем стилю. Оператор імпорту стилю повинен передувати усім іншим описувачам стилів :

```
<STYLE>
@import url(http://mysite.at.ua/my_css.css)
A { color: cyan;text-decoration: underline; }
</STYLE>
```

Стиль, що імпортується, можна перевизначити або через описувач елементу в STYLE, або через атрибут елементу STYLE.

Загальний синтаксис селекторів в CSS

Формально стиль відображення елементів розмітки задається посиланням в елементі розмітки на селектор стилю. Синтаксис опису стилів в

загальному вигляді представляється таким чином:

```
selector[, selector[, ...]]
        { attribute: value;
        [attribute: value;...] }
```

чи

```
selector selector [selector ...]
         { attribute: value;
         [attribute: value;...] }
```

У першому варіанті перераховані селектори, для яких діє цей опис стилю. Другий варіант задає ієрархію вкладеності селекторів, для сукупності яких визначений стиль. Нагадаємо, що в даному випадку йдеться про описи стилів в нотації text/css. Описи стилів розміщуються або усередині елементу STYLE, або в зовнішньому файлі.

В якості селектора можна використати ім'я елементу розмітки, ім'я класу і ідентифікатор об'єкту на HTML- сторінці.

Атрибут(attribute) визначає властивість елементу, що відображається, наприклад лівий відступ параграфа(margin — left), а значення(value) — значення цього атрибуту, наприклад, 10 друкарських пунктів(10 pt).

Селектор — ім'я елементу розмітки

Коли автор Web- вузла хоче визначити загальний стиль усіх сторінок, він просто прописує стилі для усіх елементів HTML- розмітки, які використовуватимуться на сторінках. Це дає можливість скомпонувати сторінки з логічних елементів, а стиль відображення елементів описати в зовнішньому файлі.

Такий спосіб створення сайту дозволяє авторові змінювати зовнішній вигляд усіх сторінок шляхом внесення змін до файлу опису стилів, а не у файли HTML- сторінок.

Зовнішній файл при цьому може виглядати таким чином:

```
I, EM {color:#003366;font-style: normal;}
A I {font-style: normal;font-weight: bold;
      text-decoration: line-through;}
```

У першому рядку цього опису перераховані селектори-елементи, які відображатимуться однаково :

```
<I>Це курсив</I> і це теж <EM>курсив</EM>
```

Останній рядок визначає стиль відображення вкладеного у гіпертекстове посилання курсиву :

```
<A NAME=empty><I>intuit</I></A>
```

В даному випадку перевизначення полягає в тому, що текст відображається усередині гіпертекстового посилання перекресленим,

причому жирним шрифтом.

Селектор — ім'я класу

Ім'я класу не є яким-небудь стандартним ім'ям елементу HTML-розмітки. Воно визначає опис класу елементів розмітки, які відображатимуться однаково. Для того, щоб віднести елемент розмітки до того або іншого класу, треба скористатися його атрибутом CLASS:

```
<STYLE>
.test {color: white;background-color: black;}
</STYLE>
...
<P CLASS="test"> Цей параграф ми відобразимо
    білим кольором по чорному фону
</P>
<P>Це <A CLASS="test">гіпертекстове посилання</A>
    ми відобразимо білим кольором по чорному фону.
</P>
```

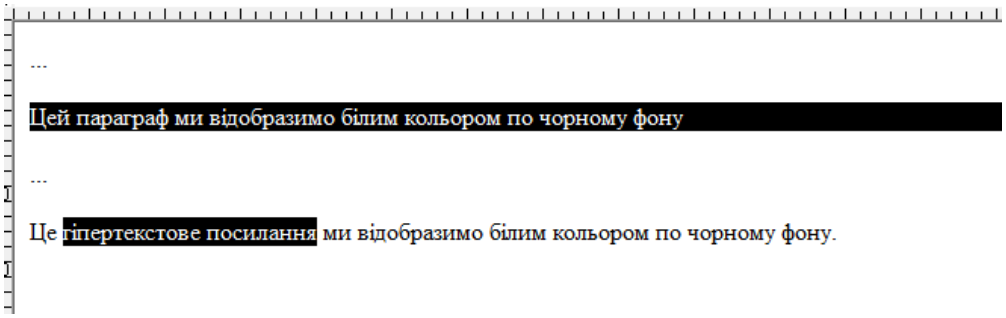


Рисунок 3.1. Селектор — ім'я класу

Таким чином у будь-якому елементі розмітки можна послатися на опис класу відображення. При цьому абсолютно необов'язково, щоб елементи розмітки були однотипними. У прикладі до одного класу віднесені і параграф, і гіпертекстове посилання в іншому параграфі.

Лідируючу точку в імені класу можна опустити. Вона задається з міркувань збереження єдності опису. Наприклад, можна визначити класи відображення однотипних елементів розмітки :

```
a.menu { color: red;background-color: white;
         text-decoration: none; }
a.paragraph { color: navy;
              text-decoration: underline; }
```

У цьому прикладі клас гіпертекстових посилань menu має один опис стилю, а клас гіпертекстових посилань paragraph — абсолютно інший. При цьому кожен з цих класів не можна застосувати до інших елементів розмітки, наприклад, параграфу або списку. Якщо ім'я елементу розмітки не задане, це означає, що клас можна віднести до будь-якого елементу розмітки — кореневий клас опису стилів. Це дуже схоже на позначення імені кореневого

домена в системі доменних імен. Власне нічого дивовижного тут немає, оскільки система класів об'єктів на HTML— сторінці є деревом. Елементи розмітки — це вузли дерева.

Селектор — ідентифікатор об'єкту

Об'єктна модель документу(Document Object Model) описує документ як дерево об'єктів. Об'єктами є: сам документ, його розділи(елемент DIV), картинки, параграфи, додатки і тому подібне. Кожному з об'єктів можна дати ім'я і звертатися до нього по імені. Ця можливість використовується при програмуванні сторінок на стороні клієнта.

Застосування ідентифікатора об'єкту виправдане ще і у разі модифікації атрибуту опису стилю для цього об'єкту в його CSS- описі. Замість двох описів класів, які відрізняються тільки одним з параметрів, можна створити один опис класу і опис ідентифікатора об'єкту. Опис стилю для об'єкту задається рядком, в якому селектор є ім'ям цього об'єкту з лідируючим символом "#":

```
a.mainlink { color: darkred; text-decoration:
     underline; font-style: italic; }
#blue { color:#003366 }
...
<A CLASS=mainlink>основна гіпертекстове
    посилання</A>
<A CLASS=mainlink ID=blue>модифіковане
    гіпертекстове посилання</A>
```

Слід зазначити, що інтерпретація ідентифікаторів об'єктів в Internet Explorer і Netscape Navigator різна. Існує ще атрибут name у елементу розмітки. При ідентифікації об'єкту Netscape Navigator зазвичай має справу саме з цим атрибутом, а Internet Explorer — з атрибутом ID.

Відмінності в інтерпретації ID у браузерах при декларативному використанні CSS не дуже страшні. Інша справа, якщо автор наважиться програмувати стилі, тобто змінювати значення атрибутів описувачів стилів. В цьому випадку різниця об'єктних моделей документів в Netscape Navigator і Internet Explorer проявиться повною мірою. Фактично, доведеться для кожного з браузерів розробляти абсолютно різні сторінки.

Наслідування і перевизначення в CSS

При обговоренні технічних специфікацій часто буває корисно вникнути в сенс назви. У назві прийнято точно визначати суть і призначення стандарту або специфікації. Опис стилів відображення елементів HTML— розмітки носить назву "Каскадні таблиці стилів". Із словом "стилів" все більш-менш зрозуміло. Під словом "таблиці" слід розуміти набір властивостей елементу розмітки, який можна представити у вигляді рядка в таблиці властивостей, тобто елементи розмітки — рядки, а властивості — стовпці. А ось слово

"каскадні" вимагає пояснення.

По-перше, існує ієрархія елементів розмітки(дерево об'єктів на сторінці). По-друге, властивості цих об'єктів можуть наслідувати. Таким чином в дереві об'єктів утворюється гілка, яка веде до листа дерева — елементу розмітки, наприклад, елементу списку або параграфу. Його властивості визначаються елементами розмітки, в які вкладений елемент, і описувачами стилю для цього елементу:

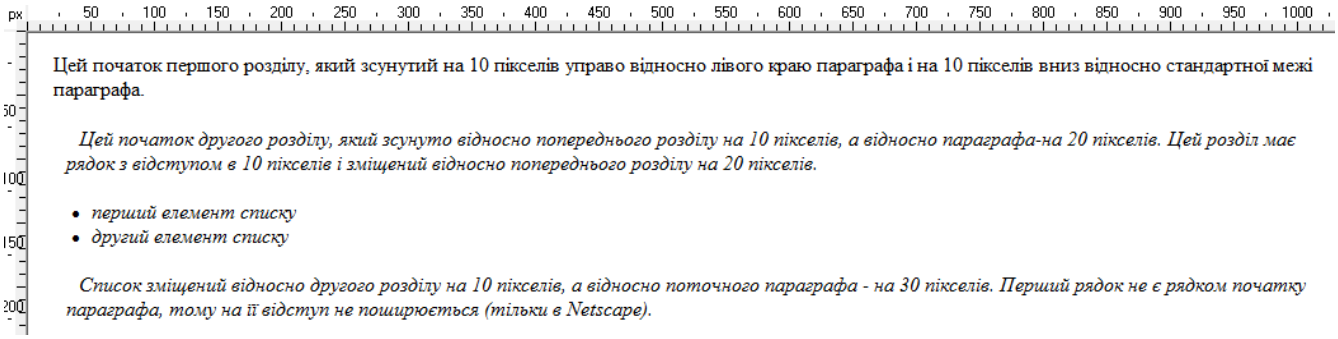


Рисунок 3.2. Демонтрація перевизначення

Попередній текст закодований в термінах розділів і списку таким чином:

```
    <DIV STYLE="margin-left: 10px;margin-top: 10px;">
     Цей початок першого розділу, який зсунутий на 10
пікселів управо відносно лівого краю параграфа і на 10
пікселів вниз відносно стандартної межі параграфа.
    <DIV STYLE="margin-left: 10px;margin-top: 20px;
     text-indent: 10px;font-style: italic;">
    Цей початок другого розділу, який зсунуто відносно
попереднього розділу на 10 пікселів, а відносно
параграфа-на 20 пікселів. Цей розділ має червоний рядок
з відступом в 10 пікселів і зміщений відносно
попереднього розділу на 20   пікселів.
    <UL STYLE="margin-left: 10px;">
    <LI>перший елемент списку
    <LI>другий елемент списку
    </UL> Список зміщений відносно другого розділу на
10 пікселів, а відносно поточного параграфа — на 30
пікселів. Перший рядок не є рядком початку параграфа,
тому на її відступ не поширюється (тільки в Netscape).
    </DIV>
    </DIV>
```

Таким чином відступи відлічуються відносно елементу, в який вкладений поточний елемент. Усі параметри, які не були перевизначені в поточному елементі, наслідують із старшого за ієрархією елементу. Останнє добре продемонстроване в застосуванні стилів відображення списку, який вкладений в розділ і тому відображається курсивом.

Коли пояснення деякого феномену HTML- розмітки розтягується на декілька параграфів, має сенс скористатися приведеною нижче графічною схемою побудови сторінки.

При використанні стилів діють наступні правила старшинства стилів :

- спочатку застосовуються стилі браузеру за умовчанням;
- стилі браузеру за умовчанням перевизначаються прилінкованими стилями(елемент LINK заголовка документу);
- прилінковані стилі перевизначаються описами стилів в елементі STYLE ;
- стилі елементу STYLE перевизначаються атрибутом STYLE у будь-якому з елементів розмітки.

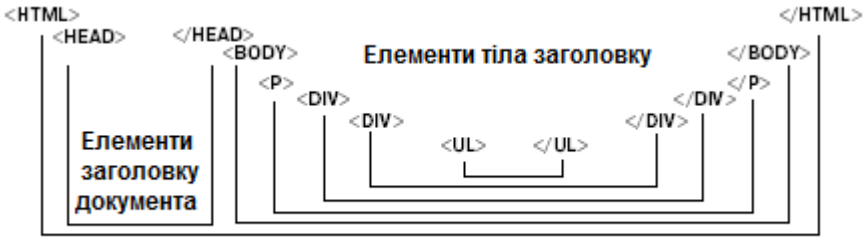


Рисунок 3.3 Правила старшинства стилів

Не усі атрибути стилю можуть наслідуватись. Наприклад, "набивання"(відступ змісту елементу від його меж) елементу BODY не наслідує вкладеними в нього елементами і визначається за умовчанням або прописується для кожного елементу окремо. Алгоритми спадкоємства в Internet Explorer і в Netscape Navigator різні, тому для єдності відображення елементів слід прописувати стиль по максимуму атрибутів.

