Міністерство освіти і науки України
ВСП «Ковельський промислово-економічний фаховий коледж
Луцького національного технічного університету»

«Web-технології та Web-дизайн»

Конспект лекцій для здобувачів освітньо-кваліфікаційного рівня фаховий молодший бакалавр IV курсу спеціальності 122 Комп'ютерні науки денної форми навчання

Ковель 2021

УДК 004.41
В-26

Затверджено до видання методичною радою ВСП «КПЕФК Луцького НТУ»,

протокол № __ від « ___ » _____ 2021 року.

Голова методичної ради КПЕФК Луцького НТУ_____І. М. Ілюшик

Розглянуто і схвалено на засіданні випускної методичної комісії зі спеціальності «Обслуговування програмних систем і комплексів» ВСП «Ковельський промислово-економічний фаховий коледж Луцького НТУ», протокол

№ ___ від «____» _____ 2021 року.

Голова випускної методичної комісії_____О.В. Присада

Електронна копія друкованого видання передана до книжкового фонду бібліотеки ВСП «Ковельський промислово-економічний фаховий коледжу Луцького НТУ»

Завідувачка бібліотеки_____Н.М.Телючик
(підпис)

Укладач: _____ Л.В.Мелещук, викладач ВСП «КПЕФК Луцького НТУ»
(підпис)

Рецензент: _____ Т.В. Селівончик, канд. техн. наук, доцент, директор ВСП «КПЕФК
(підпис) Луцького НТУ»

Відповідальний за випуск: _____ Л.В. Прокопчук, методист ВСП «КПЕФК
(підпис) Луцького НТУ»

Web-технології та Web-дизайн [Текст]: Конспект лекцій для здобувачів освітньо-кваліфікаційного рівня фаховий молодший бакалавр IV курсу спеціальності 122 Комп'ютерні науки денної форми навчання / уклад. Л.В. Мелещук. – Ковель : ВСП «КФПЕК Луцький НТУ», 2021. – 93.

В-26

У конспекті лекцій розглянуті питання, пов'язані з графічним дизайном Web-ресурсів, основні рекомендації по HTML верстці сайтів. Розглядається застосування каскадних таблиць стилів CSS для візуального форматування, розробка інтерактивних сценаріїв на мові JavaScript та використання технології AJAX.

Конспект лекцій призначений для здобувачів освітньо-кваліфікаційного рівня фаховий молодший бакалавр денної форми навчання

ЗМІСТ

ВСТУП

Ефективне використання інформації, формування оптимальних рішень у різноманітніх сферах в значній мірі залежить від рівня впровадження та використання інформаційних систем і технологій. На сучасному етапі розробка та впровадження досконалих інформаційних систем неможлива без їх адаптації до використання в глобальній мережі Interhet, а також однієї з її основних частин мережі WWW.

Конспект лекцій являє собою короткий письмовий виклад матеріалів з дисципліни «Web-технології та Web-дизайн».Основна увага спрямована на принципи та прийоми засвоєння основ веб, HTML, веб-порталів, технології AJAX та веб-дизайну. Конспект лекцій використовується студентами спеціальності 122 Комп'ютерні науки, при підготовці до здачі екзамену і ККР, при узагальненні та систематизації отриманої інформації , при закріпленні теоретичного матеріалу, а також при вивченні матеріалів інших дисциплін, пов'язаних з Інтернет програмуванням.

Даний курс базується на вивченні наступних тематичних напрямків:

- Основи Web - дизайну та графіка для Web.
- Теоретичні основи HTML , включаючи особливості HTML 5.0.
- Теоретичні основи CSS, включаючи особливості CSS 4.0.
- Базовий синтаксис JavaScript і бібліотеки JQuery.
- Технологія AJAX.

Метою дисципліни є формування та узагальнення спеціальних знань та навичок студентів з питань створення програм для глобальної мережі Інтернет. Програма розрахована на студентів, які засвоїли дисципліни «Основи програмування та алгоритмічні мови», «Алгоритми та структури даних», «Бази даних», «Комп'ютерні мережі».

У результаті вивчення дисципліни студент повинен знати:

- основні принципи створення Web-сторінок;
- засоби макетування та верстки засобами HTML5.0;
- засоби макетування та верстки засобами CSS;
- основи роботи серверних програм;
- схему роботи мереж клієнт/сервер.

У результаті вивчення дисципліни студент повинен вміти:

- створювати окремі програми для динамічних сайтів на базі HTML5.0, стилів CSS та PHP;
- працювати з розміткою HTML5.0;
- працювати зі стилями CSS;
- обробляти строкові данні;
- створювати програми, що можуть управляти зовнішнім виглядом сайту.

Самостійне вивчення матеріалу з дисципліни, передбачене програмою, є запорукою того, що студент зможе самостійно використовувати знання для розробки і розгортання складних WEB-застосувань.

У процесі викладання дисципліни необхідно виховувати зацікавленість студентів у застосуванні комп'ютерної техніки в обраній спеціальності,

організовувати інформаційні дані різних видів, широко налагоджувати міждисциплінні зв’язки, особливо зі спецдисциплінами.

Тема. Принцип доступу до інформації в мережі інтернет

1. Основи Інтернету.
2. Адресація в Інтернеті.

1. Основи Інтернету

Сьогодні у світі є сотні тисяч великих та малих комп'ютерних мереж. Багато з них з'єднані між собою й утворюють єдиний інформаційний простір, який складається з мільйонів комп'ютерів. Цей єдиний віртуальний простір називають *Інтернетом.*

За кількістю комп'ютерів та охоплюваною територією Інтернет є найбільшою у світі мережею. За типом вона належить до клієнт-серверних мереж, тобто в Інтернеті є комп'ютери-сервери, які зберігають інформацію та надають її комп'ютерам-клієнтам.

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

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

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

Найпопулярнішою зі служб є Всесвітня павутина (www), або просто Веб. Це розповсюджена по всьому світу інформаційна мультимедійна система, яка об'єднує в єдиному просторі інформацію різних типів. Робота у Веб подібна до віртуальної подорожі світом з відвідуванням цікавих місць. Ця служба базується на протоколі HTTP (протокол передавання гіпертексту).

Іншою службою, з якою вам доведеться працювати під час вивчення матеріалу, є FTP (протокол передавання файлів). Як ви, напевно, здогадалися, назва цієї служби збігається з назвою протоколу, який вона використовує. Сервери, що підтримують цей протокол, називають FTP-серверами. Частина дискового простору таких серверів доступна через Інтернет.

Крім того, до служб Інтернету належать електронна пошта, служби миттєвого передавання повідомлень (наприклад, ICQ), служба новин та інші.

2. Адресація в Інтернеті

Усі комп’ютери, підключені до Всесвітньої мережі, працюють в автоматичному режимі, без участі людей. Для того щоб такий комп’ютер мав

змогу передавати та приймати дані з використанням протоколу IP, він повинен мати унікальну адресу, яку називають IP-адресою. Вона має такий формат: xxx.xxx.xxx.xxx, де xxx число від 0 до 255 (наприклад, 193.205.31.47). Призначенням IP-адрес займаються спеціальні організації.

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

У багатьох країнах домен першого рівня є кодом країни: *ua* — Україна, ru — Росія, fr — Франція, ca — Канада, jp - Японія, *au* — Австрія

Домени першого рівня можуть також позначати сферу діяльності: com — комерційні компанії, gov — урядові організації, edu — навчальні заклади, *org* — некомерційні організації, mil — військові організації.

Отже, система доменних імен організована у зручний для людини спосіб, оскільки вона має постійну структуру і дає змогу визначити, якій організації належить сервер, та в якій країні ця організація розміщена.

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

Питання для самоконтролю

1. Дайте визначення Інтернету.
2. Опишіть основи роботи інтернету.
3. Дайте визначення IP-адресі.
4. Що таке доменні імена.
5. Охарактеризуйте доменні імена.

Тема. Вступ до Web-технологій

План

1. Основні поняття Web-технологій.
2. Класифікація Web-технологій.
3. Мови розмітки: HTML, XML, XHTML, WML.

1. Основні поняття Web-технологій

Як зазначалося, найвідомішою та найпопулярнішою службою Інтернету є Всесвітня павутина (Веб). Саме після розповсюдження став можливий масовий доступ користувачів до Всесвітньої мережі. Своєю появою Веб має завдячити

Тіму Бернесу-Лі, який винайшов протокол HTTP, адреси URL та мову HTML— технології, на яких ґрунтується Веб.

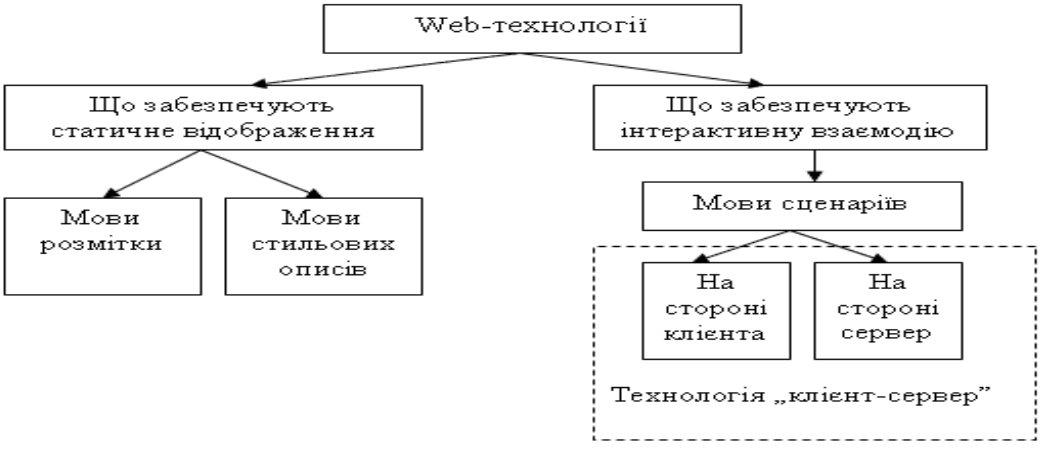
Служба Веб підтримується сукупністю серверів, які здатні обмінюватися даними за протоколом HTTP. Цих серверів мільйони, й розповсюджені вони по всьому світу. На них містяться *веб-сторінки* — спеціальні документи, створені з використанням мови HTML. Кожна веб-сторінка має адресу, за допомогою якої вона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

Перегляд веб-сторінок здійснюється у спеціальних програмах-браузерах, найпоширенішими з яких є Internet Explorer, Mozilla та Opera. Для відтворення веб-документа у вікні браузера достатньо ввести його URL в поле Адреса і натиснути клавішу enter. Основною особливістю та перевагою веб-сторінок є те, що інформація на них організована як *гіпертекст.* Це текст, в який вбудовано спеціальні коди, що керують такими додатковими елементами, як форматування, ілюстрації, мультимедійні вставки та гіпертекстові посилання.

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

Процес переходу в інші місця поточної веб-сторінки або до інших веб-сторінок за допомогою гіперпосилань називають *навігацією.* Якщо після низки переходів за гіперпосиланнями необхідно повернутися на попередню сторінку, то користуються кнопкою Назад інструментів браузера. Кількість веб-сторінок, що тематично пов'язані між собою й розкриває єдине ціле, називають *веб-сайтом* або просто *сайтом.*

2. Класифікація Web-технологій



3. Мови розмітки: HTML, XML, XHTML, WML

Вивчення основних Інтернет-технологій Web-дизайну ми починаємо з огляду мов розмітки.

Виникла необхідність створення спеціальної мови, яка би забезпечувала розміщення текстової і графічної інформації на Web-сторінці і забезпечувала би адекватне відображення даної інформації на будь-якому комп'ютері. Таку спеціалізовану мову назвали *мовою розмітки*, головними функціями якої стали – «розміщення» візуальної інформації на Web-сторінці й організація зв'язків (гіперпосилань) між останніми.