3.2 Призначення та способи застосування CSS

Каскадні таблиці стилів(CSS) в першу чергу описують властивості тексту. Це стосується як текстових блоків, так і строкових елементів розмітки змісту сторінки. У цьому розділі мова піде про управління відображенням кольору тексту(color) і кольору фону(background-color), на якому відображається текст.

Окрім кольору тексту і кольору фону CSS дозволяє визначати колір межі текстового блоку(border-color).

Атрибути стилів, які ми збираємося розглянути, згідно специфікації Microsoft, відносяться до групи атрибутів Color and Background Properties. Всього до цієї групи входить сім атрибутів, шість з яких визначають властивості фону. Окрім кольору фону і його прозорості, можна управляти фоновою картинкою(координатами її розміщення і способами повторення). На жаль, Netscape Navigator більшість з цих атрибутів не підтримує, тому ми не розглядатимемо їх детально.

Інтерпретація атрибутів кольору в Netscape Navigator і Internet Explorer

різна. У Netscape Navigator фоновий колір відображається тільки там, де є текст, а в Internet Explorer фоновий колір заливає увесь блок або строковий елемент незалежно від наявності в нім тексту.

Колір тексту

У HTML для управління кольором тексту, що відображається, використовується елемент FONT. Його аналогом в CSS є атрибут color. Цей атрибут можна застосовувати як для блокових, так і для строкових елементів розмітки.

Розглянемо в якості блокового елементу розмітки елемент таблиці :

```
<table border=0 bgcolor=white>
<tr>
<th valign=top>Перша колонка</th>
</tr>
<tr>
<td style="color: darkred;">
Перший елемент таблиці
</td>
</tr>
```

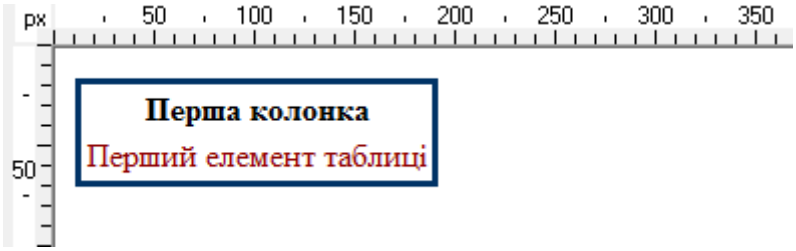


Рисунок 3.4 Блоковий елемент розмітки для елементу таблиці.

У цьому прикладі колір тексту визначений тільки для звичайної комірки, тому зміст заголовка колонки відображається основним кольором(#003366).

При визначенні кольору тексту для блокового елементу увесь текст цього елементу відображається заданим кольором. Часткова зміна кольору можлива, якщо помістити строковий елемент розмітки всередину блочного:

```
P { color: darkred; }
I { color:#003366;font-style: normal; }
```

Наприклад:

```
<table bgcolor=#003366>
<tr> <td>
<table border=0 bgcolor=white>
<tr> <th>
Перша колонка
</th>  </tr>
<tr>  <td>
```

```
<p style="color: darkred;"> Перший елемент
   таблиці.<br> У неї вставляємо  рядковий елемент
   <i style="color:#003366;font-style: normal;">
   темно-синього</i> кольору. </p>
</td>  </tr>   </table>
```

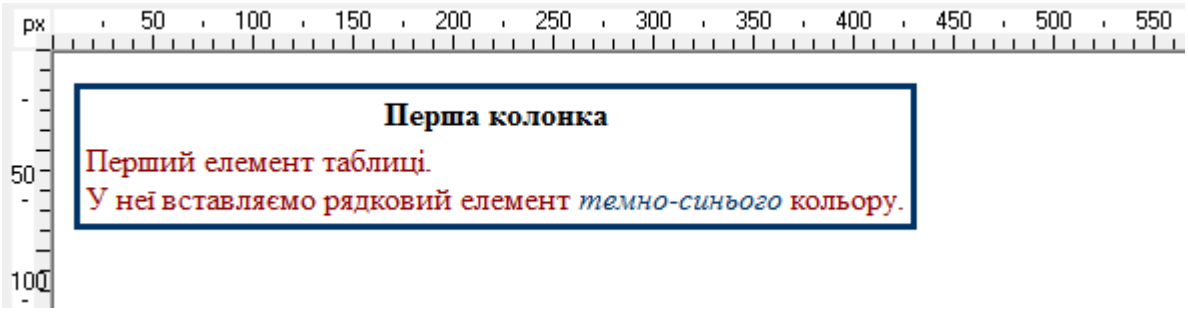


Рисунок 3.5. Часткова зміна кольору тексту усередині блокового елементу

У цьому прикладі як блоковий елемент ми використовуємо параграф, а в якості строкового елементу(in-line) вживаний I. Таблиця в даному випадку великого значення не має, але використовується для одноманітності з попереднім прикладом. У неї ми поміщаємо параграф зі вбудованим в нього in-line елементом розмітки.

Колір фону тексту

У HTML кольором фону можна управляти тільки для конкретного блокового елементу розмітки. Таким елементом може бути уся сторінка:

```
<BODY BGCOLOR=...>...</BODY>
```

Або, наприклад, таблиця:

```
<TABLE BGCOLOR=...>...</TABLE>
```

У наведеному нижче прикладі для виділення тексту застосовано інвертування кольору фону і кольору тексту :

```
Каскадні таблиці стилів дозволяють міняти колір
фону тексту безпосередньо на місці, так само <SPAN
STYLE="background-color: black;color: white;">як
строкові елементи розмітки</SPAN> у HTML міняють
звичайний стиль зображення на <i>italic</i>.
```

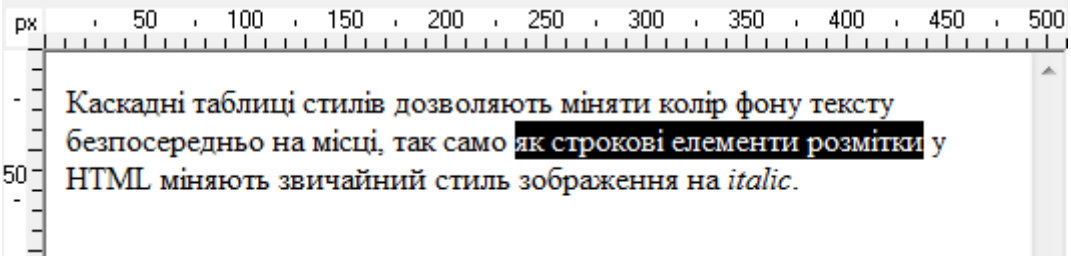


Рисунок 3.6. Інвертування кольору фону і кольору тексту для виділення тексту.

При використанні кольору фону слід пам'ятати, що підтримка цього

атрибуту реалізована для усіх блокових елементів розмітки тільки в Internet Explorer 4.0. Підтримка CSS у версіях Netscape Navigator набагато скромніша.

Для роботи з фоном елементів існує декілька атрибутів, які підтримуються тільки в Internet Explorer, починаючи з версії 4.0: background-image ; background-repeat ; background-attachment ; background-position. Усі властивості фону можна описати в атрибуті background :

```
background: transparent|color url repeat
            scroll position
```

Шрифт

Шрифтам в комп'ютерній графіці завжди приділялося багато уваги, і World Wide Web не є виключенням. Але усе багатство і різноманітність існуючих шрифтів для російської мови обмежене фактично трьома шрифтами: serif(зазвичай Times або інший шрифт із зарубками), sans-serif(Arial, Helvetica або інший шрифт без зарубок) і monospace(Courier). Якщо бути точним, то тут перераховані сімейства шрифтів. Звичайне кожне з цих сімейств представлено тільки одним кирилічним шрифтом.

Автор документу для управління відображенням букв може застосувати декілька атрибутів, що впливають на шрифт :

font-family-сімейство зображень шрифту(гарнітура);

font-style-пряме зображення або курсив;

font-weight-"посилення"(насиченість) шрифту, "жирність" букв;

font-size-розмір шрифту(кегль). Задається в пікселах(px) і друкарських пунктах(pt) .

font-variant-варіант зображення(звичайний або дрібними буквами-капітель).

Усі ці параметри можна поєднати в одному атрибуті font :

```
font: bold 12pt sans;
```

Правда, немає ніякої упевненості в тому, що останнє визначення шрифту працюватиме в усіх браузерах.

При використанні різних гарнітур(font-family) слід пам'ятати, що наявність або відсутність необхідної авторові гарнітури цілком залежить від переваг користувача. Для кирилиці це може вилитися в появу абракадабри там, де автор застосовує відсутні у користувача шрифти.

Найнеприємніше, з чим можна зіткнутися при використанні шрифтів-ця невідповідність моноширинних шрифтів, які застосовуються в HTML-формах. Зворотний зв'язок з користувачем в цьому випадку неможливий.

Специфікація CSS передбачає перерахування шрифтів в описах стилів, що дозволяє частково розв'язати проблему підбору шрифту. На жаль, в Unix і Windows шрифти не погоджені. Фактично, при розробці сторінок в CSS використовуються тільки класи шрифтів(serif, sans-serif і monospace).

Гарнітура(font-family)

Гарнітура шрифту-це набір зображень одного шрифту . Шрифт може мати "пряме" зображення(normal), курсив(italic), "скошене"(oblique),

посилене по насиченості("жирне", bold), "дрібне"(капітель, small-caps) і тому подібне

Найбільш поширені гарнітури в російській частині Web-це Times, Arial, Courier. Причому усі вони належать до різних груп шрифтів. Times-це пропорційний шрифт "із зарубками"(serif), Arial-це пропорційний шрифт "без зарубок"(sans-serif), а Courier-це моноширинний шрифт(monospace). У Unix замість Arial частіше застосовується Helvetica.

У чому різниця між цими групами шрифтів, можна показати на прикладі:

```
    <p align=left style="font-size: 24px;font-family:
serif;color: darkred;"> Цей рядок набраний пропорційним
шрифтом із зарубками.</p>
    <p align=left style="font-size: 24px;font-family:
sans-serif;color: darkred;"> Цей рядок набраний
пропорційним шрифтом без зарубок.</p>
    <p align=left style="font-size: 24px;font-family:
monospace;color: darkred;"> Цей рядок набраний
моноширинним шрифтом.</p>
```

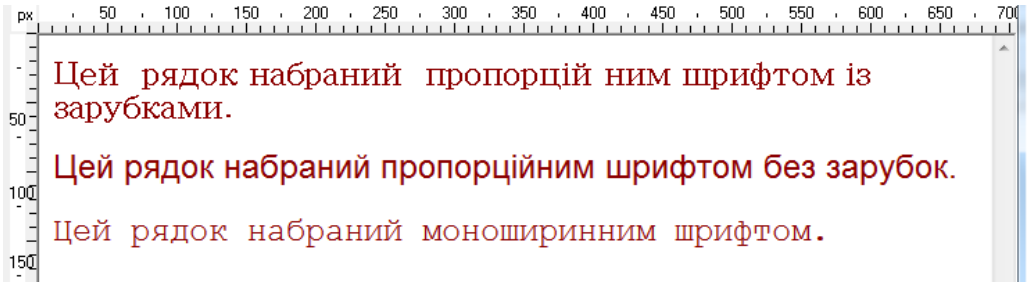


Рисунок 3.7. Демонстрація груп шрифтів.

При вказівці імені групи шрифтів, як показано вище, браузер підбирає відповідний для відображення шрифт цієї групи з наявного набору шрифтів.

Якщо оптимізація браузеру автора сторінки не влаштовує, можна вказати безпосередньо ім'я гарнітури шрифту :

```
<SPAN STYLE="font-family: symbol;
      padding-left: 65px;">
```

Кегль(font-size)

Кегль-це, якщо говорити спрощено, розмір шрифту . Детальніше пояснення слід шукати в спеціальній літературі. Нам досить знати, що CSS через параметр font-size дозволяє управляти розміром букв.

Розмір шрифту можна задавати в друкарських пунктах(pt, 0,35 мм) або пікселах(px). При установці кегля слід пам'ятати, що font-size задає не висоту букви, а розмір "комірки" під букву, який більше самої букви.

Ось декілька прикладів використання font-size:

```
<P STYLE="font-size: 12pt;">
```

```
Кегль параграфа встановлений в 12 пунктів</P>
<P STYLE="font-size: 12px;">
Кегль параграфа встановлений в 12 пікселів</P>
<P STYLE="font-size: 120%;">
Кегль параграфа встановлений в 120% від розміру
букв елементу, що охоплює параграф</P>
```

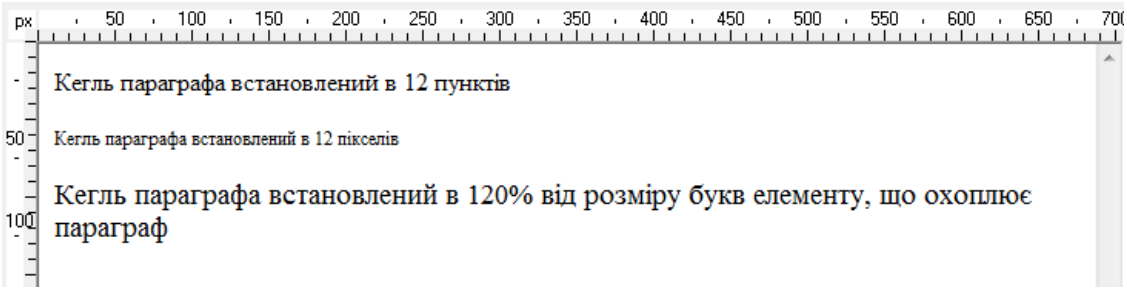


Рисунок 3.8. Установка розміру кегля в абсолютних і відносних одиницях виміру.

Як видно з останнього прикладу, кегль можна задавати не лише в абсолютних одиницях, але і у відносних. Окрім відсотків, існує ще декілька умовних одиниць виміри кегля, які можна застосовувати в CSS :

```
<P STYLE="font-size: large;">
Розмір кегля large</P>
<P STYLE="font-size: small;">
Розмір кегля small</P>
<P STYLE="font-size: x-small;">
Розмір x-small</P>
<P STYLE="font-size: xx-small;">
Розмір кегля xx-small</P>
```

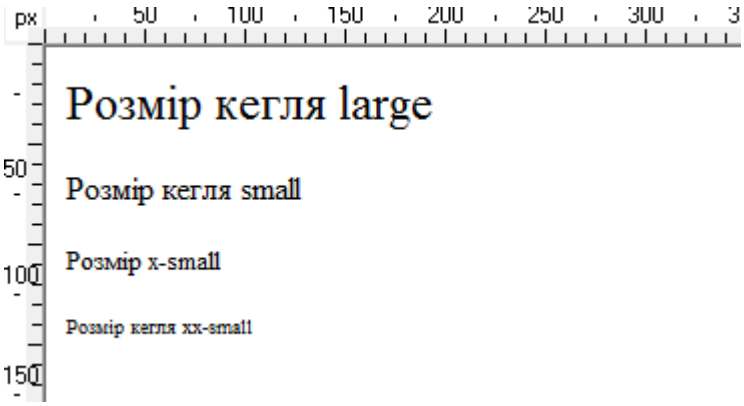


Рисунок 3.9. Установка розміру кегля в умовних одиницях виміру.

Аналогічно x-small і xx-small, існують розміри x-large і xx-large. Крім того, є larger, smaller і medium.

Зображення шрифта

У кожної гарнітури(font-family) є декілька зображень. Кожне з них визначається в CSS трьома параметрами стилю : font-style, font-variant, font-

weight.

Атрибут стилю font-style визначає пряме зображення (normal) і курсив :

```
<P STYLE="color: darkred;font-style: normal;">
Простий стиль</P>
<P STYLE="color: darkred;font-style: italic;">
Курсив</P>
<P STYLE="color: darkred;font-style: italic;
   font-weight: bold;">Курсив</P>
```

Рисунок 3.10. Зображення гарнітур шрифтів.

Якщо хочеться посилити насиченість("жирність") шрифту, то в описі стилю вказують атрибут font-weight, який набуває значень normal або bold :

Хоча шрифт і масштабується за допомогою зміни кегля, якість його зображення при цьому зазвичай страждає. Для якісного відображення дрібних букв в деяких гарнітурах є присутнім зображення капітель. У CSS для використання капітелі зарезервований атрибут font-variant, який набуває значень normal і small-caps. На практиці застосування font-variant проблематично через відсутність капітелі в стандартному наборі кирилічних шрифтів.

Контрольні питання до теми 3

1. Що таке CSS?
2. Як створити новий CSS-стиль?
3. Що таке властивість стилю? Як визначити цей параметр?
4. Як змінити стиль елемента? Навести приклад.
5. Які є способи використання стилів?
6. Що таке наслідування і пере визначення силів?
7. Як сворити селектор-ім’я класу?
8. Як сворити селектор-ідентифікатор об’єкту?
9. Як задати колір тексту і фону?
10. Яким способом задаються атрибути шрифтів у CSS?
11. Назвати загальні правила визначення й присвоєння стильових властивостей, передбачені CSS. Приклади.
12. Синтаксис селекторів. Типи селекторів.
13. Як задаються коментарі?

Тема 4. Використання блочної верстки веб-сторінок

4.1 Блокові і рядкові елементи

У описі елементів розмітки мови HTML існує поняття строкового(in-line) елементу розмітки і блокового(block) елементу розмітки. Формально вони визначені в DTD SGML- описи мови HTML. Пояснити відмінність між блочним і строковим елементами можна на прикладі:

параграф — це блоковий елемент розмітки;

виділення курсивом — це строковий елемент розмітки.

Блокові елементи можна вкладати один в одного, але вони не повинні перетинатися. Строкові елементи можна як вкладати, так і перетинати(згідно з DTD і практиці старих версій браузерів), але останнє робити не рекомендується.

Очевидно, що по набору атрибутів управління відображенням(атрибути опису стилю) строкові і блокові елементи відрізняються. Спрощено можна сказати, що атрибути опису стилю строкового елементу є підмножиною атрибутів опису стилю блокового елементу.

Узагальненнями блочного і строкового елементів, з точки зору стилів, являються елементи DIV і SPAN, відповідно.

DIV грає роль універсального блоку. Блоковий елемент завжди відокремлений від інших елементів сторінки(контексту) порожнім рядком. DIV не несе ніякого смислового навантаження. Часто говорять, що DIV — це розділ сторінки. Але насправді його застосування має сенс тільки в контексті CSS. Ніяких правил за умовчанням для відображення DIV не існує. Це просто новий рядок тексту.

DIV дозволяє застосувати атрибути стилю, пов'язані з межею блоку і відступами блоку від меж старшого елементу, а також "набивання", тобто відступ від межі блоку до межі вкладеного елементу :

```
<DIV STYLE="margin: 20px;padding: 10px;">
Блоковий елемент, заданий елементом розмітки
DIV.
<P>Для нього визначена межа і відступи як
від меж старшого елементу розмітки, так і
для вкладених в нього елементів розмітки.</P>
</DIV>
```

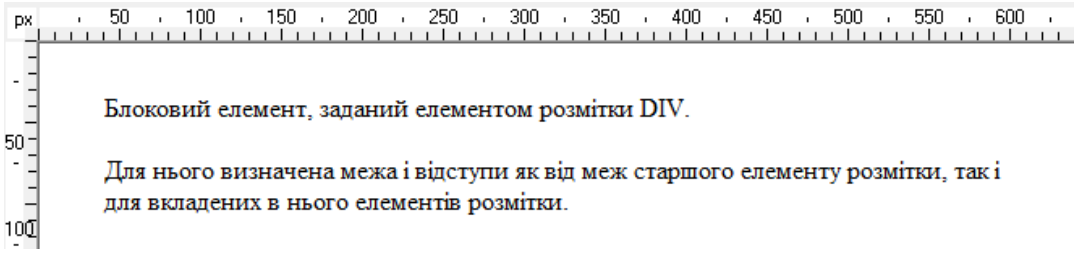


Рисунок 4.1. Відображення блокового елементу розмітки

У цьому прикладі усередині вікна браузеру розташований блоковий елемент(DIV), всередину якого поміщений ще один блоковий елемент(P). DIV має білий фон і межу.

Якщо текст буде видимий браузерами, не підтримувальними CSS, елемент DIV використати не рекомендується. В цьому випадку краще застосувати параграф або інший відповідний по сенсу елемент розмітки із стандартного набору HTML.

Елемент розмітки SPAN — це узагальнений строковий елемент розмітки, застосування якого не призводить до утворення блоку тексту. Він може замінити елементи FONT, I, B, U, SUB, SUP і тому подібне. Наведемо приклади таких відповідностей :

Таблиця 2.1.

HTML- елемент	**CSS- аналог**
<FONT COLOR=red> ...</FONT>	<SPAN STYLE="color: red; ">...</SPAN>
<I>...</I>	<SPAN STYLE="font-style: italic; ">...</SPAN>
<B>...</B>	<SPAN STYLE="font-weight: bold; ">...</SPAN>
<U>...</U>	<SPAN STYLE="text-decoration: underline; ">...</SPAN>
	і тому подібне

У нових версіях браузеру Netscape опису строкових стилів перетинатися не повинні. Тег кінця елементу строкового типу закриває найближчий елемент, а не той, який відкритий тегом початку цього строкового стилю. Також і у разі застосування елементу SPAN, де тег кінця можна співвіднести тільки з найближчим тегом початку елементу SPAN :

```
<B>Речення <I>із стилями, що </B> перетинаються</I>
```

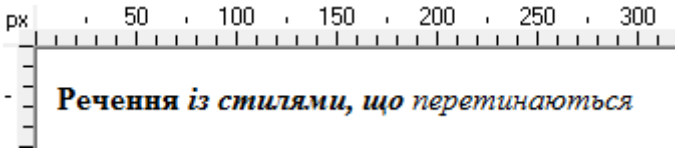


Рисунок 4.2. Перетин стилів

```
<SPAN STYLE="font-weight: bold;">Речення
<SPAN STYLE="font-style: italic;">
із стилями, що </SPAN> перетинаються</SPAN>
```

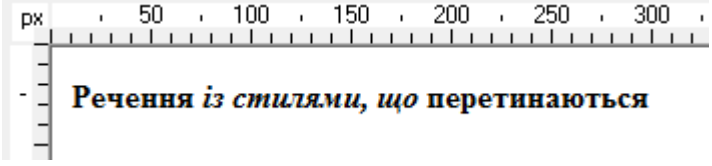


Рисунок 4.2. Вкладення стилів

Застосування елементу SPAN обмежене браузерами, які підтримують CSS. При цьому не усі атрибути специфікації CSS підтримуються у

браузерах. Наприклад, атрибут vertical-align, який покликаний замінити елементи SUP і SUB, може не підтримуватися деякими браузерами.

Властивості блоків

Блокові елементи(блоки тексту або box) дозволяють оперувати з текстом в термінах прямокутників, які цей текст займають. При цьому блок тексту стає елементом дизайну сторінки з тими ж властивостями, що і картинка, таблиця або прямокутна область додатка.

Блок тексту має властивості: висоти(height), ширини(width), межі(border), відступу(margin), набивання(padding), довільного розміщення(float), управління обтіканням(clear).

Графічно властивості можна представити таким чином:

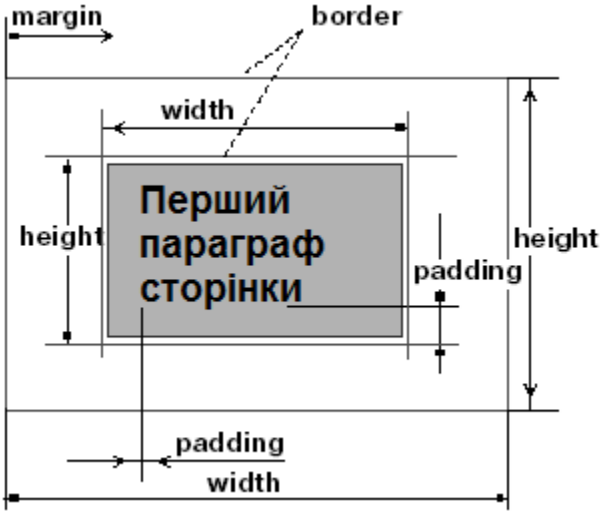


Рисунок 4.3. Властивості блоків

З шириною і висотою блоку тексту усе більш або менш зрозуміло. Задаватися вони можуть в друкарських пунктах(pt), пікселах(px) і умовних одиницях(em) :

```
<DIV STYLE="width: 200px;">пікселі</DIV>
<DIV STYLE="width: 200pt;">друкарські пункти
```

З висотою блоку тексту слід бути обережним, оскільки в четвертій версії Netscape Navigator багато хто з атрибутів CSS не підтримується, у тому числі висота звичайного блокового елементу.