HTML (HyperText Markup Language)

Спочатку система World Wide Web призначалася для відображення текстової інформації, і HTML розроблялася як мова форматування тексту. В даний час HTML є самою популярною мовою в WWW. Документ, написаний мовою HTML, являє собою текстовий файл, що містить дані й інструкції з їхнього форматування. Засоби, призначені для зв'язування документів і форматування даних, називаються ***тегами*** (tag).

Текст Web-сторінки і теги розміщаються в одному файлі, що називається HTML-документом. Від того, які теги ви виберете і як їх застосуєте, залежить зовнішній вигляд Web-сторінки у вікні браузера. Для форматування і розміщення даних у мові HTML передбачені сотні тегов. Наприклад, теги і формують абзац, а пара тегов і вказує на те, що текст, що міститься між ними, повинен відображатися курсивом. Теги також використовуються для створення гіпертекстових посилань. Завдяки наявності гіпертекстових посилань користувач може викликати інший документ щигликом миші на фрагменті тексту або зображенні. Мова HTML постійно удосконалюється. Застарілі елементи видаляються, а нові включаються до складу мови. Незважаючи на це, структура HTML залишається незмінною.

XML (eXtensible Markup Language)

XML схожий на мову HTML тим, що для опису різних розділів документа в ньому використовуються теги. На відміну від HTML, XML дозволяє розроблювачам задавати власні теги і ставити у відповідність їм способи відтворення інформації. XML-теги чуттєві до регістра символів.

XHTML

XHTML являє собою сполучення HTML і XML. XHTML 1.0 у даний час є стандартною мовою розмітки і рекомендований W3C для використання замість HTML, але на сьогоднішній день більшість розроблювачів продовжують застосовувати HTML. Мова XHTML гарантує, що зовнішній вигляд документа не буде змінюватися в залежності від платформи (Windows, Mac або Unix). Незважаючи на розходження в структурі, у XHTML використовуються дескриптори HTML, що розпізнаються сучасними браузерами, тобто, наявні на даний час браузери по суті підтримують XHTML 1.0. В даний час розробляються нові мови розмітки, такі як, WML (Wireless Markup Language) застосовується для створення інформаційних ресурсів, призначених для власників стільникових телефонів; CDF (Channel Definition Format), використовуваний для створення push-каналів у броузерах виробництва Microsoft; мова SMIL (Synchronized

Multimedia Interchange Language) служить для створення презентацій і підтримується програмою RealPlayer.

Питання для самоконтролю

1. Дайте визначення Web-сторінки.
2. Дайте визначення Web-сайту.
3. Дайте визначення Web-серверу.
4. Опишіть класифікацію Web-технологій.
5. Охарактеризуйте мови розмітки: HTML, XML, XHTML, WML.

Тема. Мови сценаріїв. Технологія «клієнт-сервер»

План

1. Мови сценаріїв. Технологія «клієнт-сервер».
2. Різниця між front-end і back-end.
3. Основи front-end розробки.

1. Мови сценаріїв. Технологія «клієнт-сервер»

На сучасному етапі розвитку Web-сторінки стають усе більш і більш інтерактивними. Web-сайти поступово стають схожими на інтерфейс додатків. Усе це досягається за допомогою сучасних технологій Web-програмування. Розуміння основних ідей програмування для Web-сторінок не є чимось надмірно складним, але серед безлічі технологій цілком можна потонути, особливо якщо врахувати, що усі вони досить сильно відрізняються одна від одної. Насправді, можна розділити технології Web-програмування на дві основні групи: програмування на *стороні клієнта* (client-side) і на *стороні сервера* (server-side). Для розуміння цих технологій необхідно знати сутність технології «клієнт-сервер». Програмний код, що забезпечує інтерактивність Web-сторінок, називається *сценарієм*. Даний термін був обраний, очевидно виходячи з їхньої специфіки таких програм. Основне їхнє призначення, цей опис «реакції» Web-сторінки на дії користувача. Таким чином, розділяють сценарії, що виконуються на стороні клієнта і виконуються на стороні сервера. Сценарії на стороні клієнта виконуються під керуванням браузера. Сценарії на стороні сервера виконуються під керуванням Web-сервера.

Перевага сценаріїв на стороні клієнта полягає в тому, що вони можуть перевіряти коректність інформації, введеної користувачами, і обробляти її, не звертаючись до сервера. Найчастіше сценарії, призначені для виконання на стороні клієнта, створюються на мовах JavaScript і VBScript.

JavaScript

JavaScript – це мова сценаріїв, розроблена Netscape і Sun Microsystems для підтримки додаткових функціональних можливостей статичних Web-сторінок. За допомогою JavaScript звичайно реалізуються такі ефекти, як, поява вікон з повідомленнями, відображення анімації. Крім того, JavaScript-сценарії часто

використовуються для визначення типу броузера і платформи, на якій він виконується. JavaScript-сценарії також успішно застосовуються для перевірки коректності даних, введених користувачем.

VBScript

Мова VBScript була розроблена корпорацією Microsoft як складова мови Visual Basic. VBScript створювався для роботи з Internet Explorer і Microsoft Internet Information Server (IIS). VBScript має багато спільного з мовою JavaScript, однак працює лише з Microsoft Internet Explorer, що обмежує сферу її застосування. VBScript є інтерпритованою мовою і може бути використана у сполученні з Web-технологіями Microsoft, такими як ASP (Active Server Page). Незважаючи на те, що VBScript призначено для написання сценаріїв, що виконуються на стороні клієнта, засоби ASP працюють на стороні сервера.

Сценарії, призначені для виконання на стороні сервера, зазвичай розташовуються в спеціальному каталозі усередині папки сайта. Приймаючи запит користувача, у якому зазначена програма, сервер запускає цю програму на виконання. У результаті виконання програми вихідні дані передаються web-серверові, а потім клієнтові. Для написання сценаріїв, що працюють на стороні сервера, звичайно використовуються такі технології, як Perl, ASP, ColdFusion, PHP, JSP і SSI.

Perl

Мова Perl користується великою популярністю серед розроблювачів Web-додатків. Могутні засоби пошуку і редагування тексту, зручність при роботі з файлами привели до того, що Perl став одним з основних мов, застосовуваних у Internet. Perl – інтерпритована мова, тому для того, щоб сценарій на стороні сервера працював нормально, на комп'ютері повинен бути встановлений Perl-інтерпретатор. Необхідність інтерпретації призводить до зниження ефективності Perl-коду в порівнянні з компілюючимися програмами. Перевагою використання Perl є той факт, що на сьогоднішній день розроблені Perl-інтерпретатори практично для будь-якої платформи і усі вони поширюються безкоштовно. Багато Web-серверів працюють у середовищі UNIX, де інтерпретатор Perl є частиною операційної системи.

PHP

PHP – це мова сценаріїв, оброблюваних сервером. Коди PHP безпосередньо включаються до складу HTML-документа. Назва PHP – це абревіатура першої версії програми, що повністю називалася Personal Home Page Tools. У PHP реалізовані кращі розв’язки багатьох мов, таких як C і Perl, крім того, PHP надає розроблювачеві могутні засоби для роботи з базами даних. Подібно Perl, PHP – вільно розповсюджувана відкрита система, і співтовариство розроблювачів має можливість модернізувати її.

2. Різниця між front-end і back-end

При створенні сайтів часто можна почути слова «фронтенд» і «бекенд». Вони втілюють у собі протилежну «філософію», але при цьому фронтенд і бекенд один без одного не змогли б повноцінно існувати. Що ж таке фронтенд і бекенд та чим вони відрізняються? Який би приклад підійшов у цьому випадку? Коли ви

ведете автомобіль, ви використовуєте кермо, щоб привести його в рух і дати інші важливі команди. Але те, що дійсно приводить авто в рух, заховане всередині. Це його двигун. Інший приклад – годинник. Циферблат, стрілки, цифри, механізм управління стрілками – все те, що бачить користувач – це є фронтенд. Ми бачимо конкретний результат – котра година, ми можемо цей час підкорегувати. Але от як працюють шестерні, як працює сам механізм, процесор чи пружинка користувач не бачить. Це і є бекенд.

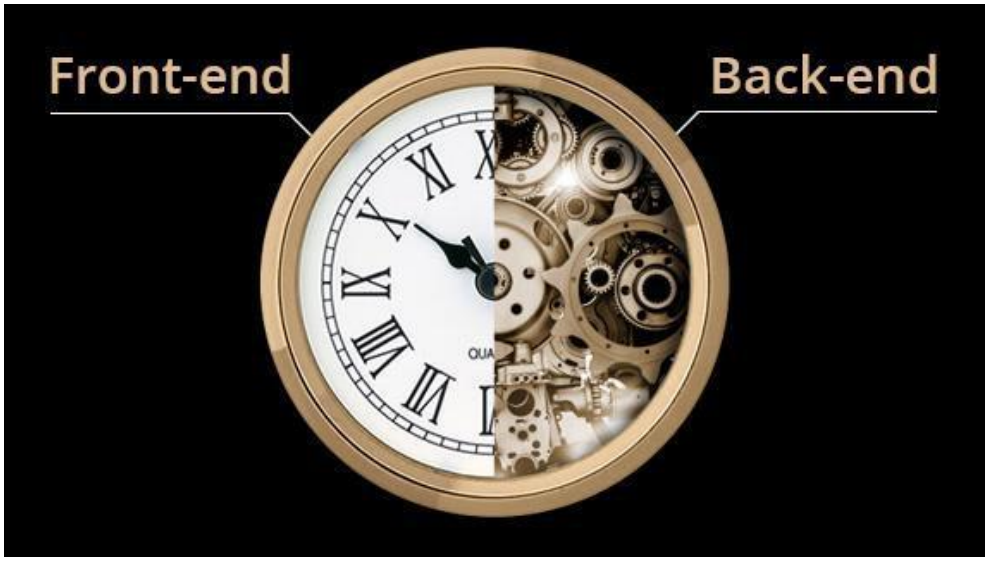


Рис.1 Різниця між фронтендом та бекендом

Фронтенд-розробка зосереджена на тих елементах сайту, які ви бачите у браузері і якими безпосередньо взаємодієте. Бекенд-розробка відповідає за функціонал сайту і має справу з речами, яких ви не бачите, — такими як бази даних та сервери.

В програмній інженерії терміни «front end» та «back end» розрізняють за принципом розділення відповідальності між рівнем представлення та рівнем доступу до даних відповідно.

Front end — це інтерфейс для взаємодії між користувачем і back end. Front end та back end можуть бути розподілені між однією або кількома системами.

В програмній архітектурі може бути багато рівнів між апаратним забезпеченням та кінцевим користувачем. Кожен з цих рівнів може мати як front end, так і back end. Front — це абстракція, спрощення базового компоненту через надання користувачу зручного (user-friendly) інтерфейсу.

Розділення програмних систем на front end та back end спрощує розробку і розділяє підтримку. Емпіричне правило полягає в тому, що front (чи «клієнтська») сторона — це будь-який компонент, яким керує користувач. Код серверної сторони (чи «back end») знаходиться на сервері.

Фронтенд-розробка: завжди «на передньому плані». Front-end - це та частина програмного забезпечення з якою безпосередньо контактує користувач.

Щоб представити контент сайту найкращим чином та забезпечити ідеальну взаємодію користувача з сайтом, фронтенд-розробники використовують багато корисних інструментів, найважливіші з яких — HTML, CSS та JavaScript. HTML (Hypertext Markup Language) відповідає за представлення коду користувачам у зрозумілій та звичній формі. CSS (Cascading Style Sheets) пропонує чудові можливості із вдосконалення стилів (кольори, фони, блики і т.д.) Javascript робить взаємодію користувача з сайтом швидшою та цікавішою за допомогою слайдів, випадних меню, безкінечного скролінгу та багатьох інших способів — вибір безмежний. Низка інших елементів та фреймворків (SAAS, LESS, Bootstrap, jQuery, Angular, Ember і т.д.) використовуютья разом із цією «великою трійкою» для значного полегшення роботи.