Відстань від межі блокового елементу до межі вкладеного в нього блокового елементу називається padding. У рамках цього курсу лекцій для позначення цієї властивості використовується слово "набивання" або словосполучення "внутрішній відступ" .

Відступ від "набивання" зовнішнього блокового елементу до межі вкладеного елементу називається margin. Для його позначення ми використовуватимемо термін "відступ" або словосполучення "зовнішній відступ" .

Таким чином padding і margin характеризують відступи блокового

елементу відносного початку його змісту і відносно межі елементу розмітки, що охоплює його.

Відступи і "набивання" можуть бути лівими, правими, верхніми і нижніми. CSS дозволяє змінювати будь-які з них.

При відображенні блоку тексту можна показати його видиму межу. CSS дозволяє визначити її стиль, ширину і колір. При використанні видимої межі слід зважати на специфіку браузерів. Одним з можливих способів застосування межі є видиме обмеження "плаваючих" блоків тексту.

"Плаваючий" текстовий блок дозволяє реалізувати можливість обтікання цього блоку текстом.

Притиснемо блок тексту управо. Ліворуч його обтікатиме інший текст.

Обтікання одного тексту іншим відбувається в тому ж ключі, що і Обтікання текстом картинки або таблиці. Текст блоку, що охоплює, прагне "втиснутися" на вільне місце, залишене "плаваючим" блоком. Коли межа "плаваючого" блоку кінчається, блок, що охоплює, поширюється на усю ширину відведеного для тексту простору.

CSS дозволяє вирівнювати блок тексту не лише по краю сторінки, але і по центру(тільки у Netscape Navigator).

Відступи(margin)

При відображенні блоку тексту на папері навколо нього зазвичай залишають поля. Поля можна задавати або відносно межі сторінки, або відносно самого блоку тексту. У першому випадку ми маємо справу з "відступом"(margin), а в другому — з набиванням(padding). Власне, ширина поля визначатиметься сумою ширини "набивання" і ширини відступу :

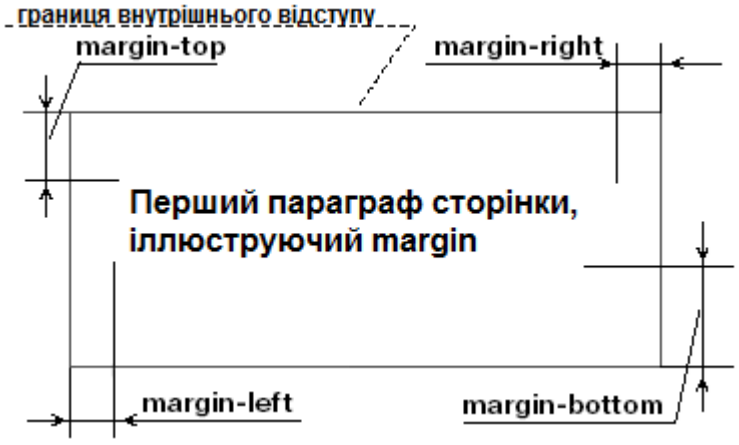


Рисунок 4.4. Види відступів

Зазвичай пунктирна лінія і межа блоку є невидимими лініями. Вони вгадуються по краю тексту, що вирівнюється. Вірніше, вгадується сумарна ширина полів. Стрілки вказують напрям відліку відступу. Padding відлічується від зовнішньої межі блоку всередину блоку, тоді як margin — від зовнішньої межі блоку в область блоку(назовні), що охоплює його.

Зовнішній відступ(margin) може відлічуватися по будь-якому напряму відносно сторін блоку: margin-left, margin-right,margin-top, margin-bottom.

margin — задає загальний зовнішній відступ від усіх сторін блоку тексту. Застосовується у тому випадку, якщо блок тексту рівновіддалений від усіх меж внутрішнього відступу елементу, що охоплює.

При застосуванні зовнішнього відступу слід пам'ятати, що він відлічується від межі елементу до межі внутрішнього відступу("набивання", padding) елементу, що охоплює. Якщо цей факт не враховувати, то загальна ширина видимих полів може виявитися більше, ніж вказано в зовнішньому відступі.

Набивання(padding)

Текст усередині блоку починається не від самої його межі. Між межею і змістом блоку є вільний простір. Воно називається внутрішній відступ текстового блоку або padding. Спільно із зовнішнім відступом(margin) текстового блоку padding утворює загальне поле відступу від межі елементу, що охоплює блок, розмітки.

Padding можна проілюструвати на прикладі лівого внутрішнього відступу тексту в параграфі:

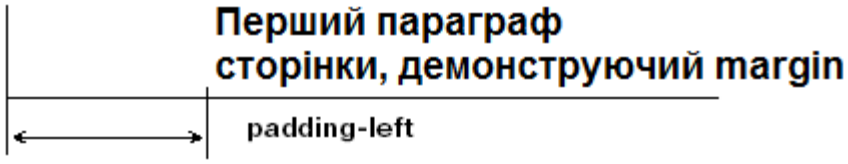


Рисунок 4.5 Демонстрація Padding

Для цього прикладу при описі параграфа використовувався стиль:

```
P { padding-left: 100px;text-align: left;
    border-width: 1px; }
```

Щоб браузер правильно відображав стилі, не слід розміщувати опис стилю на декількох рядках, як це зроблено в прикладі. Для Internet Explorer це не має значення, а Netscape Navigator може "спіткнутися".

У блоку тексту існує чотири сторони. Відповідно, padding може бути:

padding-left, padding-right, padding-top, padding- bottom .

padding — визначає єдиний розмір внутрішнього відступу блоку. Цей параметр задається у разі однакового розміру відступу від усіх сторін блоку.

Проілюструємо застосування padding на прикладі:

```
P { padding-left: 100px;padding-right: 50px;
    padding-top: 20px;padding-bottom: 10px;
    text-align: left;border-width: 1px; }
```

При установці padding слід пам'ятати, що цей параметр задає розмір відступу від межі блоку до межі зовнішнього відступу(margin) змісту блоку. З цієї причини загальний розмір поля може виявитися більше, ніж задано в параметрі padding.

Межа(border)

У кожного блокового елементу розмітки є межа. Від межі відлічуються відступи(margin і padding). Уздовж межі "плаваючого" блоку його обтікає текст.

Для опису меж блоків застосовуються наступні атрибути:

border-top-width - ширина верхньої межі блоку ;

border-bottom-width - ширина нижньої межі блоку ;

border-left-width - ширина лівої межі блоку ;

border-right-width -ширина правої межі блоку ;

border-width - ширина межі блоку. Задається у тому випадку, якщо ширина межі блоку однакова по усьому периметру блоку;

border-color-колір межі блоку. Згідно специфікації CSS1 може бути заданий для кожної з меж блоку. Наприклад, border-right-color: red. Може задаватися як мнемонікою(red, blue, navy і тому подібне), так і в нотації RGB(border-color:#003366). Вказівка кольору для кожної з меж підтримується не усіма браузерами;

border-style-тип лінії межі блоку. Може набувати значень: none, dotted, dashed, solid, double, groove, ridge, inset, outset. Згідно специфікації CSS1, може бути заданий для кожної з меж блоку. Наприклад, border-right-style: dotted. Вказівка типу лінії межі підтримується не усіма браузерами.

Для опису межі немає необхідності вказувати в стилі усі атрибути. Існує скорочений запис атрибутів. Наприклад, для опису верхньої лінії межі можна використати запис типу :

```
P { border-top: 1px dotted red; }
```

Якщо необхідно обмежити блок тексту межею, то це може виглядати приблизно так:

```
P { text-align: left;border-width: 2px;
    border-color: black;border-style: solid; }
```

Застосування межі для позначення блоку-не самий кращий спосіб оформлення документу. В усякому разі, його застосовують нечасто.

Вказуючи межу в Internet Explorer, треба обов'язково визначати її тип, інакше вона не відображатиметься.

Обтікання блоку тексту

Під обтіканням блоку текстом розуміють той же ефект, який можна реалізувати для графіки, коли картинка не розриває блок тексту, а вбудовується в нього. Текст в цьому випадку "обтікає" картинку з одного боку-там, де є вільне поле між межею сторінки(елементу) і картинкою. Обтікання картинки текстом від звичайного вбудовування картинки в текст

документу відрізняється тим, що уздовж вертикальної межі картинки розташовується декілька рядків тексту, а не одна.

Обтіканням блоку тексту іншим текстом управляють два атрибути CSS : float і clear.

Атрибут float визначає "плаваючий" блок тексту. Він може набувати значень:

left-блок притиснутий до лівої межі елементу, що охоплює;

right-блок притиснутий до правої межі елементу, що охоплює;

both-текст може обтікати блок з обох боків.

Проілюструвати обтікання дозволяє наступний приклад:

```
<font color=darkred size=-1>
<span style="float: left;border-color:#003366; border-width: 1px;border-style: solid;font-size: 15px; font-family: monospace;width: 30%">
Цей блок тексту ми притиснули до лівої межі розділу. Він має межу шириною в один піксел і ширину в 30% від ширини розділу.
</span>
Розмір шрифту обтічного блоку тексту навмисно зменшений до 8-ми пікселів, щоб блок тексту виділявся на загальному фоні вмісту сторінки.
</font>
```

При використанні значення right блок тексту буде притиснутий управо.

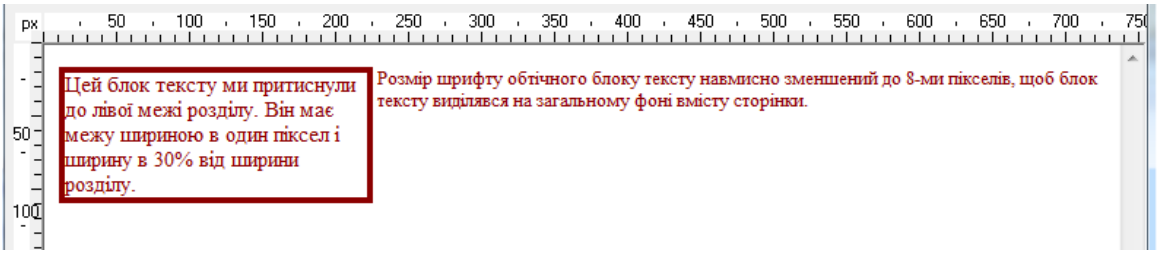


Рисунок 4.6 - Обтікання зправа

Другий атрибут опису стилів clear дозволяє управляти власне обтіканням. Він не допускає наявності "плаваючих" блоків біля блоку тексту. Атрибут може набувати значень: right, left, none, both:

```
<font color=darkred size=+1>
<span style="float: right;border-color:#003366; border-width: 1px;border-style: solid;font-size: 15px;font-family: monospace;width: 30%"> Цей блок тексту ми притиснули до правої межі розділу. Він має межу шириною в один піксел і ширину в 30% від ширини розділу.</span>
<p style='clear: right;text-align: justify;color: darkred'> У цього блоку тексту не може бути
```

"плаваючого" правого блоку, оскільки ми його заборонили. З цієї причини він починається нижче притиснутого управо обмеженого блоку.</p>

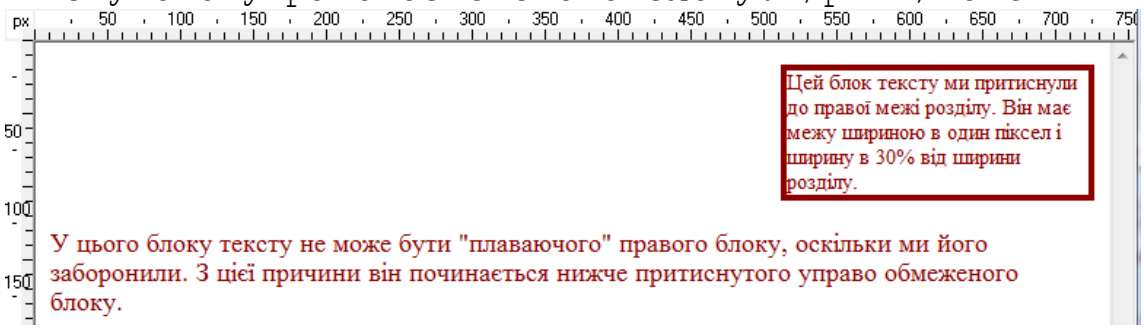


Рисунок 4.7. Застосування атрибуту clear

4.2 Позиціонування елементів

CSS пропонує чотири типи позиціонування.

Абсолютне(position: absolute;). Таке позиціонування дозволяє визначати розташування елементу, задаючи позиції left, right, top або bottom у будь-яких одиницях виміру CSS. Крім того, абсолютно розміщені елементи повністю вилучені з потоку сторінки, певного HTML - кодом: інші елементи на сторінці розміщуються без урахування елементів, що абсолютно позиціонуються.

Відносне(position: relative;). Елемент з таким позиціонуванням розміщується відносно його потокового положення в потоці HTML. Так, наприклад, встановлюючи значення top рівним 20 пікселів і значення left рівним 200 пікселів для відносно розміщеного заголовка, ви перемістите його на 20 пікселів вниз і 200 пікселів ліворуч від того місця, де він з'явився б на сторінці. На відміну від абсолютного позиціонування, тут інші елементи сторінки регулюють старе HTML - розміщення об' єкту, що відносно позиціонується. Відповідно, переміщення об' єкту з відносним позиціонуванням залишає "діру", на місці якої він повинний був знаходитися. Основна користь відносного позиціонування не в тому, щоб перемістити елемент, а в установці нової точки прив'язки для елементів, що абсолютно позиціонуються, які вкладені в нього.

Фіксоване(position: fixed;). Фіксований елемент закріплюється у визначеному місці на екрані. Визначення такого позиціонування грає ту ж роль, що і установка значення fixed для властивості background — attachment при позиціонуванні фонових зображень. Колі відвідувач прокручує сторінку, фіксовані елементи залишаються на екрані як абзаци і заголовки, тоді як фотографії прокручуються разом із сторінкою. Використання фіксованих елементів — відмінний спосіб створити нерухоме бічне меню або відтворити ефект HTML — фреймів, де тільки певна частина сторінки прокручується.

Статичне позиціонування відповідає нормальному потоку HTML.

Параметри позиціонування в CSS

Параметр 'bottom' визначає нижній край елементу.

Примітка: Якщо параметр "position" має значення "static", параметр "bottom" не робить впливу.

Приклади:

У прикладі показано, як задати нижній край елементу pre на 50 px вище за нижній край вікна :

```
pre
{
position: absolute;
bottom: 50px
}
```

У прикладі показано, як задати нижній край елементу pre на 50 px нижче нижнього краю вікна :

```
pre
{
position: absolute;
bottom: - 50px
}
```

Може набувати наступних значень:

Значення	Опис
auto	Дозволяє браузеру самостійно вичислити нижню позицію
%	Задає нижню позицію в % від положення нижнього краю вікна
length	Задає нижню позицію в px, см, і так далі від нижнього краю вікна. Допускаються негативні значення.

Параметр 'clip'

Цей параметр задає форму елементу.

У ситуації якщо, наприклад, зображення більш того елементу в якому розташовується, параметр дозволяє визначити розміри зображення, обрізувати за формою і вивести.

Примітка: Цей параметр не можна використати для елементів з параметром "overflow", заданим як "visible".

Наслідування: немає.

Приклад:

```
p
{
position: absolute;
clip: rect(2px 175px 100px 0px)
}
```

Може набувати наступних значень:

Значення	Опис
shape	Задає форму елементу. Допустимим значенням форми є:

	rect(top, right, bottom, left)
auto	Браузер задає форму елементу

Параметр 'left'

Цей параметр визначає лівий край елементу.

Примітка: Якщо параметр "position" має значення "static", параметр "left" не робить впливу.

Наслідування: немає.

Приклади:

У прикладі показано, як задати лівий край елементу pre в 50 px праворуч від лівого краю вікна :

```
pre
{
position: absolute;
left: 50px
}
```

У прикладі показано, як задати лівий край елементу pre в 50 px зліва від лівого краю вікна :

```
pre
{
position: absolute;
left: -50px
}
```

Може набувати наступних значень:

Значення	Опис
auto	Дозволяє браузеру вичислити ліву позицію.
%	Задає ліву позицію в % від значення для лівого краю вікна.
length	Задає ліву позицію в px, см і так далі від лівого краю вікна. Допускаються негативні значення.

Параметр 'overflow'

Цей параметр визначає, що відбувається, коли вміст елементу переповнює його область.

Наслідування: немає.

Приклад:

```
div
{
overflow: auto
}
```

Може набувати наступних значень:

Значення	Опис
visible	Вміст не обрізується. Воно виводиться за межами елементу.
hidden	Вміст обрізується, але браузер не виводить смугу прокрутки для

	перегляду усього вмісту.
scroll	Вміст обрізується, але браузер виводить смугу прокрутки для перегляду усього вмісту.
auto	Якщо вміст обрізується, то браузер повинен вивести смугу прокрутки для перегляду усього вмісту.

Параметр 'position'

Цей параметр поміщає елемент в статичне, відносне, абсолютне або фіксоване положення.

Наслідування: немає.

Приклад:

```
p
{
position: static;
}
```

Може набувати наступних значень:

Значення	Опис
static	Елемент поміщається в звичайне положення(згідно з нормальним потоком). Для значення "static" параметри "left" і "top" не використовуються.
relative	Переміщає елемент відносно нормального положення, так що "left: 20" додає 20 пікселів до позиції LEFT елементу
absolute	За допомогою значення "absolute" елемент можна розмістити у будь-якому місці сторінки. Позиція елементу визначається параметрами "left", "top", "right", і "bottom"
fixed	

Параметр 'right'

Цей параметр визначає правий край елементу.

Примітка: Якщо параметр "position" має значення "static", то параметр "right" не робить впливу.

Наслідування: немає.

Приклади:

У прикладі показано, як задати правий край елементу pre на 50 px ліворуч від правого краю вікна :

```
pre
{
position: absolute;
right: 50px
}
```

У прикладі показано, як задати праве поле елементу pre на 50 px вправо від правого краю вікна :

```
pre
{
```

```
position: absolute;
right: -50px
}
```

Може набувати наступних значень:

Значення	Опис
auto	Дозволяє браузеру вичислити праву позицію.
%	Задає праву позицію в % від значення правого краю вікна.
length	Задає праву позицію в px, см, і так далі від правого краю вікна. Допускаються негативні значення.

Параметр 'top'

Цей параметр визначає верхній край елементу.

Примітка: Якщо параметр "position" має значення "static", то параметр "top" не робить впливу.

Наслідування: немає.

Приклади:

У прикладі показано, як задати верхній край елементу pre на 50 px нижче верхнього краю вікна :

```
pre
{
position: absolute;
top: 50px
}
```

У прикладі показано, як задати верхній край елементу pre на 50 px вище за верхній край вікна :

```
pre
{
position: absolute;
top: -50px
}
```

Може набувати наступних значень:

Значення	Опис
auto	Дозволяє браузеру вичислити верхню позицію.
%	Задає верхню позицію в % від значення верхнього краю вікна.
length	Задає верхню позицію в px, см, і так далі від верхнього краю вікна. Допускаються негативні значення.

Параметр 'vertical-align'

Цей параметр задає вертикальне вирівнювання елементу.

Наслідування: немає.

Приклади:

```
img
{
```

```
vertical-align: baseline
}
```

Може набувати наступних значень:

Значення	Опис
baseline	Елемент розміщується на базовому рядку батьківського елементу.
sub	Вирівнює елемент як нижній індекс
super	Вирівнює елемент як верхній індекс
top	Вершина елементу вирівнюється з вершиною найвищого елементу в рядку
text - top	Вершина елементу вирівнюється з вершиною шрифту батьківського елементу
middle	Елемент поміщається в середині батьківського елементу
bottom	Нижня частина елементу вирівнюється з самим нижнім елементом в рядку
text - bottom	Нижня частина елементу вирівнюється з мінімальною нижньою точкою батьківського елементу
length	
%	Вирівнює елемент в % від значення параметра "line-height". Допускаються негативні значення.

Параметр 'z-index'

Цей параметр задає порядковий номер елементу в стеку. Елемент з великим порядковим номером стека завжди знаходиться перед елементом з меншим порядковим номером стека.

Примітки:

Елементи можуть мати негативні порядкові номери стека.

Z - index працює тільки з тими елементами, які позиціонувалися не як static(наприклад, position: absolute;)!

Наслідування: немає.

Приклад:

```
img
{
position: absolute
z-index: 1
}
```

Може набувати наступних значень:

Значення	Опис
auto	Порядковий номер елементу в стеку дорівнює номеру батьківського елементу
number	Задає порядковий номер елементу в стеку

Контрольні питання до теми 4

1. Яка різниця між блоковими і рядковими елементами? Приклад.
2. Назвіть теги для формування блокових і рядкових елементів.
3. Назвіть властивості блоків та відповідні атрибути.
4. Яка різниця між атрибутами padding та margin?
5. Як задати обрамлення?
6. Які є способи обтікання тексту? Приклад.
7. Які є види позиціонування?
8. Назвіть основні параметри позиціонування.
9. Параметр накладання та його застосування.
10. Порівняйте позиціонування різних видів.

Тема 5. Основи мови Java Script

5.1 Загальний огляд мови JavaScript.

JavaScript – це мова програмування, що використовується в складі HTML-сторінок для збільшення їх функціональності та можливостей взаємодії з користувачем. JavaScript є однією із складових динамічного HTML. Ця мова програмування була створена фірмами Netscape та Sun Microsystems на базі мови програмування Sun's Java. На сьогодні є декілька версій JavaScript. Однією із найбільш поширених є версія JavaScript 1.3. За допомогою JavaScript на HTML-сторінці можливо зробити те, що не можливо зробити за допомогою стандартних тегів HTML.

Код програми JavaScript розміщується або в середині HTML-сторінки, або в текстовому файлі, що пов'язаний за допомогою спеціальних команд з HTML-сторінкою. Цей код, як правило, розміщується в середині тегу HTML та завантажується в браузер разом з кодом HTML-сторінки. Програма JavaScript не може існувати самостійно, тобто без HTML-сторінки.

Виконання програми JavaScript відбувається при перегляді HTML-сторінки в браузері, звісно, тільки в тому випадку, коли браузер містить інтерпретатор JavaScript. Практично всі сучасні популярні браузери оснащені таким інтерпретатором. Відзначимо, що крім JavaScript на HTML-сторінках можливо використовувати інші мови програмування. Наприклад, VBScript або JScript, яка є варіантом JavaScript від фірми Microsoft. Але виконання програм VBScript та JScript гарантовано коректне тільки при перегляді HTML-сторінки за допомогою браузеру Microsoft Internet Explorer. Тому в більшості випадків використання JavaScript доцільніше, хоча функціональність програм VBScript та JScript дещо краща.

Досить часто програму JavaScript називають *скриптом* або *сценарієм*. Скрипти виконуються в результаті того, що відбулась деяка подія, пов'язана з HTML-сторінкою. В багатьох випадках виконання вказаних подій ініціюється діями користувача.

Скрипт може бути пов'язаний з HTML-сторінкою двома способами:

За допомогою парного тегу SCRIPT;

Як оброблювач події, що стосується конкретного тегу HTML.

Сценарій, вбудований в HTML-сторінку з використанням тегу SCRIPT, має наступний формат:

```
<SCRIPT>
   // Код програми
</SCRIPT>
```

Все, що розміщується між тегами <SCRIPT> та </SCRIPT>, інтерпретується як код програми на мові JavaScript. Обсяг вказаного коду не обмежений. Інколи скрипти розміщують в середині HTML-коментарію. Це роблять для того, щоб код JavaScript не розглядався старими браузерами, які не мають інтерпретатора JavaScript. В цьому випадку сценарій має формат:

```
<SCRIPT>
    <!--
   // Код програми
    -->
</SCRIPT>
```

Тег SCRIPT має декілька необов'язкових параметрів. Найчастіше використовуються параметри *language* та *src*. Параметр *language* дозволяє визначити мову та версію мови сценарію. Параметр src дозволяє задати файл з кодом сценарію. Для пояснення використання параметрів тегу SCRIPT розглянемо задачу.