Бекенд-розробка: «серце» вашого сайту. Бекенд — «невидимий двигун» сайту.

Бекенд-розробники пишуть код, використовуючи такі популярні мови програмування, як PHP, Ruby on Rails, Python, .NET та інші. Коли потрібно виконати якусь операцію, бекенд-код взаємодіє з базою даних (використовуючи MySQL, SQL, Microsoft Access і т.д.). Після цього необхідна інформація повертається користувачу в формі фронтенд-коду. Займаєтесь онлайн-шопінгом? Підписуєтесь на розсилку? Редагуєте контент на сторінці? Яку б операцію ви не виконували, врешті-решт за неї відповідає бекенд-код. Інформація оновлюється, змінюється або видаляється в базі даних.

В бекенді в основному використовується математика та логіка, в фронтенді більше потрібне відчуття форми. Оскільки фронтед – це зовнішній вигляд, а бекенд – це власне вся механіка сайту. Отже, як фронтенд, так і бекенд необхідні для вашого сайту.

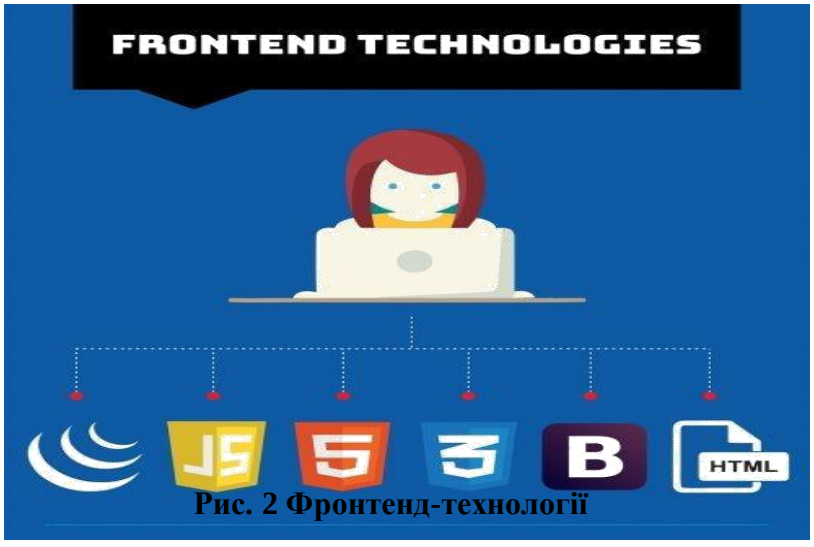


Рис. 2 Фронтенд-технології

3. Основи front-end розробки

Спеціальність Front-end розробник (веб-дизайнер) – одна з найактуальніших і високооплачуваних, попит на Front-end розробників продовжує зростати. Це пов’язано з надзвичайною швидкістю та мобільністю ведення сучасного бізнесу з використанням новітніх інформаційних технологій.

Front-end розробник – це фахівець в галузі веб-розробки та дизайну, в завдання якого входить проектування користувацьких інтерфейсів для сайтів або додатків. Робота фахівця з веб-дизайну включає в себе як оригінальні дизайнерські, так технічні рішення в області проектування веб-інтерфейсів, що забезпечують зручність користування веб-ресурсом. Професійний Front-end developer повинен постійно ставити себе на місце віддаленого користувача і створювати сторінки з інтуїтивно-зрозумілим інтерфейсом. Веб-дизайнер повинен бути знайомий як з останніми веб-технологіями, так і володіти навичками в галузі дизайну. Спеціалісти даної області повинні досконало знати HTML, CSS, JavaScript, JQuery, розуміти термін «кросбраузерність», «юзабіліті» та завжди застосовувати їх на практиці. Створення сайтів – справа захоплююча, прибуткова і цікава. Ще одна приємна річ – можна працювати фрілансером. Тому цю спеціальність дуже часто обирають студенти, щоб отримувати перший практичний досвід і забезпечити себе матеріально.

Головне завдання Front-end розробника – «одягнути» Internet-проект так, щоб якомога більше користувачів ним зацікавилися. За великим рахунком web-дизайнер повинен знати те ж, що і просто дизайнер, але його специфічне завдання полягає в тому, щоб вміти розробити стильове оформлення проекту з урахуванням специфіки Інтернету. Тобто, крім того, щоб оформлення проекту було просто стильним, воно має відповідати стандартним вимогам, що пред'являються Мережею: графічні елементи (логотипи, банери, малюнки тощо) повинні бути оптимізовані, при виборі кольору та шрифту необхідно враховувати той факт, що користувач побачить графічне втілення проекту на моніторі, а не на папері. Крім того, web-дизайнер повинен знати та використовувати загальноприйняті в Інтернеті символи, наприклад зображення збільшувального скла позначає сервіс пошуку по сайту.

Що входить до посадових обов'язків сучасного Front-end розробника? Оформлення сайтів; створення ідеї і розробки макету сервера; створення стилю виконання макета сервера; забезпечення найкращого сприйняття Web - документів на екрані монітора з урахуванням часу завантаження документів, пропускної здатності каналу передачі даних, розміру графічних файлів документа, якості колірної палітри;визначення правил компонування web - сторінок, вибір формату, фону, кількості та якості елементів оформлення;створення стильових зразків web - документів; робота з вузлом Internet; написання програмної частини і коду сторінки; управління гіпертекстовими документами; проведення інформаційної політики фірми в World Wide Web (WWW); установка і робота із засобами підготовки та перевірки web - сторінок; створення інтерактивних web -

додатків; оновлення, модернізація web - документа; створення і робота з додатками для статистичної обробки.

Професійні навички: загальні художні знання - малюнок, графіка, композиція; володіння графічними програмами (в основному продукти Adobe); навички в області проектування користувацьких інтерфейсів (юзабіліті); розуміння принципів навігаційного функціоналу інтернет-сайтів; розуміння принципів створення веб-сторінок і сайтів в цілому: HTML, CSS, JavaScript, jQuery, основиSEO та контент-менеджменту.

Фронт-енд – це робота з тим, що бачить звичайна людина на екрані свого ноутбука чи смартфона, коли заходить на сайт. Усе, починаючи від шаблону, та закінчуючи розташуванням тексту – завдання front-end розробника. Він повинен мати великий досвід

У HTML та CSS, гарні знання написання скриптів та володіння Javascript. Основною задачею фронтендника є створення платформи для спілкування з користувачами сайту, передачі та отримання інформації. *Бекенд-розробка* відповідає за функціонал сайту і має справу з речами, яких ви не бачите, — такими як бази даних та сервери. *Отже, як фронтенд, так і бекенд важливі важелі для розробки сайту.*

Питання для самоконтролю

1. Охарактеризуйте технологію «клієнт-сервер».
2. Які ви знаєте мови сценаріїв?
3. Що таке Фронт-енд розробка?
4. Що таке Бекенд-розробка?
5. Опишіть основи front-end розробки.

Тема. Основні елементи структури web-сторінки. Основи HTML

План

1. Структура веб-сторінки та її об’єкти.
2. Контейнер тіла.
3. Групи тегів HTML.

1. Структура веб-сторінки та її об’єкти

Web-сторінки мають вигляд звичайних текстових документів, в які введено вказівки, форматування. Принцип роботи браузера полягає в інтерпретації цих вказівок. При відображені таких документів браузером самі вказівки не відображаються, проте впливають на спосіб відображення решти частини документу. Згадані вказівки називають *дескрипторами* або *тегами.* З їх допомогою текст можна робити кольоровим, використовувати шрифти різного

розміру, вбудовувати мультиплікацію, відео фрагменти тощо. Формат дескрипторів задається в описі спеціальної *мови розмітки.* Вона називається *мовою розмітки гіпертексту-HTML {HyperText Markup Language}.*

Ускладнення HTML і поява мов програмування призвели до того, що розробка Web-вузлів стала справою високопрофесійною, потребуючою спеціалізації по напрямках діяльності і постійного вивчення нових Web-технологій. Але можливості Internet дозволяють користувачам, що володіють основами HTML, створювати і розміщати власні Web-вузли без великих зусиль.

Документи, розмічені за допомогою цієї мови, називають *HTML-документами.* HTML-документи мають розширення .htm або .html.

Інколи процес розробки Web-документів засобами мови HTML називають Web-програмуванням. Проте слід розуміти, що HTML не є мовою програмування у звичайному розумінні, а є мовою розмітки (опису). Термін Web-програмування мовою HTML має історичне походження. Під терміном Web-програмування будемо розуміти процес розробки Web-документів (і не лише засобами HTML).

Мова HTML розроблена спеціально для Web. Її популярність забезпечують зокрема такі властивості:

- документ, створений за допомогою деякої програми, наприклад текстового редактора, часто важко (а іноді і неможливо) використовувати в іншій програмі; HTML у цьому відношенні є універсальною;
- HTML — це відкритий стандарт;
- HTML не є власністю якої-небудь фірми;
- можливість використання гіпертексту;
- HTML підтримує мультимедіа.

2. Контейнер тіла

У мові HTML використовуються поняття *контейнера.* Контейнер — це дескрипторна пара, яка складається з початкового і кінцевого *дескрипторів (тегів).* Початковий дескриптор має вигляд <TAG>, де TAG ім’я певного HTML-дескриптора. Кінцевий дескриптор має вигляд </TAG>.

Наприклад

<B> Програми для перегляду Web-сторінок називаються броузерами </B>

▪ *Контейнер <B></B> дає вказівку броузеру відображати текст «Програми для перегляду Web-сторінок називаються броузерами» напівжирним шрифтом.*

Тобто, контейнери впливають на частину документа, розміщену між ними. Зазначимо, що контейнери можуть бути вкладені.

Одиночний дескриптор, звичайно, має самостійне завдання, не пов’язане з конкретним текстом. Наприклад, дескриптор <*HR*> (від слів Horisontal Line) служить для створення і відображення горизонтальної лінії.

Дію дескриптора можна дещо змінити, задаючи певні *атрибути (параметри). Атрибути* — це додаткові ключові слова, які відокремлюються від ключового слова, що визначає дескриптор, пропуском і розміщуються до символа

«>». У контейнерах атрибути додаються тільки до початкового дескриптора. Атрибути задаються своїми значеннями.

Наприклад

<H1 ALIGN= "LEFT">**Транспорний рівень моделі TCP/IP *****</H1>***

- *Це вказівка броузеру відобразити текст «Транспортний рівень моделі TCP/ IP» у вигляді заголовка та вирівняти його по лівому краю.*

HTML-код Web-сторінки може містити коментарі, тобто деякий текст, який не відображається браузером і служить для пояснення призначення сторінки або частини її коду. Текст, що повинен служити коментарем, необхідно помістити між символами «<!--» і «>».

Наприклад

<HR> <!—створюємо горизонтальну лінію >

Коментар можна поставити у будь-яке місце коду сторінки, де дозволяються пропуски.

3. Групи тегів HTML

Усі теги HTML по їхньому призначенню й області дії можна розділити на наступні основні групи:

- ✓ визначаючі структуру документа;
- ✓ оформлення блоків гіпертексту (параграфи, списки, таблиці, картинки);
- ✓ гіпертекстові посилання і закладки;
- ✓ форми для організації діалогу;
- ✓ виклик програм.