Задача. Необхідно для HTML-сторінки hi.htm створити сценарій на мові JavaScript 1.3 для показу на екрані вікна повідомлення з текстом "Привіт!".

Відзначимо, що для показу на екрані вікна повідомлення можливо використати функцію alert.

Для ілюстрації можливостей пов'язування скриптів з HTML-кодом вирішення задачі реалізуємо двома варіантами.

Варіант 1. Визначення сценарію безпосередньо на HTML-сторінці hi.htm

```
<html><head>
    <title>Використання JavaScript</title>
</head>
<body>
<script language="JavaScript1.3">
   alert('hi');
</script>
</body></html>
```

Варіант 2. Визначення сценарію в файлі a.js, пов'язаному з HTML-сторінкою hi.htm за допомогою параметру src тегу SCRIPT. Код HTML-сторінки hi.htm:

```
<html><head>
    <title>Використання JavaScript</title>
    <script language="JavaScript1.3"
            src="a.js"> </script>
</head><body>
</body></html>
```

Програмний код, записаний в файлі a.js:

```
alert('hi');
```

Результат виконання обох варіантів вирішення задачі однаковий і показаний на рисунку 6.1.

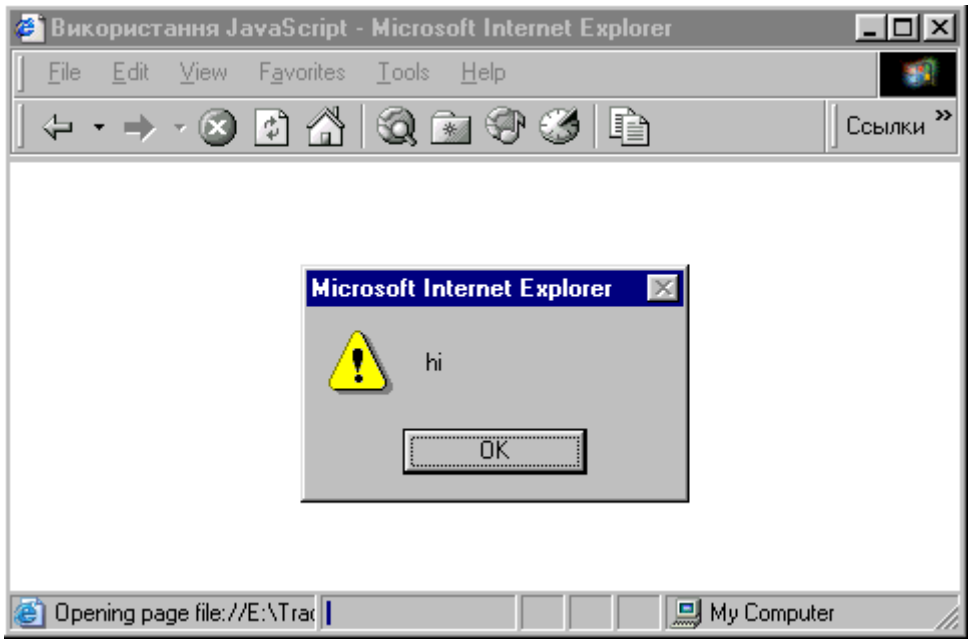


Рисунок 6.1 Показ вікна повідомлення засобами JavaScript

Змінні та вирази JavaScript можливо використовувати в якості значень параметрів тегів HTML. В цьому випадку елементи JavaScript розміщуються між амперсандом (&) та крапкою з комою (;), але повинні бути обмежені фігурними дужками {} і використовуватись тільки в якості значень параметрів тегів.

Наприклад, нехай визначена змінна *c* і їй присвоєно значення *green*. Наступний тег буде виводити текст зеленого кольору:

```
<font color="&{c};"> текст зеленого кольору </font>
```

Відзначимо, що мінімальним комплектом програмного забезпеченням для розробки та тестування програм JavaScript є текстовий редактор та браузер з підтримкою JavaScript.

Синтаксис. Визначення та ініціалізація змінних

Сценарій JavaScript являє собою набір операторів, що послідовно інтерпретуються браузером. Оператори можливо розміщувати як в одному, так і в окремих рядках. Якщо оператори розміщені в одному рядку, то між ними необхідно поставити ;. В протилежному випадку ; не обов'язкова. Будь-який оператор можливо розмістити в декількох рядках без символу продовження.

Будь-яка послідовність символів, розміщених в одному рядку, якій передує //, розглядається як *коментар*. Для визначення *багаторядкових коментарів* використовується конструкція:

```
/*
Багаторядковий коментар
*/
```

В мові JavaScript рядкові та приписні букви вважаються різними символами. JavaScript використовує змінні для зберігання даних визначеного типу. При цьому JavaScript є мовою з вільним використанням типів. Тобто не

обов’язково задавати тип змінної, який залежить від типу даних, що в ній зберігаються. При зміні типу даних автоматично змінюється і тип змінної.

JavaScript підтримує чотири простих типи даних. Ілюстрацією цього є табл. 6.1. Для присвоєння змінним значень основних типів використовуються *літерали*.

Таблиця 6.1 Типи даних JavaScript

Тип даних	Пояснення	Приклад літерала
Цілий	Послідовність цілих чисел	789 +456 -123
З плаваючою крапкою	Числа з крапкою, яка відділяє цілу частину від дробової, або числа в науковій нотації	7.25 0.525e01 71.2E-4
Рядковий	Послідовність алфавітно-цифрових символів, взятих в одинарні (’), або подвійні (") лапки.	"Привіт" "234" "Hello World!!!"
Булевий або логічний	Використовуються для оброблення ситуацій так/ні в операторах порівняння	true false

Ім’я змінної повинно містити тільки букви латинського алфавіту, символ підкреслення _, арабські цифри та починатись з букви або символу підкреслення _. Довжина імені повинна бути менша від 255 символів. Заборонено використовувати імена, що збігаються з ключовими словами JavaScript. Приклади імен: _hello, go, go123.

Визначити змінну можливо:

Оператором *var*, наприклад, var a;

При *ініціалізації*, за допомогою оператору присвоєння (=), наприклад, b=126.

Визначення та ініціалізацію змінних можливо реалізувати в будь-якому місці програми.

Вирази та оператори

Вираз – це комбінація змінних, літералів та операторів, в результаті обчислення яких можливо отримати тільки одне значення, яке може бути числовим, рядковим або булевим.

Для реалізації обчислень в JavaScript використовуються арифметичні, рядкові, логічні вирази та декілька типів операторів.

Арифметичні вирази – обчислюють число, наприклад, a=7+5;

Рядкові вирази – обчислюють рядок символів, наприклад, "Джон" або "234";

Логічні вирази – обчислюють true (істина) або false (хибна).

Оператор присвоювання (=) – присвоює, значення лівому операнду, базуючись на значенні правого операнда. Наприклад, для присвоєння змінній *a*

значення числа 5 необхідно записати:

a=5

До стандартних *арифметичних операторів* відносяться: оператори додавання (+), віднімання (-), множення (*), ділення (/), остача від ділення чисел (%), збільшення числової змінної на 1 (++), зменшення числової змінної на 1 (--).

Відзначимо, що оператор додавання можна використовувати не тільки для чисел, але й для додавання (контрактації / конкатенації) текстових рядків.

Для створення логічних виразів використовуються *логічні оператори* та *оператори порівняння*.

До логічних операторів відносяться – логічне *І* (&&), логічне *АБО* (||), логічне *НІ* (*!*).

Оператори порівняння не відрізняються від таких операторів в інших мовах програмування. До операторів порівняння відносяться (==, >, >, =<, <=,*!*=).

Оператори вибору

Оператори вибору відносяться до *операторів управління,* призначенням яких є зміна напрямку виконання програми. Крім операторів вибору до операторів управління відносяться: оператори циклу та оператори маніпулювання об'єктами.

Оператори вибору призначені для виконання деяких блоків операторів в залежності від істинності деякого логічного виразу. До операторів вибору відносяться: оператор умови *if...else* та перемикач *switch*.

Синтаксис оператора умови такий:

```
if (умова) {
    група операторів 1
    . . . . . .
}
[else] {
    група операторів 2
    . . . . . .
}
```

Перша група операторів виконується при умові істинності виразу умова. Необов'язковий блок *else* визначає другу групу операторів, яка буде виконуватись в випадку хибності умови, заданої в блоці if. В середині групи операторів можуть бути використані будь-які інші оператори, в тому числі і інші оператори умови. Це дозволяє створювати групу вкладених операторів умови if та реалізовувати складні алгоритми перевірки. Однак, якщо кількість вкладених операторів if більша ніж три, то програма стає складною для розуміння. В такому випадку доцільно використовувати оператор *switch.* В цьому операторі обчислюється деякий вираз та порівнюється з значенням, заданим в блоках *case*. Синтаксис оператора switch такий:

```
switch  (вираз) {
case значення1:
```

```
[оператори1]
break;
case значення2:
[оператори2]
break;
...
default:
[оператори]
}
```

Якщо значення виразу в блоці *switch* дорівнює *значення1*, то виконується група операторів *оператори1*, якщо дорівнює *значення2*, то виконується група операторів *оператори2* і так далі. Якщо значення виразу не дорівнює ні одному із значень, що задані в блоках *case*, то обчислюється група операторів блоку *default*, якщо це блок заданий, інакше – виконується вихід із оператору *switch*. Необов'язковий оператор *break,* який можливо задавати в кожному із блоків *case*, виконує безумовний вихід із оператору *switch*. Якщо він не заданий, то продовжується виконання операторів в наступних блоках *case* до першого оператору *break* або до кінця оператору *switch*.

Оператори циклу

Цикл – це деяка група команд, що повторюється доки вказана умова не буде виконана. JavaScript 1.3 підтримує дві форми циклу: *for* та *while*. Крім того оператори *break* та *continue* використовуються разом з циклами.

Цикл for повторює групу команд до тих пір, доки вказана умова хибна. Синтаксис оператору for такий:

```
for([initial-expression];[condition];[increment-expression])
{
   statements
}
```

Виконання циклу *for* проходить в такій послідовності:

1. Вираз *initial-expression* служить для ініціалізації змінної лічильника. Цей вираз розраховується один раз на початку виконання циклу

2. Вираз *condition* розраховується на кожній ітерації циклу. Якщо значення виразу *condition* дорівнює true, виконується група операторів *statements* в тілі циклу. Якщо значення виразу *condition* дорівнює false, то цикл for закінчується. Якщо вираз *condition* пропущено, то він вважається рівним true. В цьому випадку цикл продовжується до оператора break.

3. Вираз *increment-expression* використовується для зміни значення змінної лічильника.

4. Розраховується група операторів *statements* та реалізується перехід на наступну ітерацію циклу, тобто на крок 2.

Приклад. Цикл для розрахунку суми цілих чисел від 1 до 100.

```
s=0
for (i=1;i<101;i++) {
```

```
    s=s+1;
}
```

Оператор *while* повторює цикл, доки вказана умова істина. Оператор while виглядає таким чином:

```
 while (condition) {
    statements
 }
```

Цикл *while* виконується таким чином. Спочатку перевіряється умова *condition.* Якщо умова істинна, то виконується група операторів *statements* в середині циклу. Перевірка істинності виконується на кожному кроці циклу. Якщо умова хибна, то цикл закінчує своє виконання.

Інколи необхідно закінчити цикл не по умові, що задана в його заголовку, а в результаті виконання деякої умови в тілі циклу. Для цього використовуються оператори *break* та *continue*. Оператор *break* завершує цикл while або for та передає керування програмою першому оператору після циклу. Оператор continue передає управління оператору перевірки істинності умови в циклі while та оператору оновлення значення лічильника в циклі for і продовжує виконання циклу.

Оператор `for…in` виконує задані дії для кожної властивості об'єкта чи для кожного елемента масиву і має такий вигляд:

```
for (змінна in вираз) оператор
```

Оператор `for…in` діє таким чином:

1. *Змінній* надає назву чергової властивості об'єкту чи чергового елемента масиву (залежно від природи *виразу*).
2. Виконують *оператор*.
3. Переходять до етапу 1.

При ітерації властивостей об'єкту неможливо передбачити, в якому порядку їх буде проглянуто. Але їх буде проглянути усі без виключення. Подамо приклад створення об'єкту `ob` з наступним послідовним виведенням усіх його властивостей на екран користувача.

```
<html><script>
var ob = {"а": "Літера а", "б" : 2012};
for (var key in ob) document.write(key+": "+ob[key]
    +"<BR>");</script></html>
```

На екрані побачимо такий текст:

а: Літера а

б: 2012

Оператор `with` задає назву об'єкту за замовчуванням і має такий вигляд:

`with` (*вираз*) *оператор*

Цей оператор діє таким чином. Для кожної назви в *операторі* перевіряють, чи є вона назвою властивості об'єкту, заданого згідно із

замовчуванням. Якщо відповідь ствердна, то цю назву вважають назвою відповідної властивості, інакше — назвою змінної. Це допомагає скоротити код. Наприклад, для доступу до математичних функцій маємо кожного разу вказувати назву об'єкту `Math`, як у такому коді:

```
x = Math.cos(Math.PI/2) + Math.sin(Math.LN10) +
Math.tan(2 * Math.E);
```

Того самого можна досягнути таким чином:

```
with (Math) { x=cos(PI/2) +sin(LN10)+tan(2 * E);};
```

Оператор `with` можна застосовувати лише до наявних методів.

Обробка виключних ситуацій

При виконанні сцернарію можливе винекнення помилок, які називають *виключеннями*: звертання до відсутнього об'єкта чи неможливість перетворення величини до заданого типу тощо.

Оператор `try…catch` використовують для опрацювання такого виключення. Він має такий вигляд:

`try { `*оператор*$_1$` } catch ( `*виключення*` ) { `*оператор*$_2$` }`

Тут *виключення* — довільна назва змінної, *оператор*$_1$ — містить код, що може спричинити виключення. Якщо виключення не відбулося, то після виконання *оператору*$_1$ здійснюють перехід до наступного за `try…catch` оператору. Інакше інформацію про виключення зберігають як величину локальної змінної *виключення*, а керування передають *оператор*$_2$, який має містити код опрацювання цього виключення. Якщо виключення неможливо на даному рівні опрацювати, то *оператор*$_2$ має містити оператор `throw` для переходу до опрацювання виключення на вищому рівні (див. далі).

Оператор `throw` породжує виключення, яке вже можна опрацювати оператором `try…catch`. Він має такий вигляд:

```
throw expression
```

Тут *expression* — Будь-який вираз. Результат обчислення expression буде кинутий як виняток.

Подамо приклад породження й опрацювання виключення.

```
<html><body><script>
var e;
function getMonthName(m)
  {    var    months=new    Array    ("січень","лютий",
  "березень","квітень","травень","червень","липень",
  "серпень", вересень","жовтень","листопад","грудень");
  m = m - 1;
  if (months[m] != null) return months[m]
```

```
  else throw "trow";
}
  try        { monthName = getMonthName(7)}
  catch (e) { monthName="Неправильний № місяця"};
  document.write(monthName);
  </script></body></html>
```

При заміні (7) на (77) замість слова «липень на екрані побачимо текст: «Неправильний № місяця».

5.2. Використання функцій і об'єктів в JavaScript

Функція – JavaScript це іменована група команд, які вирішують певну задачу та можуть повернути деяке значення. Функція визначається за допомогою оператору *function* , що має такий синтаксис:

```
function Ім'я_функції ([параметри])
{
[оператори]
return [значення_що повертається]
}
```

Параметри, що передаються функції, розділяються комами. Необов'язковий оператор *return* в тілі функції (блок операторів, що обмежений фігурними дужками), визначає значення, що повертається функцією. Визначення функції тільки задає її ім'я и визначає, що буде робити функція при її визові. Безпосереднє виконання функції реалізується, коли в сценарії відбувається її виклик та передаються необхідні параметри. Відзначимо, що визначення функції необхідно реалізувати на HTML-сторінці до її виклику. Наприклад, для показу на екрані вікна повідомлення з текстом "Це виклик функції" визначимо функцію Go та реалізуємо її виклик:

```
<html><head><title>Використання JavaScript</title>
    <script>
    function Go() {
    alert("Це виклик функції")
    }
    </script>
</head><body>
<script>
Go();
</script></body></html>
```

Об'єктна модель JavaScript

JavaScript відноситься до об'єктно-орієнтованих мов програмування. *Об'єкт* – це цілісна конструкція, що має *властивості*, які є змінними JavaScript та *методи* їх обробки. Властивості можуть бути іншими об'єктами. Функції, пов'язані з об'єктом, називаються *методами* об'єкта. Для звернення

до властивостей об’єкту використовується наступний синтаксис:

```
objectName.propertyName
```

Ім’я об’єкту, імена властивостей та методів чуттєві до регістру. Для визначення властивостей їм необхідно присвоїти значення. Наприклад, якщо існує об’єкт з іменем *myCar,* то для визначення властивості model необхідно:

```
myCar.model = "Таврія"
```

Для визначення методів необхідно спочатку визначити звичайну функцій, а після цього, необхідно зв’язати цю функцію з існуючим об’єктом:

```
object.methodname = function_name
```

де *object* – існуючий об’єкт, methodname – ім’я, що призначається методу, *function_name* – ім’я функції.

Виклик методу в контексті об’єкту реалізується так:

```
object.methodname (params);
```

Для створення екземпляру об’єкта необхідно:

- Написати функції, які будуть використані в якості методів об’єкту.
- За допомогою звичайної функції визначити об’єкт.
- За допомогою оператора *new* створити екземпляр об’єкту.

Наприклад необхідно створити об’єкт з іменем car та властивостями model та color та методом go. Для цього необхідно написати функцію when, яка буде використана для визначення методу go:

```
function when() {
//код функції
}
```

Після цього необхідно написати функцію для визначення об’єкта:

```
function car( model, color) {
   this.model = model;
   this.color = color;
   this.go =when;
}
```

Відзначимо, що оператор *this* використовується для того, щоб присвоїти значення властивостям об’єкту, базуючись на параметрах, що передаються функції.

Створення об’єкту з іменем *mycar* можливо реалізувати так:

```
mycar = new car("Таврія", "Зелений")
```

В JavaScript всі елементи (теги) на HTML-сторінці вистроєні в ієрархічну структуру. Причому кожен елемент представлений в вигляді об’єкту, з визначеними властивостями та методами. Керування об’єктами на HTML-сторінці можливо багато в чому за рахунок того, що JavaScript дозволяє одержати доступ до цих властивостей та методів. При реалізації доступу необхідно враховувати ієрархію об’єктів на HTML-сторінці. Відзначимо, що загальним об’єктом контейнером є об’єкт *window*, який відповідає вікну браузера. В свою чергу цей об’єкт містить деякі елементи оформлення, наприклад рядок стану. Завантаженій в вікно браузера HTML-сторінці відповідає об’єкт *document*. Всі без виключення елементи HTML-сторінки є властивостями об’єкту document. Прикладами об’єктів HTML є

таблиця, гіперпосилання або форма. Для доступу до методів/властивостей елементів на HTML-сторінці використовується наступний синтаксис:

```
document.ім'я_об'єкту.ім'я_методу()
```

Обробка подій

Важливою ознакою інтерактивних HTML-сторінок є можливість реакції на дії користувача. Наприклад, натиск на кнопці повинен викликати появу діалогового вікна, або виконання перевірки правильності введених користувачем даних. В JavaScript інтерактивність реалізована за допомогою *перехоплення* та *обробки подій*, викликаних в результаті дій користувача. Для цього в теги деяких елементів введені параметри *обробки подій*. Ім'я параметру обробки події починається з префіксу on, за яким йде назва події. Наприклад, події клік кнопкою миші Click, відповідає параметр обробки події з назвою onClick. Назви та характеристики деяких подій наведені в табл. 5.1.

Таблиця 5.1 Події JavaScript

Подія	Характеристика події	Обробник події
Click	Клік кнопкою миші на елементі форми або гіперпосилання	onClick
KeyDown	Натиск на клавіші клавіатури	onKeyDown
Load	Завантажується документ в браузер	onLoad
MouseDown	Натиск на кнопці миші	onMouseDown
MouseOver	Курсор знаходиться над елементом	onMouseOver
MouseOut	Курсор покидає зону над елементом	onMouseOut

Задача. Необхідно, щоб при наведені курсору на комірку таблиці із написом "Привіт" з'являлось вікно повідомлення з фразою "Hello". Можливі рішення:

Варіант 1:

```
<td onClick="alert('Hello')"> Привіт </td>
```

Варіант 2:

```
<script>
function Go() {
alert("Hello")
}
</script>
<td onClick="Go()"> Привіт </td>
```

В варіанті вирішення 1, код JavaScript був записаний безпосередньо в тезі, а в варіанті 2 наслідком кліку став виклик функції. Варіант 2 слід використовувати, якщо код обробки події великий за обсягом.

Стандартні об’єкти і функції JavaScript

В ядрі JavaScript визначені об’єкти та функції, які можливо використовувати не використовуючи контекст завантаженої сторінки. До основних об’єктів відносяться: Array, Date, Math, String.

Array –масив. Масив це упорядкований набір однотипних даних, до елементів якого можливо звернутись по імені або по індексу. Для створення масиву необхідно використати одну із двох конструкцій:

```
ім’я_масиву=new Array([елемент1],[елемент2],[елемент3],...)
ім’я_масиву = new Array([довжина масиву])
```

Відзначимо, що перший елемент масиву має номер 0.

В першій конструкції в якості параметрів використовуються елементи масиву, в другій конструкції використовується довжина масиву.

Наприклад:

ar1 = new Array(1, 2, 3)

ar2 = new Array(3)

Для доступу до значень елементів масиву в квадратних дужках біля імені масиву необхідно вказати порядковий номер елементу. Наприклад:

a = ar1[2]

ar1[0] = 7

В цьому прикладі, змінній *a* присвоюється значення елементу масиву за номером 2, а в елемент масиву за номером 0 записується значення 7.

Особливістю масивів JavaScript є те, що розмір масиву може встановлюватись динамічно. Наприклад, якщо для масиву із попереднього прикладу написати:

ar1[100] = 7,

то розмір масиву буде автоматично установлений рівним 101.

Для визначення довжини масиву можна скористатись властивістю *length*. Наприклад:

a = ar1. length

Зручність використання масивів забезпечується рядом методів, представлених в табл. 5.2

Таблиця 5.2 Методи об’єкту Array

Метод	Призначення

concat	Об’єднує два масиви в один.
	var alpha = ["a", "b", "c"]; var numeric = [1, 2, 3];
	// створює масив ["a", "b", "c", 1, 2, 3];
	var alphaNumeric = alpha.concat(numeric);

join	Об’єднує всі елементи масиву в один рядок.
	var arr = [1, 2 , 3];
	arr.join('+') ; // "1+2+3"
	arr.join() ; // "1,2,3"

pop	Знищує останній елемент із масиву і повертає його

	значення.
	myFish = ["angel", "clown", "mandarin", "surgeon"];
	popped = myFish.pop();

push	Додає один або декілька елементів в кінець масиву і повертає останній добавлений елемент. var array = ["one", "two"] // додати елементи "three", "four" var pushed = array.push("three", "four")
reverse	Переставляє елементи масиву в зворотному порядку: перший елемент стає останнім, а останній першим. arr = [1,2,3] a = arr.reverse()
shift	Знищує перший елемент масиву і повертає його значення. var arr = ["мій","маленький", "масив"] var my = arr.shift() // => "мій" alert(arr[0])

slice	Створює перетин масиву в вигляді нового масиву
	var arr = [1, 2, 3, 4, 5]
	arr.slice(2) // => [3, 4, 5]
	arr.slice(1, 4) // => [2, 3, 4]

splice	Додає та/або знищує елементи масиву
	arr = ["a", "b", "c", "d", "e"];
	removed = arr.splice(1,2);

sort	Сортує елементи масиву arr = [1,-1, 0]; a = arr.sort();

unsift	Додає один або більше елементів в початок масиву та повертає нову довжину масиву
	var arr = ["a", "b"];
	unshifted = arr.unshift(-2, -1);

Об’єкт *Date* використовується для роботи з датами. Синтаксис оператора створення екземпляра об’єкту дати:

ім’я_об’єкту_дати = new Date([параметри])

Якщо параметри відсутні, то значенням об’єкту буде поточна дата. Параметром може бути рядок типу: "місяць день, рік часи:хвилини;секунди".

Наприклад, для створення дати – "5 лютого 2005 року 23:12:07" необхідно:

day = new Date("February 5, 2005 23:12:07")

Прочитати або змінити параметри створеного об’єкту Date можливо за допомогою ряду методів.