Структура HTML-документа й елементи розмітки заголовка документа HTML-документ — це один великий контейнер, що починається з тега <HTML> і закінчується тегом </HTML>:

<HTML>Зміст документа</HTML>

Контейнер HTML або гіпертекстовий документ складається з двох інших вкладених контейнерів: заголовка документа (HEAD) і тіла документа (BODY).

Всі HTML-документи мають однакову загальну структуру.

```
<HTML>
    <HEAD>
        <TITLE> Назва WEB-сторінки</TITLE>
    </HEAD>
    <BODY>
        «Тіло»  – це вміст WEB-сторінки:текст, графіка, гіперпосилання
    </BODY>
</HTML>
```

Контейнер <HTML></HTML> є ознакою того, що даний файл містить документ HTML. У HTML-документах є «голова» (заголовок) і «тіло» (основна частина).

Розділ <HEAD></HEAD> містить дескриптори, які описують документ в цілому. Зокрема, тут вказується назва документа.

Контейнер <TITLE></TITLE> служить для визначення назви документа. Текст, включений в нього, відображається у верхній частині вікна браузера. Назва сторінки — це один з елементів, які мають важливе значення для привернення уваги «відвідувачів». Вона повинна відображати зміст сторінки. Системи пошуку орієнтуються саме на назву сторінки, тому сторінка із змістовною назвою має більше шансів бути вибраною у процесі проведення пошуку інформації з конкретної теми.

Контейнер <B0DY></B0DY> задає основну частину документа — його «тіло». Інформація, розміщена між дескрипторами <BODY> та </BODY>, відображається в області документа.

Питання для самоконтролю

1. Що таке тег в HTML?
2. Що називається дескриптором в HTML?
3. Що називається атрибутом в HTML?
4. Опишіть контейнер тіла в HTML.
5. На які групи поділяються теги?

<u>Тема.</u> Мова гіпертекстової розмітки HTML

План

1. Загаловок документа HTML.
2. Основні контейнери заголовка.
  2.1.Елемент розмітки HEAD
  2.2.Елемент розмітки TITLE
  2.3.Елемент розмітки BASE і LINK
  2.4.Елемент розмітки ISINDEX
  2.5.Елемент розмітки META
  2.6.Елемент розмітки STYLE
  2.7. Елемент розмітки SCRIPT

1. Загаловок документа HTML

Заголовок HTML-документа є необов'язковим елементом розмітки, однак добре складений загаловок документа може бути дуже корисним. Завданням заголовку документа є надання необхідної інформації для програми, що інтерпритує документ. Елементи, що знаходяться всередині заголовку документа

(окрім назви документа, що записуєтьмя з допомогою розділа TITLE) є невидимими на екрані (у всякому разі явно).

Заголовок документа виконує в основному такі функції:

- задає назву документа, яка відображається в стічці заголовка програми переглядача;
- визначає тип гіпертекстових зв’язків між деякими інформаційними вузлами;
- задає пошуковий образ документа для індексування роботи пошукових систем;
- за необхідністю дає вказівку браузеру створити форму для пошуку;
- керування HTTP-обміном (надання спеціальних повідомлень певному браузеру або іншій програмі перегляду);
- визначає зону видимості функцій і змінних (при використанні елемента розмітки SCRIPT).

2. Основні контейнери заголовка

Основні контейнери заголовка — це елементи HTML-розмітки, що найбільше часто зустрічаються в заголовку HTML-документа, тобто всередині елемента розмітки HEAD.

Ми розглянемо тільки вісім елементів розмітки, включаючи сам елемент розмітки HEAD:

1. HEAD (елемент розмітки HEAD);
2. TITLE (заголовок документа);
3. BASE (база URL);
4. ISINDEX (пошуковий шаблон);
5. META (метаінформація);
6. LINK (загальні посилання);
7. STYLE (опис стилів);
8. SCRIPT (скріпти).

Найчастіше застосовуються елементи TITLE, SCRIPT, STYLE. Використання елемента META говорить про інформованість автора про правила індексування документів у пошукових системах і можливості керування HTTP-обміном даними. BASE і ISINDEX останнім часом практично не застосовуються. LINK вказують тільки при використанні зовнішніх щодо даного документа описів стилів.

2.1. Елемент розмітки HEAD

Елемент розмітки HEAD містить заголовок HTML-документа. Даний елемент розмітки не є обов'язковим. При наявності тега початку елемента розмітки бажано використовувати і тег кінця елемента розмітки. За замовчуванням елемент HEAD закривається, якщо зустрічається тег початку контейнера BODY. Атрибутів у тега початку контейнера немає, хоча може бути прописаний один необов'язковий атрибут. Контейнер заголовка служить для розміщення інформації, що відноситься до всього документа в цілому.

Необов'язковий атрибут PROFILE указує на зовнішній файл META-тегів. Як значення цього атрибута вказується URL даного файлу.

2.2. Елемент розмітки TITLE

Елемент розмітки TITLE служить для іменування документа в World Wide Web. Більш прозаїчне його призначення — іменування вікна браузера, у якому проглядається документ. Складається контейнер з тега початку, змісту і тега кінця. Наявність тега кінця обов'язкове. Тег початку елемента не має специфічних атрибутів. У різних браузерах алгоритм відображення елемента TITLE може відрізнятися. При виборі тексту для змісту контейнера TITLE варто враховувати, що відображається він системним шрифтом, тому що є заголовком вікна браузера. У нелокалізованих версіях операційних систем і графічних оболонок російський текст змісту елемента TITLE буде відображатися абракадаброю.

Синтаксис контейнера TITLE у загальному виді виглядає в такий вигляд:

<TITLE>назва документа</TITLE>

Заголовок не є обов'язковим контейнером документа. Його можна опустити. Роботи багатьох пошукових систем використовують зміст елемента TITLE для створення пошукового образу документа. Слова з TITLE попадають в індекс пошукової системи. З цих розумінь елемент TITLE завжди рекомендується використовувати на сторінках Web-вузла.

2.3. Елемент розмітки BASE і LINK

У переважній більшості HTML-документи пов’язані між собою, тобто, посилаються один на одного. Посилання можуть бути, як абсолютні, так і відносні. І ті і інші мають свої недоліки. Абсолютні посилки можуть бути дуже громісткими і перестають працювати, якщо переміщено нижні по ієрархії документи. Відносні посилання легше вводити і обновляти, але ці зв’язки знову ж таки стають недієздатними якщо буде переміщено верхні по ієрархії документи. Обидва цих види зв’язків можуть порушитись при зміні дерева каталогів, в яких містяться ці HTML-документи. Тому розробники HTML, передбачаючи ці проблеми і добавили елементи BASE і LINK, які включаються в заголовок для того, щоб зв’язки між документами не порушувались. Елемент розмітки BASE служить для визначення базового URL для гіпертекстових посилань документа, заданих у неповній (часткової) формі. Крім того, BASE дозволяє визначити мішень (вікно) завантаження документа за замовчуванням при виборі гіпертекстового посилання поточного документа. Елемент BASE має один обов’язковий атрибут HREF, після якого вказуэться повна URL-адреса документа.

Елемент розмітки ***LINK*** – це результат давно початої спроби додати HTML академічний вид. Відповідно до теорії гіпертекстових систем, усі гіпертекстові зв'язки розділяють на два типи: контекстні і загальні. Такий розподіл чисто умовний і визначається тим, що контекстний зв'язок можна прив'язати до визначеного місця документа, а загальний — віднести тільки до всього документа цілком. Контекстний зв'язок визначає відношення на парі вузлів. При цьому в моделі World Wide Web один з вузлів є джерелом, а другий — мішенню. Власне, це і відбито в назві елемента розмітки <A> (anchor), що визначає гіпертекстове посилання (не плутати з гіпертекстовим зв'язком). Загальні посилання не можна прив'язати по контексту. Наприклад, два інформаційних вузли знаходяться у

відношенні проходження, тобто при «лінійному» перегляді одна Web-сторінка є наступною для іншої Web-сторінки. У цьому випадку мова йде про сторінки цілком, а не про окремі їхні частини. Таким же загальним зв'язком є приналежність до Web-вузла, що асоціюється зі своєю домашньою сторінкою.

У загальному випадку контейнер LINK має такий вигляд:

<LINK [REL=тип_відносини] [HREF=URL] [TYPE=тип_змісту]>

Для різних типів змісту дії по інтерпретації елемента розмітки будуть різними. В даний час йде процесс розробки специфікацій опису метаданных, де можливе застосування елемента розмітки LINK.

2.4. Елемент розмітки ISINDEX

Елемент розмітки ISINDEX не є найбільш вживаним елементом розділу заголовку документа HEAD. Елемент розмітки ISINDEX використовується для вказівки пошукового шаблона і успадкований від ранніх версій HTML. У HTML 4.0 цей контейнер не визначений. Опущення даного контейнера пояснюється широким застосуванням форм і CGI-скриптів. Проте всі браузери його підтримують.

У класичному варіанті при використанні ISINDEX список ключових слів, що вводяться в пошуковому шаблоні і розділені символом "+", приєднується до базової адреси HTML-документа після символу "?".

http://intuit.ru/isindex.html?keyword+list

Очевидно, що сам HTML-документ не здатний виконати пошук. Це може зробити тільки пошукова програма. Приєднання запиту до документа успадковано від першого сервера CERN (Conseil Europeen pour la Recherche Nucleaire, Європейська організація по ядерних дослідженнях), у якому воно використовувалося за аналогією з пошуковими серверами Gopher. Сучасний підхід, заснований на HTML-формах, дозволяє вказувати URL пошукової програми, що дає великі можливості при розмітці сторінок.

Сучасний синтаксис ISINDEX дозволяє застосувати аналогічний формам підхід. Для цієї мети в тезі початку контейнера ISINDEX можна вказати атрибут ACTION.

<ISINDEX ACTION=/cgi-bin/search.cgi>

Однак і традиційна форма контейнера дозволяє звертатися до зовнішніх CGI-скриптів. Зробити це можна або в сукупності з контейнером BASE, або з використанням SSI. У першому випадку для всього документа встановлюється базовий URL пошукової програми. Усі URL гіпертекстових посилань на інші сторінки задаються в повній формі або базовій адресі перепризначується після ISINDEX. Це цілком виправдано, якщо дана сторінка нічого, крім пошукового критерію і посилання на домашню сторінку Web-вузла, не містить.

```
<HTML>
   <HEAD>
      <BASE HREF=http://intuit.ru/cgi-bin/search.cgi>
      <ISINDEX>
  </HEAD>
  <BODY>
     <BASE HREF=http://intuit.ru/>
  </BODY>
```

</HTML>

У другому випадку в документ вбудовується звертання до CGI-скрипту, що реалізує функції пошукової програми. Таке сполучення — властивість сучасного підходу до компонування пошукових сторінок. Як правило, і пошуковий шаблон, і результати пошуку відображаються на одній сторінці, тому що це дозволяє коректувати запит у міру одержання результатів пошуку. Вбудований у сторінку скрипт аналізує змінні оточення сервера, і у випадку відсутності запиту може взагалі ніяк не виявляти свою присутність у середині документа.

Тег початку елемента може містити два необов'язкових атрибути: ACTION і PROMPT. Синтаксис елемента ISINDEX у загальному виді виглядає в такий вигляд:

<ISINDEX [PROMPT=текст] [ACTION=URL]>

Перший необов'язковий атрибут тега початку ISINDEX — PROMPT. Він дозволяє замість стандартного запрошення до введення ключових слів задати запрошення, що, на думку автора документа, краще відбиває суть пошукового шаблону. Наприклад, можна задати запрошення до введення ключових слів російською мовою.

2.5. Елемент розмітки META

Дескриптор <META> дозволяє автору документа описати інформацію або виконати дії, які ще не підтримуються офіційною версією HTML. Він найчастіше використовується завдяки його властивості Keywords (ключові слова). У багатьох пошукових системах використовуються слова, визначені в цьому елементі для складання вказівника документа.

Наприклад:

<META NAME=”Keywords” CONTENT =”внз, університет, фізика, математика, інформатика”>

<META NAME=”Description” CONTENT=”Броузера фізико-метематичного факультету”>

Цей дескриптор дозволяє також задати метаінформацію про кодову сторінку, яка використовується для кодування літер кирилиці:

<META content=”text/html; charset=Windows-1251” http-equiv=Content-Type>

2.6. Елемент розмітки STYLE

Елемент розмітки STYLE призначений для розміщення опису стилів. При цьому опис стилю з даного елемента розмітки, якщо він збігається по імені класу і/або ідентифікаторові підкласу зі стилем, описаним у зовнішньому файлі, заміняє опис стилю з зовнішнього файлу. З огляду впливу на весь документ, опис стилів задає правила відображення контейнерів HTML-документа для всієї сторінки.

У даний час цей контейнер використовується тільки з одним атрибутом TYPE, що задає тип опису стилю. Це може бути або text/css , або text/javascript. Якщо елемент розмітки відкритий тегом початку, то він повинен бути закритий тегом кінця. У загальному видгляді запис елемента STYLE виглядає так:

<STYLE TYPE=тип_опису_стилів>
опис стилю/стилів
</STYLE>

2.7. Елемент розмітки SCRIPT

Елемент розмітки SCRIPT служить для розміщення коду JavaScript, VBScript або JScript. Взагалі кажучи, SCRIPT можна використовувати не тільки в заголовку документа, але й у його тілі. На відміну від контейнера STYLE, йому не потрібен додатковий контейнер LINK для завантаження зовнішніх файлів кодів. Це можна зробити безпосередньо в самому контейнері SCRIPT:

<SCRIPT [TYPE=тип_мови_програмування] [SRC=URL]> JavaScript/VBScript-код
</SCRIPT>

Питання для самоконтролю

1. Як записується заголовок документу в HTML.
2. Охарактеризуйте елемент розмітки HEAD.
3. Охарактеризуйте елемент розмітки TITLE.
4. Охарактеризуйте елемент розмітки BASE і LINK.
5. Охарактеризуйте елемент розмітки ISINDEX.
6. Охарактеризуйте елемент розмітки META.
7. Охарактеризуйте елемент розмітки STYLE.
8. Охарактеризуйте елемент розмітки SCRIPT.

Тема. Контейнери тіла документу HTML

План

1. Тіло документа – контейнер BODY.
2. Теги керування розміткою:
   2.1 Заголовки.
   2.2 Абзац.
   2.3 Теги керування відображенням символів.
   2.4 Створення списків у HTML.

Теги тіла документу призначені для керування відображенням інформації в програмі інтерфейсу користувача. Вони описують гіпертекстову структуру документу за допомогою вбудованих у текст контекстних гіпертекстових посилань. Тіло документа складається з:

- ієрархічних контейнерів і заставок;
- заголовків (від H1 до H6);
- блоків (параграфи, списки, форми, таблиці, картинки і т.п.);
- горизонтальних підкреслювань і адрес;
- тексту, розбитого на області дії стилів (підкреслення, виділення, курсив);
- математичних описів, графіки і гіпертекстових посилань.

1. Тіло документа – контейнер BODY

Опис тегів тіла документа варто почати з тега BODY. На відміну від тега HEAD, тег BODY має атрибути. Атрибут BACKGROUND визначає тіло, на якому відображається текст документа. Так, якщо джерелом для тіла HTML- документа є графічний файл image.gif, то у відкриваючому тезі тіла BODY з'являється відповідний атрибут:

<BODY BACKGROUND="image.gif">

Як видно з цього приклада, як значення даного атрибута використовується адреса в скороченій формі URL. У даному випадку це адреса локального файла. Варто помітити, що різні інтерфейси користувача підтримують різні додаткові атрибути для тега BODY.

Таблиця 1. Атрибути

Атрибут	Значення
BGCOLOR=#FFFFFF	Колір тіла
TEXT=#0000FF	Колір тексту
VLINK =#FF0000	Колір пройдених гіпертекстових посилань
LINK =#00FF00	Колір гіпертекстового посилання

У даній таблиці рядок #XXXXXX визначає колір у термінах RGB у шістнадцятковому записі. Також існує можливість задавати кольори за назвою. Далі в таблиці наведені назви кольорів, визначених в стандарті HTML5.0 і відповідні їм RGB-коди. Відзначимо, що багато сучасних браузерів виходять за рамки стандартів і підтримують набагато більше назв кольорів.

Таблиця 2 Кольори

Назва	Код	Назва	Код
aqua	#00FFFF	navy	#000080
black	#000000	olive	#808000
blue	#0000FF	purple	#800080
fuchsia	#FF00FF	red	#FF0000
gray	#808080	silver	#C0C0C0
green	#008000	teal	#008080
lime	#00FF00	white	#FFFFFF
maroon	#800000	yellow	#FFFF00

Так, значення атрибутів у таблиці 2 визначають колір тексту як синій, тіла — білий, пройденого посилання червоні, а нові посилання зелені. Якщо як атрибути тега BODY вказати :

<BODY BGCOLOR=#FFFFFF TEXT=#0000FF VLINK=#FF0000 LINK=#00FF00>,

- *то колір тіла буде білим, текст буде синім, посилання — зеленими, а пройдені посилання стануть червоними.*

Однак користуватися цими атрибутами потрібно вкрай обережно, тому що в користувача може виявитися інший інтерфейс, що ці параметри не інтерпретує.

2. Теги керування розміткою

2.1 Заголовки

Заголовок позначає початок розділу документа. У стандарті визначено 6 рівнів заголовків: від H1 до H6. Текст, оточений тегами <H1></H1>, виходить

великим — це основний заголовок. Якщо текст оточений тегами <H2></H2>, то він виглядає трохи меншим (підзаголовок); текст всередині <H3></H3> ще менший і так далі до <H6></H6>. Деякі програми дозволяють використовувати більшу кількість заголовків, однак реально більше трьох рівнів зустрічається рідко, а більш 5 — вкрай рідко.

Нижче на малюнку показаний результат використання наступних заголовків:

<H1>Заголовок 1</H1>
<H2>Заголовок 2</H2>

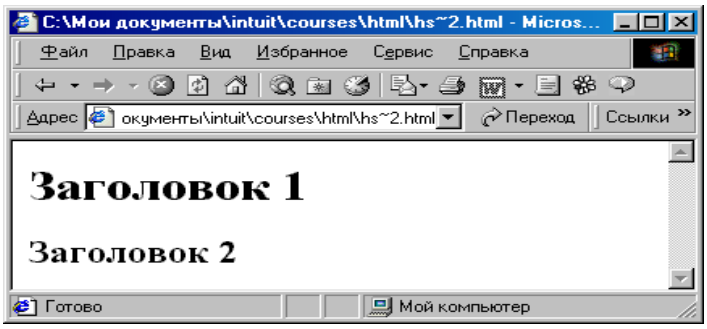


Рис. 1

2.2 Абзаци

Тег <P> застосовується для поділу тексту на параграфи. У ньому використовуються ті ж атрибути, що й у заголовках.

Атрибут ALIGN

Атрибут ALIGN дозволяє вирівняти текст по лівому або правому краї, по центрі або ширині. За замовчуванням текст вирівнюється по лівому краї. Даний атрибут застосуємо також до графіки і таблиць.

Далі наведені можливі значення атрибута ALIGN:

- **ALIGN= justify** вирівнювання по ширині.
- **ALIGN=left** вирівнювання по лівому краї. За замовчуванням текст HTML вирівнюється по лівому краї і не вирівнюється по правому. Оскільки вирівнювання по лівому краю задається автоматично, атрибут ALIGN=left можна опустити.
- **ALIGN=right** вирівнювання по правому краю.
- **ALIGN=center** центрування тексту і графіки.

Графічні об'єкти. Важливу роль в оформленні Web-сторінок відіграють графічні зображення. Включення їх у HTML-документи дозволяє надати сторінці «настрою» (ліричного, гумористичного, ділового), подати інформацію, яку неможливо отримати з тексту, розбити сторінку на «теми», що дозволить користувачеві краще орієнтуватися в матеріалі. Для того, щоб сторінка подобалась відвідувачам, необхідно добитись оптимального співвідношення форми (дизайну) і змісту.

Швидкість появи зображення на екрані залежить від розміру файла зображення. Графіка, яка повільно завантажується, може «відлякати» відвідувачів сторінок. Для зменшення розміру файла необхідно використовувати зображення типу JPEG або GIF. Конвертацію графіки у ці формати можна здійснити за допомогою редакторів растрових зображень. Допустимими зображеннями є такі, розмір яких не перевищує 40-50 Кбайт.

На HTML-сторінках графіку можна розміщувати різними способами.

Вбудовані зображення — це графічні зображення, які завжди залишаються в одному і тому самому місці сторінки (справа від тексту чи іншого об'єкта, що безпосередньо передує зображенню) і не обрамлюються текстом. Для того, щоб помістити на Web-сторінці вбудоване зображення використовується елемент <IMG SRC=URL_3ображення>. Дескриптор <IMG> є одиночним. В атрибуті SRC (від слова source) необхідно вказати повну або відносну URL-адресу файла зображення.

Наприклад:

<P>Це зображення нової моделі модема

<IMG SRC="modem8.jpg">

цей текст розміщується одразу після малюнка використано для розміщення на сторінці зображення, файл якого знаходиться в тому самому каталозі, що і сам HTML-файл (Рис.2).

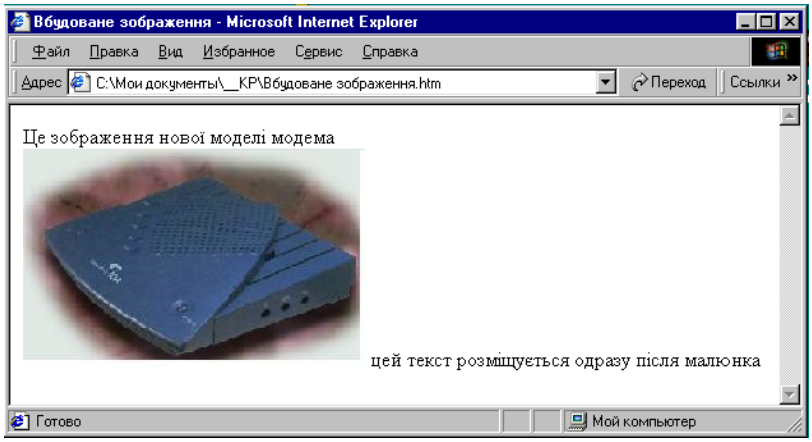


Рис.2 Графічне зображення на Web-сторінці

За замовчуванням текст вирівнюється по нижній частині зображення. Для зміни типу вирівнювання використовують дескриптор IMG з атрибутом ALIGN. Нижче наведені значення, які можна присвоїти цьому атрибуту:

Значення атрибуту	Призначення атрибуту
TOP	вирівнює текст по верхньому краю зображення;
MIDDLE	вирівнює текст по середній частині зображення;
BOTTOM	вирівнює текст по нижньому краю зображення

***Використання тега
***

Примусовий розрив рядка використовується для того, щоб порушити стандартний порядок відображення тексту. При звичайному режимі інтерпретації програма інтерфейсу користувача відображає текст у робочому вікні, автоматично розбиваючи його на рядки. У цьому режимі кінці рядків тексту ігноруються. Іноді для більшої виразності потрібно почати виведення з нового рядка. Для цього і потрібний тег BR. Атрибут CLEAR у тезі
 використовується для того, щоб зупинити в зазначеній частині обтікання об'єкту текстом і потім продовжити текст у порожній області за об'єктом. Наступний за об'єктом текст вирівнюється у відповідності зі значеннями LEFT, RIGHT або ALL атрибута CLEAR.