Найбільш вживані методи показані в табл. 5.3

Таблиця 5.3 - Методи об’єкту Date

Метод	Призначення
getDate	Повертає число місяця для вказаної дати var sputnikLaunch = new Date("October 4, 1957 19:28:34 GMT")

getDay	Повертає день тижня для вказаної дати
getHours	Повертає годину для вказаної дати
getMinutes	Повертає хвилини для вказаної дати
getMonth	Повертає місяць для вказаної дати
getSeconds	Повертає секунди для поточного часу
getYear	Повертає рік для вказаної дати
setDate	Встановлює число місяця для вказаної дати theBigDay = new Date("July 27, 1962 23:30:00") theBigDay.setDate(24)
setDay	Встановлює день тижня для вказаної дати
setHours	Встановлює годину для вказаної дати
setMinutes	Встановлює хвилини для вказаної дати
setMonth	Встановлює місяць для вказаної дати
setSeconds	Встановлює секунди для вказаної дати
setYear	Встановлює рік для вказаної дати

Об’єкт *Math* дозволяє використовувати вбудовані в JavaScript математичні функції та константи. При зверненні до методів та властивостей цього об’єкту створювати його не потрібно, але необхідно явно вказувати його ім’я.

Наприклад для того, щоб записати в змінну a результат розрахунку функції sin від 1 радіану необхідно:

a = Math.sin(1)

Для того, щоб записати в змінну a результат виразу 5 в степені 6 необхідно:

a = Math. pow(5,6)

Методи об’єкту Math, що використовуються найбільш часто представлені в табл. 7.4.

Таблиця 7.4 Методи об’єкту Math

Метод	Призначення
abs	Повертає абсолютне значення змінної. Math.abs(-2);
sin, cos, tan	Повертають значення тригонометричних функцій. Аргументи задаються в радіанах.
acos, asin, atan	Повертають значення обернених тригонометричних функцій
exp, log	Повертають значення експоненціальної функції та функції натурального логарифму
ceil	Повертає найменше ціле число, більше або рівне значенню аргументу
floor	Повертає найменше ціле число, менше або рівне значенню аргументу
min, max	Повертає найбільше/ найменше значення з двох аргументів

pow	Повертає значення функції: pow(x,y)=x^y
round	Повертає значення аргументу, округлене до найближчого цілого числа
sqrt	Повертає квадратний корінь аргументу

Об'єкт *String* використовується для роботи з рядковими типами даних. Створення об'єкту String відбувається коли змінній присвоюється рядковий літерал:

a = "Не явний спосіб створення рядкового об'єкту"

Крім того, можливо явно створити рядковий об'єкт, використовуючи оператор new та конструктор String:

ім'я_об'єкту = new String(Рядок)

Параметром конструктору може бути будь-який рядок. Наприклад:

a=new String("Явний спосіб створення рядкового об'єкту")

Єдиною властивістю об'єкту String є *length*, що зберігає довжину рядка. Наприклад для запису в змінну h довжини рядка a необхідно:

h=a.length

Методи об'єкту String, що використовуються найбільш часто перераховані в табл. 5.5. Наведемо приклад використання методу toLowerCase для переводу рядкової змінної a в верхній регістр:

a=a.toLowerCase();

Таблиця 5.5 Методи об'єкту String

Метод	Призначення
anchor	Створює HTML якір, який використовується, як гіпертекстове посилання.
link	Створює гіпертекстове посилання, по якій можливо перейти на інший URL.
fontsize	Виводить рядок, з встановленим розміром шрифта
bold	Виводить рядок, що відображається напівжирним шрифтом
italics	Виводить рядок, що відображається курсивом
strike	Виводить рядок, що відображається перекресленим шрифтом
substring	Повертає частину рядка об'єкта string
sub	Виводить рядок, що відображається як нижній індекс
sup	Виводить рядок, що відображається як верхній індекс
toLowerCase, toUperCase	Переводить зміст рядка в нижній/верхній регістр

На додаток до стандартних об'єктів JavaScript існує декілька функцій, для виклику яких не потрібно створювати об'єктів. Ці функції дістали назву "функцій верхнього рівня". До цих функцій відносяться: *parseFloat(параметр)* та *parseInt(параметр)*. Їх призначенням є аналіз

рядкового аргументу та повернення відповідно число з плаваючою крапкою або цілого числа. Також досить часто використовуються функції *Number(об'єкт)* та *String(об'єкт)*, які перетворюють об'єкт, що використовується в якості параметру в число або рядок.

Використання об'єктів window, document, location.

Об'єкт *window* створюється автоматично при запуску браузера. Крім того нове вікно можливо створити і засобами JavaScript. Для цього необхідно використати метод open. Синтаксис методу такий:

Ім'я_змінної = window.open([Ім'я_ файлу],[Ім'я_вікна],[Параметри])

Ім'я_змінної – це ім'я для звернення до нового вікна в програмі JavaScript.

Ім'я_ файлу – це повна або відносна URL-адреса документу, що буде відкриватись в вікні браузеру.

Ім'я_вікна – це ім'я, що буде вказане в якості цілі в гіпертекстовому посиланні на це вікно із іншого HTML-документу.

Параметри – задають значення параметрів вікна. Основні параметри представлені в табл. 5.6. Якщо можливе значення властивості yes або no, то при стандартних настройках використовується значення yes.

Таблиця 5.6 - Основні параметри вікна

Назва	Призначення	Можливі значення
directories	Наявність/відсутність панелі "Ссылки"	yes/no
height	Висота вікна	кількість пікселів
location	Наявність/відсутність адресного рядка	yes/no
menubar	Наявність/відсутність рядка меню	yes/no
resizable	Можливість/не можливість зміни розмірів віна користувачем	yes/no
scrollbars	Наявність/відсутність смуг прокрутки віна	yes/no
status	Наявність/відсутність рядка стану браузера	yes/no
toolbar	Наявність/відсутність панелей інструментів	yes/no
width	Ширина вікна	кількість пікселів

Наприклад, для створення нового вікна браузера в якому буде завантажено файл a.html необхідно:

```
<script>
myw=window.open("a.html","displayWindow",
    "width=400,height=300,status=no,toolbar=no,
    menubar=no")   </script>
```

При цьому, ширина вікна дорівнює 400 пікселів, висота вікна 300 пікселів, рядок стану вікна, панель інструментів та рядок меню будуть відсутні. Відзначимо, що наведений код необхідно записати в одному рядку.

Для закриття вікна браузера використовується метод close. Наприклад для закриття вікна попереднього прикладу необхідно:

```
myw.close()
```

Відзначимо, що при звернення до методів та властивостей вікна браузера в якому знаходиться програма JavaScript імені вікна або ключового слова window можна не вказувати. Наприклад, закрити поточне вікна браузера можливо так:

```
close()
```

Цікавим методом об'єкту window є метод *setTimeout*, за допомогою якого можливо запрограмувати виконання деяких команд після закінчення встановленого терміну часу. Синтаксис методу такий:

```
setTimeout("Код_JavaScript",інтервал_часу)
```

В якості першого параметру функції setTimeout, як правило використовують функцію. Відзначимо, що цю функцію необхідно визначити на HTML-сторінці до використання функції setTimeout. Другим параметром функції setTimeout є інтервал часу закінчення якого буде сигналом про початок виконання команд JavaScript. Цей параметр задається в мілісекундах. Наприклад, виконання функції myfunction через 3000 мілісекунд після завантаження HTML-сторінки можливо запрограмувати так:

```
setTimeout("myfunction()",3000)
```

Досить часто функція setTimeout використовується для створення анімаційних ефектів. Це може бути, наприклад, циклічна зміна кольору тексту, або циклічна заміна одного зображення іншим.

Об'єкт *document* містить інформацію про завантажену сторінку. Всі елементи HTML-сторінки є властивостями цього об'єкту. Найбільш використовуваним методом об'єкту document є метод *write*, за допомогою якого можливо зробити запис в HTML-сторінку. Наприклад, для запису рядка "Привіт JavaScript" в документ в якому знаходиться сценарій необхідно:

```
document.write("Привіт JavaScript")
```

Ще одним широко вживаним методом цього об'єкту є *getElementById*, що реалізує доступу до будь-якого об'єкту з визначеним id. Синтаксис методу такий:

```
document.getElementById(ім'я_id)
```

Наприклад, встановлення значення стилю display елемента з id=myid рівним block, можливо реалізувати так:

```
document.getElementById("myid").style.display="block"
```

Властивості об'єкту *location* дозволяють одержати інформацію про URL-адресу завантаженої HTML-сторінки. Метод *reload* дозволяє пере завантажити в браузер поточну HTML-сторінку. Метод *replace* завантажує в вікно браузера сторінку, адреса якої використана в якості параметру цього методу.

Контрольні питання до теми 5

1. Навіщо використовується мова JavaScript?
2. Як визначити в HTML- документі код JavaScript?
3. Синтаксис запису змінних в JavaScript.
4. Синтаксис запису виразів в JavaScript.
5. Типи даних JavaScript.
6. Оператори JavaScript.
7. Оператори вибору JavaScript.
8. Оператори циклів JavaScript.
9. Як достроково закінчити цикл?
10. Як достроково перейти на наступну ітерацію циклу?
11. Правила запису функцій JavaScript?
12. Що таке об’єкт в мові JavaScript?
13. Як визначити об’єкт користувача в JavaScript?
14. Стандартні об’єкти JavaScript.
15. Як реалізована інтерактивність в JavaScript?
16. Стандартні функції JavaScript.
17. Методи та властивості об’єкту Array.
18. Методи та властивості об’єкту Date.
19. Методи та властивості об’єкту Math.
20. Методи та властивості об’єкту String.
21. Методи та властивості об’єкту Window?
22. Як за допомогою JavaScript відкрити нове вікно браузеру ?
23. Як за допомогою JavaScript запрограмувати виконання деяких команд після закінчення встановленого терміну часу?
24. Методи та властивості об’єкту Document.
25. Методи та властивості об’єкту Location.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Юринець В.Є. Комп'ютерні мережі. Інтернет: навч. посіб. Львів: ВЦ ЛНУ, 2006.

2. Возняк, Л.С. Комп'ютерний практикум. Формування навичок роботи із сервісами мережі Інтернет. І.-Ф.: ВДВ ЦІТ, 2006.

3. Глинський, Я. М. Інтернет. Сервіси, HTML і web-дизайн: Навч. посіб.- 3-є вид. Львів : Деол, СПД Глинський, 2005.

4. Татенбаум Э. Компьютерные сети. СПб. : Питер, 2008.

5. Глушаков, С.В. Работа в сети Internet. Фолио, 2002.

6. Иванов В. Интернет для начинающих. Самоучитель. К.: Питер; BHV, 2005.

7. Самсонов В. В. Методи та засоби Інтернет-технологій. Харків: Компанія СМІТ, 2008.

8. Юринець В.Є. Комп'ютерний практикум. Формування навичок роботи із сервісами мережі Інтернет. Львів: ВЦ ЛНУ, 2006

9. Гаевский А.Ю. Самоучитель по созданию Web-страниц HTML JavaScript Dynamic HTML.К.: А.С.К., 2002.

10. Матвієнко О.В. Internet -технології: проектування Web-сторінки: Навч.посібник. К. : ЦУЛ, 2004.

ІНФОРМАЦІЙНІ РЕСУРСИ

1. Підручники HTML і CSS. [Електронний ресурс]. – Режим доступу: http://htmlbook.at.ua/news/tutorial_html/1-0-1

2. Для тех., кто делает сайты. [Електронний ресурс]. – Режим доступу: http://htmlbook.ru/

3.Введение в HTML. [Електронний ресурс]. – Режим доступу: http://www.intuit.ru/studies/courses/33/33/info

4. Применение каскадных таблиц стилей (CSS). [Електронний ресурс]. – Режим доступу: http://www.intuit.ru/studies/courses/34/34/info

5. Практикум по программированию на JavaScript. [Електронний ресурс]. – Режим доступу: http://www.intuit.ru/studies/courses/119/119/info

6. Основы программирования на JavaScript. [Електронний ресурс]. – Режим доступу: http://www.intuit.ru/studies/courses/1093/132/info

7. Введение в HTML и CSS:. [Електронний ресурс]. – Режим доступу: http://www.intuit.ru/studies/courses/1005/276/info

7. Введение в JavaScript. [Електронний ресурс]. – Режим доступу: http://www.intuit.ru/studies/courses/35/35/info

8. Практикум по программированию на JavaScript. [Електронний ресурс]. – Режим доступу: http://www.intuit.ru/studies/courses/119/119/info