Елемент розмітки <NOB>

Тег <NOB> (No Break, без обриву) дає браузерові команду відображати весь текст в одному рядку, не розриваючи його. Якщо текст, вкладений у теги <NOB>, не поміститься на екрані, браузер додасть у нижній частині вікна документа горизонтальну смугу прокручування. Якщо ви хочете обірвати рядок у визначеному місці, поставте там тег
.

2.3 Теги керування відображенням символів

Усі ці теги можна розбити на два класи: теги, що керують формою відображення (font style), і теги, що характеризують тип інформації (information type). Часто зовні різні теги при відображенні дають однаковий результат. Це залежить, в основному, від налагодження інтерпретуючої програми і смаків користувача.

Теги, що керують формою відображення

Курсив, виділення, підкреслення, верхній індекс, нижній індекс, шрифт великий, маленький, червоний, синій, різні комбінації — усе це робить сторінки більш цікавими. Microsoft Internet Explorer і Netscape Navigator дозволяють визначити шрифт за допомогою ***тега FONT***. Тепер можна поєднувати на одній сторінці кілька видів шрифтів, незалежно від того, який з них заданий за замовчуванням у браузері користувача.

Теги <BI> і <SMALL> — зміна розмірів шрифту

Текст, розташований між тегами <BI></BI> або <SMALL> </SMALL>, буде, відповідно, більшим або меншим відносно стандартного.

Верхні і нижні індекси

За допомогою тегів <SUP> і <SUB> можна задавати верхні і нижні індекси, необхідні для запису торговельних знаків, символів копірайта, посилань і зносок. Розглянуті теги дозволяють створити всередині текстової області верхні або нижні індекси будь-якого розміру. Щоб вони здавалися меншими оточуючого тексту, можна використовувати теги <SUP> і <SUB> з атрибутом FONT SIZE=-1, що зменшує розмір шрифту.

Атрибут SIZE

Атрибут SIZE тега <FONT> дозволяє задавати розмір тексту в даній області. Якщо ви не користуєтеся тегом <BASEFONT SIZE=n> для задання визначеного розміру шрифту на всій сторінці, то за замовчуванням приймається 3. Деякі браузери тег <FONT> не підтримують, тому бажано вживати його тільки

всередині текстової області. В інших випадках краще використовувати теги <H1>, <H2>, <H3> і т.д. Головна перевага тега <FONT> полягає в тому, що після закінчення дії він не розбиває рядок, як теги <Hn>. Тому тег <FONT> буває дуже корисний для зміни розміру шрифту в середині рядка.

Атрибут COLOR

Якщо ви хочете зробити свою сторінку більш яскравою, можете скористатися атрибутом COLOR у тезі FONT, і тоді єдиним обмеженням буде колірна палітра на комп'ютері користувача.

Теги, що керують формою відображення, приведені в таблиці .

Таблиця 3

Тег	**Значення**
<I>...</I>	Курсив (Italic)
<B>...</B>	Виділення (Bold)
<U>...</U>	Підкреслення
<S>...</S>	Перекреслений текст
<BIG>...</BIG>	Збільшений розмір шрифту
<SMALL>...</SMALL>	Зменшений розмір шрифту
_{...}	Підрядкові символи
^{...}	Надрядкові символи

Таблиця 4. Теги, що характеризують тип інформації

Тег	**Значення**
<IM>...</IM>	Типографське виділення
<CITE>...</CITE>	Цитування
<STRONG>...</STRONG>	Виділення
<CODE>...</CODE>	Відображає приклади коду (наприклад, "коди програм")
<SAMP>...</SAMP>	Послідовність літералів
<KB>...</KB>	Приклад введення символів із клавіатури
<VAR>...</VAR>	Змінна
<DFN>...</DFN>	Визначення
<Q>...</Q>	Текст, вкладений у подвійних лапках

Ці теги допускають вкладеність і перетинання один з одним, тому усі вони мають тег початку і кінця. При використанні таких тегів варто пам'ятати, що їхнє відображення залежить від налаштування програми-інтерфейсу користувача, що можуть і не збігатися з налаштуваннями програми-розроблювача гіпертексту.

2.4 Створення списків у HTML

Списки є важливим засобом структурування тексту і застосовуються у всіх мовах розмітки. У HTML є наступні види списків: ненумерований список (невпорядкований) (Unordered Lists <UL>), нумерований список (впорядкований) (Ordered Lists <OL>) і список визначень. Теги для ненумерованих і нумерованих списків — це основа HTML.

Невпорядковані списки — тег <UL>

Ненумерований список призначений для створення тексту типу:

- ✓ перший елемент списку;

- ✓ другий елемент списку;
- ✓ третій елемент списку.

Записується даний список у виді послідовності:

```
<UL>
    <LI>перший елемент списку
    <LI>другий елемент списку
    <LI>третій елемент списку
</UL>
```

Теги <UL> і </UL> — це теги початку і кінця ненумерованого списку, тег <LI> (List Item) задає тег елемента списку.

Атрибути маркерів у ненумерованому списку

Щоб не застосовувати ті самі маркери на різних рівнях вкладеності, можна використовувати атрибут TYPE. Ви можете задати будь-який тип маркера в довільному місці списку. Можна навіть змішувати різні типи маркерів в одному списку.

Нижче, перераховані теги з атрибутами стандартних маркерів:

- **<UL TYPE=DISK>**Тег створює суцільні маркери такого типу, як у списках першого рівня за замовчуванням.
- **<UL TYPE=CIRCLE>**Тег створює маркери у виді кіл.
- **<UL TYPE=SQUARE>**Тег створює суцільні квадратні маркери.

Впорядковані списки — тег <OL>

Нумеровані списки. Тег <OL> разом з атрибутом TYPE= у HTML дозволяє створювати нумеровані списки, використовуючи як номери не тільки звичайні числа, але і рядкові і прописні букви, а також рядкові і прописні римські цифри. За необхідності можна навіть змішувати ці типи нумерації в одному списку:

<OL TYPE=1> Тег створює список з нумерацією у форматі 1., 2., 3., 4. і т.д.

<OL TYPE=A> Тег створює список з нумерацією у форматі A., B., C., D.

<OL TYPE=a> Тег створює список з нумерацією у форматі a., b., c., d. і т.д.

<OL TYPE=I> Тег створює список з нумерацією у форматі I., II., III., і т.д.

Список визначень — тег <DL>

Теги списку (Definition List: <DL>, <DT>, <DD>) використовують для створення списку термінів і їхніх визначень. Схема використання тега наступна.

```
<DL>
    <DT>Термін</DT>
        <DD>Визначення</DD>
</DL>
```

Обумовлений термін записується в одному рядку, а його визначення — з наступного, з невеликим відступом вправо. Тег <DL> дозволяє створювати окремі абзаци з відступом без нумерації або маркерів. Відступ робиться від лівого краю. Якщо на сторінці декілька тегів <DL>, то текст поступово зміщується все більше вправо. Наприкінці визначення міститься закриваючий тег </DL>. Пам'ятайте, що тег <DL> зрушує тільки ліву границю абзацу.

Горизонтальні лінійки — тег <HR>

Горизонтальне відкреслення (Horizontal Rule) застосовується для поділу документа на частини. За допомогою одного лише тега <HR> можна надати сторінці оригінальний вид. Спробуйте поекспереметувати з тегом <HR>, і ви отримаєте лінії, зовсім не схожі на ті, котрими зазвичай користуєтеся.

Переформатоване виведення — тег <PRE>

Застосування цього тега дозволяє відобразити текст «як є» (без форматування), тими ж символами і з тією же розбивкою на рядки.

Питання для самоконтролю

1. В якому тезі описується тіло документа?
2. Охарактеризуйте теги керування розміткою.
3. Якими тегами записується заголовок HTML.
4. Якими тегами записується абзац HTML.
5. Якими тегами створюються списки HTML.

Тема. Основні інструменти створення HTML сторінки

План

1. Гіпертекстові посилання.
2. Використання графіки в HTML.
3. Створення таблиць у HTML.
4. HTML-форми.

1. Гіпертекстові посилання

Усі розглянуті вище засоби керування відображенням тексту, безумовно, важливі, але вони тільки доповнюють основний тег HTML-документа — гіпертекстове посилання.

Гіперпосилання — це основа HTML. За допомогою гіперпосилань користувачі можуть переходити з однієї сторінки до іншої.

Існує три основних типи гіперпосилань: внутрішні, зовнішні та відносні.

Внутрішні гіперпосилання — це посилання на об'єкти в межах одного документа. З їх допомогою користувач може переміщатись всередині однієї Web-сторінки. Такі посилання доцільно використовувати на довгих сторінках, щоб мати можливість швидко переміщуватись між їх розділами.

Зовнішні гіперпосилання — це посилання на сторінки, розміщені на іншому Web-сервері.

Відносні* або *локальні посилання — це посилання на Web-сторінки, розміщені на тому ж сервері. Адреси цих посилань задаються відносно адреси Web-сторінки, на якій вони поміщені.

У HTML для зовнішніх гіперпосилань використовується синтаксис:

<A HREF=«URL»>Текст посилання </A>

де URL — повна адреса документа, на який здійснюється посилання. Текст, поміщений у контейнер <A HREF=«URL»> </A>, зображається в браузері з підкресленням та іншим кольором (найчастіше синім). Клацнувши на ньому лівою кнопкою миші, даємо вказівку браузеру завантажити сторінку, розміщену за адресою URL.

Наприклад:

<A HREF= «http://www.firma.com/ibm/computers.html»> Комп'ютери компанії IBM </A>

- *створюється гіперпосилання на Web-сторінку, розміщену за адресою http://www.ibm.com/ibm/computers.html.*

Зв'язуючи гіперпосиланнями сторінки одного сервера, немає необхідності включати в URL доменне ім'я.

Наприклад

<A HREF="computers.htm">Комп'ютери виробництва IBM</A>

- *створюється гіперпосилання на файл computers.htm, який повинен знаходитися на тому самому сервері і в тому самому підкаталозі, що і файл, в якому знаходиться це гіперпосилання.*

Якщо файл знаходиться в іншому підкаталозі, то необхідно вказати шлях до нього, наприклад:

<A HREF="ibm/computers.htm">Комп'ютери виробництва IBM</A>

Створюючи великі Web-сторінки, необхідно також забезпечити можливість для користувача легко переміщуватись з одного їх розділу до іншого. Це досягається завдяки внутрішнім гіперпосиланням. Для реалізації такого посилання необхідні два елементи: посилання, яке вказує, куди треба перейти, і мітка, яка фіксує місце переходу.

Посилання задається кодом:

<A HREF= "#Назва мітки">Текст посилання</A>

Мітка також використовує контейнер <A> </A> і задається так:

<A NAME ="#назва мітки">текст</A>

Назва мітки — це довільний текст, який не з'являється на екрані й однозначно визначає місце переходу. Якщо на сторінці є кілька міток, то всі вони повинні мати різні назви. Символ «#» при описі посилання вказує що за ним записана назва мітки, а не назва файла.

Можна зробити так, щоб після натискування на деякому посиланні звучала музика або відтворювався відеофільм. Посилання на мультимедійні файли .avi, .wav створюються так само, як посилання на HTML-документи. В атрибуті HREF

контейнера <A></A> необхідно вказати ім’я файла, в якому міститься потрібна мелодія або відеофільм.

Наприклад:

<P>Натисніть щоб почути мою улюблену композицію
<A HREF=”my_music.wav”>Тут</A>

Файл my_music.wav повинен знаходитись на тому самому сервері, у тому самому каталозі, що і HTML-файл, з якого він викликається. Для створення гіперпосилань на файли мультимедіа, розміщені на інших серверах Internet, використовуються повні URL-адреси.

Часто виникає необхідність включити у Web-сторінку посилання на адресу електронної пошти. Для цього в атрибуті HREF необхідно задати адресу електронної пошти у вигляді «mailto: адреса».

Наприклад:

Щоб створити посилання на адресу moi@npu. Kiev.ua, необхідно у код сторінки помістити такий дескриптор:

<A HREF=«mailto:moi@npu.kiev.ua»>moi@npu.kiev.ua</A>

згідно з яким браузер відображає текст «moi@npu.kiev.ua» як гіперпосилання. Якщо клацнути на ньому лівою кнопкою миші, браузер відкриє поштову програму в режимі створення листа, і в полі «Адреса:» помістить адресу moi@npu.kiev.ua. Текст, що поміщається в контейнер <A></A>, може бути будь-яким, проте доцільно використовувати адресу E-mail, оскільки не всі користувачі сторінки можуть користуватися поштовими програмами, налагодженими на комп’ютері, з якого переглядається сторінка. Їм необхідно прочитати реальну адресу електронної пошти.

2. Використання графіки в HTML

Для того, щоб вставити в Web-сторінку зображення, необхідно або намалювати його, або взяти вже готове. У будь-якій програмі малювання можна створити просте зображення і зберегти його в потрібному форматі. Якщо програма цей формат не підтримує, необхідно перетворити файл у необхідний формат. Існує безліч програм, призначених для перетворення одного графічного формату в інший. Запозичити ж картинки можна з різних програмних пакетів або з інших Web-сторінок у Internet, що містить бібліотеки вільного доступу художніх зображень. Коли браузер виводить Web-сторінку з зображенням, відповідний графічний файл тимчасово зберігається в пам'яті комп'ютера. У більшості браузерів є команда, що дозволяє зберегти файл на локальному диску. Існує також безліч інших варіантів одержання графічних файлів.

Зображення несуть визначену інформацію, та й просто додають Web-сторінці привабливий вигляд. Наведемо найбільш розповсюджені випадки застосування зображень:

- логотип компанії на діловій сторінці;
- графіка для рекламного оголошення;
- різні малюнки;
- діаграми і графіки;

- художні шрифти;
- підпис автора сторінки;
- застосування графічного рядка в якості горизонтальної розділової лінії;
- застосування графічних маркерів для створення красивих маркірованих списків.

Тепер розглянемо як вставити зображення в Web-сторінку. Тегом HTML, що змушує браузер виводити зображення, є <IMG> з обов'язковим атрибутом SRC (SouRCe, джерело). Ім'я файла являє собою ім'я виведеного графічного файлу. Замикаючого тега не потрібно.

Приклад вставки зображення:

<IMG SRC="image.gif" ALT="ЗОБРАЖЕННЯ">

Зображення на Web-сторінці можуть використовуватися як гіпертекстові посилання, так і як звичайний текст. Читач клацає на зображенні і відправляється на іншу сторінку або переходить до іншого зображенню. Для позначення зображення як гіпертекстової мітки використовується той же тег <A>, що і для тексту, але між <A> і </A> вставляється тег зображення <IMG> :

<A HREF="адреса файлу або зображення"> <IMG SRC="image.gif"></A>

При цьому зображення, використовуване як гіпертекстове посилання, обводиться додатковою рамкою.

Атрибути і їхні аргументи

Тег зображення має один обов'язковий атрибут SRC і необов'язкові: ALT, ALIGN, USEMAP, HSPACE, VSPACE, BORDER, WIDTH, HEIGHT.

Атрибут SRC

Указує файл зображення і шлях до нього; зображення повинне бути завантажене в браузер і розміщено в тому місці документа, де розташований тег зображення.

Атрибут ALT

Дозволяє вказати текст, що буде виводитися замість зображення браузерами, нездатними виводити графіку. У деяких випадках при недостатній пропускній здатності ліній зв'язку користувачі відключають відображення графіки. Наявність назв замість картинок полегшує сприйняття Web-сторінок у такому режимі.

Атрибут ALIGN

Визначає положення зображення щодо навколишнього його тексту. Можливі значення аргументу — ["top" | "middle" | "bottom"] (відповідно, "вгорі", "посередині", "внизу").

Атрибут USEMAP

Якщо присутні атрибут USEMAP і тег <MAP>, зображення стає чуттєвою картою, або "графічним меню". Якщо клацнути кнопкою миші на активній області зображення, для якого визначений атрибут USEMAP, відбудеться гіпертекстовий перехід до інформаційного ресурсу, установленому для цієї області. Більш докладно це питання буде висвітлено нижче.

Атрибут BORDER

Цілочисельне значення аргументу визначає товщину рамки навколо зображення. Якщо значення дорівнює нулеві, рамка відсутня. Щоб не вводити

користувачів в оману, не варто використовувати BORDER=0 у зображеннях, що являють собою частину елемента посилання, оскільки малюнки, застосовувані як гіперпосилання, звичайно виділяються кольоровою рамкою.

Атрибут HSPACE

Цілочисельне значення цього атрибута задає горизонтальну відстань між вертикальною границею сторінки і зображенням, а також між зображенням і текстом, що його обгинає.

Атрибут VSPACE

Цілочисельне значення цього атрибута задає вертикальну відстань між рядками тексту і зображенням.

Атрибути WIDTH і HEIGHT

Обидва атрибути задають цілочисельні значення розмірів зображення по горизонталі і по вертикалі відповідно. Це дозволяє зменшити час завантаження сторінки з графікою. Браузер відразу відводить рамку для зображення і продовжує завантажувати текст на сторінку. Поки завантажується графіка, користувач може почати читати текст. Визначити розмір зображення неважко, для цього досить скористатися будь-як програмою перегляду графічних файлів, наприклад ACDSee або графічним редактором Corel PhotoPaint, або Adobe Photoshop. Відкрийте файл у графічному редакторі і визначите розмір картинки в пікселях. У тезі зображення варто вказати ширину і висоту картинки.

<IMG SRC="image.gif" ALT="зображення" WIDTH="100" HEIGHT="200"

HSPACE="10" SPACE="10" BORDER="2" ALIGN="left">

Активні зображення

Активні зображення (image maps), або зображення, чуттєві до щигликів миші, дозволяють створити на сторінці графічні меню довільної форми. Активне зображення — це зображення з так називаними активними областями (hot spots), що посилаються на URL інших сторінок або вузлів.

Є два методи формування активних зображень: на сервері і в клієнта. Зображення першого типу використовують сервер для того, щоб знайти відповідний до даної активної області URL і передати браузеру потрібну сторінку. Активні зображення, що працюють на клієнтській машині, задають інформацію про активну область на HTML-страниці, так що браузер сам з'ясовує, які області є активними, і запитує у сервера відповідну сторінку.

Активні зображення, що працюють у клієнта, мають кілька переваг.

По-перше, сторінки з ними можна перенести на інший сервер. По-друге, серверові не потрібно виконувати зайву роботу (наприклад, переглядати всю інформацію про активні області), тобто навантаження на сервер зменшується. Під час використання працюючих на сервері активних зображень у каталозі cgi-bin сервера повинний бути відповідний сценарій. З розумінь безпеки багато системних адміністраторів не записують сценарії в каталог cgi-bin. Тому більш докладно 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 потрібно зробити активними, а потім співставити їх з посиланнями на

інші URL. Активні області задаються перерахуванням їхніх координат (у пикселах). Усе це можна зробити вручну, визначивши координати кутів активних областей, але набагато простіше скористатися якою-небудь програмою, наприклад Mapedit.

Визначити карту легко. Потрібно відкрити в Mapedit HTML-файл, що містить зображення, на якому потрібно створити активні області, після чого обране зображення буде завантажено в робоче вікно. Потім варто вибрати тип активної області (квадрат, трикутник і коло), клацнути і потягнути мишкою, позначивши границю області. Програма автоматично робить запис у HTML-файл, що описує границі активної області. Потім для цієї області потрібно прописати URL. У будь-яких місцях зображення можна намалювати активні області і визначити для кожної з них URL. Важливо залишати між областями небагато місця, щоб при читанні бути впевненим, що активізується правильне посилання. Границі активних областей задаються координатами кутів прямокутника і багатокутника або центра і радіуса кола. Якщо ви вирішили робити активне зображення в клієнта, Map Edit надає 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 "#" у такий спосіб:

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

Активні зображення в клієнта працюють незалежно від програмного забезпечення сервера і не перестануть функціонувати, навіть якщо файли будуть перенесені на інший сервер.

Зображення в мініатюрі

Часто для ілюстрації якоїсь теми потрібні зображення великого розміру, завантажуватися вони будуть досить довго. У тому місці, де потрібно розмістити великий малюнок, можна помістити маленьку його копію і зробити посилання на повномасштабне зображення. Ті відвідувачі, яким це дійсно цікаво, зможуть подивитися зображення цілком, а всі інші пролиснуть сторінку, не затримуючись. Така методика особливо гарна для обкладинок книг, фотографій, рекламних листків, що не всі читачі захочуть вивчити в деталях.

Щоб використовувати мініатюри, необхідно мати дві копії зображення — велике зображення-оригінал і зменшене зображення. Для зменшення розмірів зображень використовують можливості графічних програм. Щоб зв’язати мініатюру з оригіналом необхідно помістити дескриптор, що задає мініатюру, всередині дескриптора <A HREF> </A>, який є гіперпосиланням на велике зображення.

<A HREF= «big.gif»> <IMG SRC= «small.gif»> </A>

2. Створення таблиць у HTML

Таблиця є частиною HTML-документа. Вона являє собою прямокутну сітку, що складається з вертикальних стовпців і горизонтальних рядків.

Перетин рядка і стовпчика називається коміркою таблиці. Комірка може містити в собі текст, графіку або іншу таблицю.

Таблиця складається з трьох основних частин:

- назви таблиці,

- заголовків стовпців,
- комірок таблиці.

Таблиця в Web-документі заповнюється по рядках (зліва направо по рядку, потім перехід на новий рядок).

Кожна комірка таблиці має бути заповнена (хоча б пробілами, які використовуються для створення порожніх клітинок).

Для додавання таблиці на веб-сторінку використовується тег <table>. Цей елемент служить контейнером для елементів, які визначають вміст таблиці. Будь-яка таблиця складається з рядків і клітинок, які задаються відповідно за допомогою тегів <tr> і <td>. Таблиця повинна містити хоча б одну клітинку. Допускається замість тега <td> використовувати тег <th>. Текст в клітинці, оформленій за допомогою тега <th>, браузер відображає шрифтом жирного накреслення і вирівнює по центру клітинки. В іншому, різниці між клітинками, створеними через теги <td> і <th> немає.

Наприклад

```
<!DOCTYPE html>
<html>
      <head>
        <title> тег table </title>
      </head>
      <body>
        <table border = "1"  width = "50%" cellpadding = "5">
          <tr>
            <th> клітинка 1 </th>
            <th> клітинка 2 </th>
          </tr>
          <tr>
            <td> клітинка 3 </td>
            <td> клітинка 4 </td>
          </tr>
        </table>
      </body>
</html>
```

Результат роботи

клітинка 1	клітинка 2
клітинка 3	клітинка 4

Атрибути тега <table>

Той факт, що таблиці застосовуються досить часто і не тільки для відображення табличних даних, зобов'язаний не тільки їх гнучкості і універсальності, а й різноманітності атрибутів тегів <table>, <tr> і <td>. Далі перераховані деякі атрибути тега <table>, які застосовуються найчастіше.

align − задає вирівнювання таблиці по краю вікна браузера. Можна вибрати зі значень *left* − вирівнювання таблиці по лівому краю, *center* − по центру і *right* − по правому краю. Результат буде помітний лише в тому випадку, якщо ширина таблиці не використовує усю доступну область, тобто менша ніж 100% сторінки. Окрім цього, align не тільки встановлює вирівнювання, а й змушує текст обтікати таблицю з вільного боку.

- ***bgcolor*** − встановлює колір фону таблиці.
- ***border*** − встановлює товщину границі в пікселях навколо таблиці. При наявності цього атрибута також відображаються границі між клітинками.
- ***cellpadding*** − визначає відстань між границею клітинки і її вмістом. Цей атрибут додає порожній простір до клітинки, збільшуючи тим самим її

розміри. Без cellpadding текст в таблиці «налипає» на рамку, погіршуючи тим самим його сприйняття. Додавання ж cellpadding дозволяє поліпшити читабельність тексту. При відсутності границь особливого значення цей атрибут не має, але може допомогти, коли потрібно встановити порожній проміжок між клітинками.

- ***cellspacing*** − задає відстань між зовнішніми границями клітинок. Якщо встановлений атрибут border, товщина границі приймається в розрахунок і входить в загальне значення.
- ***cols*** − вказує кількість стовпців в таблиці, допомагаючи браузеру в підготовці до її відображення. Без цього атрибута таблиця буде показана тільки після того, як весь її вміст буде завантажено в браузер і проаналізовано. Використання атрибуту cols дозволяє дещо прискорити відображення вмісту таблиці.
- ***rules*** − повідомляє браузеру, де відображати границю між клітинками. За замовчуванням рамка малюється навколо кожної клітинки, утворюючи тим самим сітку. На додаток можна вказати відображати лінії між колонками (значення cols), рядками (rows) або групами (groups), які визначаються наявністю тегів <thead>, <tfoot>, <tbody>, <colgroup> або <col>. Товщина границі вказується за допомогою атрибута border.
- ***width*** − задає ширину таблиці. Якщо загальна ширина вмісту перевищує зазначену ширину таблиці, то браузер буде намагатися «втиснутися» в задані розміри за рахунок форматування тексту. У разі, коли це неможливо, наприклад, в таблиці знаходяться зображення, атрибут width буде проігнорований, і нова ширина таблиці буде обчислена на основі її вмісту.

Атрибути тега <td>

Кожна клітинка таблиці, яка задається через тег <td>, в свою чергу теж має свої атрибути, частина з яких збігається з атрибутами тега <table>, наприклад, align та bgcolor, значення яких аналогічні атрибутам тега <table>. Але, використовуючи атрибут bgcolor для клітинки одночасно з цим же атрибутом таблиці, можна отримати в таблиці різноманітні колірні ефекти.

colspan − встановлює число клітинок, які повинні бути об'єднані по горизонталі. Цей атрибут має сенс для таблиць, які складаються з декількох стовпців.

Приклад таблиці з горизонтальним об'єднанням клітинок

```
<!DOCTYPE html>
<html>
  <head>
    <title> Об'єднання клітинок </title>
  </head>
  <body>
    <table border = "1"  width = "50%" cellpadding = "5">
      <tr>
        <th colspan="2"> клітинка 1 </th>
      </tr>
      <tr>
        <td> клітинка 2 </td>
        <td> клітинка 3 </td>
        </tr>
    </table>
  </body>
</html>
```

Результат

клітинка 1	
клітинка 2	клітинка 3

У наведеній вище таблиці містяться два рядки і дві колонки, причому верхні горизонтальні клітинки об'єднані за допомогою атрибута colspan.

- ***height*** − встановлює висоту клітинок. Браузер сам встановлює висоту таблиці і її клітинок, виходячи з їх вмісту. Однак при використанні атрибута height висота клітинок буде змінена. Тут можливі два варіанти. Якщо значення height менше, ніж вміст клітинки, то цей атрибут буде проігнорований. У разі, коли встановлена висота клітинки, яка перевищує її вміст, додається порожній простір по вертикалі.
- ***rowspan*** − встановлює число клітинок, які повинні бути об'єднані по вертикалі. Цей атрибут має сенс для таблиць, які складаються з декількох рядків.
- ***valign*** − встановлює вертикальне вирівнювання вмісту клітинки. За замовчуванням вміст клітинки розташовується по її вертикалі в центрі. Можливі значення: top − вирівнювання по верхньому краю рядка, middle − вирівнювання по середині, bottom − вирівнювання по нижньому краю, baseline − вирівнювання по базовій лінії, при цьому відбувається прив'язка вмісту клітинки до однієї лінії.
- *width* − задає ширину клітинки. Сумарне значення ширини всіх клітинок може перевищувати загальну ширину таблиці тільки в тому випадку, якщо вміст клітинки пе
- ревищує зазначену ширину.

Заголовок таблиці

При великій кількості таблиць на сторінці кожній з них необхідно задати заголовок, який містить назву таблиці і її опис. Для цієї мети в HTML існує спеціальний тег ***<caption>***, який встановлює текст і його положення щодо таблиці.

Найпростіше розміщувати текст по центру таблиці зверху або знизу від неї, в інших випадках браузери по різному інтерпретують атрибути тега <caption>, через що результат виходить неоднаковий. За замовчуванням заголовок

поміщається зверху таблиці по центру, його ширина не перевищує ширини таблиці і в разі довгого тексту він автоматично переноситься на новий рядок.

Для зміни положення заголовка у тега <caption> існує атрибут align, який може набувати наступних значень:

- ***left*** – вирівнює заголовок по лівому краю таблиці. Браузер Firefox розміщує текст збоку від таблиці, Internet Explorer і Opera розташовують заголовок зверху, вирівнюючи його по лівому краю.
- ***right*** – в браузері Internet Explorer і Opera заголовок розташовується зверху таблиці і вирівнюється по правому краю. Firefox відображає заголовок праворуч від таблиці.
- ***center*** – заголовок розташовується зверху таблиці по її центру. Таке розташування задано в браузерах за замовчуванням.
- ***top*** – результат аналогічний дії атрибута center, але на відміну від нього входить в специфікацію HTML4 і розуміється всіма браузерами.
- ***bottom*** – заголовок розміщується внизу таблиці по її центру.

4. HTML-форми

Форми є невід'ємною частиною Інтернету, тому що пропонують сайтам метод збору інформації від користувачів і обробки запитів, а також елементи управління практично для будь-якого можливого застосування. За допомогою елементів управління або полів, форми можуть запросити невеликий обсяг інформації − часто це пошуковий запит, ім'я користувача або пароль, або великий обсяг інформації − можливо, дані про посилку, платіжна інформація або пропозиція роботи.

Ми повинні знати, як створювати форми, щоб отримати вхідні дані від користувача. У цьому розділі ми обговоримо, як використовувати HTML для розмітки форми, які елементи використовувати для захоплення різних типів даних.

Ініціалізація форми

Щоб додати форму на сторінку будемо використовувати елемент ***<form>*** . Даний елемент визначає, де на сторінці з'являться елементи управління. Крім того, елемент <form> обгортає всі елементи включені в форму, подібно елементу .

```
<form action="login" method="post">
  ...
</form>
```

До елементу <form> може застосовуватися декілька різних атрибутів, найбільш поширеними з яких є action і method. Атрибут action містить URL, на який інформація у формі буде відправлена для обробки сервером. Атрибут method є методом HTTP, який повинні використовувати браузери для

відправки даних форми. Обидва ці атрибута <form> мають відношення до відправки і обробки даних.

Атрибут method має два значення − get і post.

- ***get*** − є одним з найпоширеніших методів і призначений для отримання необхідної інформації та передачі даних в адресному рядку. Пари «ім'я = значення» приєднуються в цьому випадку до адреси після знаку питання і розділяються між собою амперсандом (символ &). Зручність використання методу get полягає в тому, що адресу з усіма параметрами можна використовувати неодноразово, зберігши її, наприклад, в закладки браузера, а також змінювати значення параметрів прямо в адресному рядку. Це значення встановлено для атрибута method за замовчуванням.
- ***post*** − метод, який посилає на сервер дані в запиті браузера. Це дозволяє відправляти більшу кількість даних, ніж за методом get, оскільки у нього встановлено обмеження в 4 Кб. Великі обсяги даних використовуються у форумах, поштових службах, заповненні бази даних, при пересиланні файлів тощо.

Текстові поля і текстові області

Коли справа доходить до збору текстової інформації від користувачів, є кілька різних елементів, доступних для отримання даних в формах. Зокрема, для збору даних на основі тексту або рядків застосовуються текстові поля і текстові області. Ці дані можуть включати в себе уривки тексту, паролі, номери телефонів та іншу інформацію.

Текстові поля

Одним з основних елементів, які використовуються для отримання тексту від користувачів, є елемент <input>. Даний елемент включає атрибут type для визначення, який тип інформації буде отримано в елементі управління. Найбільш популярне значення атрибута type – це text, який позначає введення одного рядка тексту. Поряд з установкою атрибута type, гарною практикою буде також дати елементу <input> атрибут name. Значення атрибута name застосовується в якості імені елементу управління і відправляється разом з вхідними даними на сервер.

Атрибути текстового поля

- ***name*** − ім'я поля, призначене для того, щоб обробник форми міг його ідентифікувати.
- ***Disabled*** − блокує поле для введення тексту і не відправляє дані на сервер.
- ***Form*** − ідентифікатор форми для зв'язування текстового поля з елементом <form>.
- ***Type*** − перелік значень для текстого поля, який наведено далі.
- ***Maxlength*** − встановлює максимальне число символів, яке може бути введено користувачем у текстовому полі. Коли ця кількість досягається при

наборі, подальше введення стає неможливим. Якщо даний атрибут опустити, то можна вводити рядок довший ніж саме поле.

- ***Readonly*** − блокує поле для введення тексту.
- ***Size*** − ширина текстового поля, яка визначається числом символів моноширинного шрифту. Зазвичай не вказують, тому що через стилі задати бажані розміри простіше і точніше.
- ***Value*** − початковий текст, який відображається в полі.
- ***Autocomplete*** − введений раніше текст запам'ятовується браузером і підставляється при наступному введенні.
- ***Autofocus*** − поле отримує фокус після завантаження документа.
- ***List*** − значення атрибута id елемента <datalist>, для зв'язку з цим елементом.
- ***Pattern*** − шаблон введення тексту. Вказує регулярні вирази, відповідно до яких треба вводитти і відправляти дані форми.
- ***Required*** − вказує, що поле є обов'язковим для заповнення.
- ***Placeholder*** − додає підказку, яка зникає при введенні тексту.
- ***Dirname*** − ім'я змінної, яка відправляється на сервер і автоматично отримує значення ltr (для тексту зліва направо) або rtl (для тексту справа наліво).

Приклад: Текстове поле

```
<form action="/login" method="post">
  <input type="text" name="username" placeholder="login"
  maxlength="25" size="40" >
</form>
```

Елемент ***<input>*** є самостійним, тобто він задіє тільки один тег і не обертає ніякого контенту. Значення елемента забезпечується його атрибутами і їх відповідними значеннями. Спочатку було тільки два текстових значення атрибута type – text і password (для введення паролів), проте стандарт HTML5 додав кілька нових значень атрибута type.

Ці значення були додані, щоб забезпечити чітке смислове значення для полів вводу, а також надати краще управління користувачам. Якщо браузер не розуміє одне з цих HTML5-значень атрибута type, він автоматично повернеться до значення text. На наступному рисунку наведено список нових типів HTML5 та декілька прикладів.

Текстові області

Ще одним елементом, який використовувується для збору текстових даних, є елемент ***<textarea>***. Він відрізняється від елемента ***<input>*** тим, що може приймати великі уривки тексту в декілька рядків. Елемент <textarea> також містить відкриваючий і закриваючий теги, які можуть обернути простий текст, який буде відображено у текстовому полі. При цьому у браузері буде відображено усі пробіли та переноси, які є у html-коді. Оскільки <textarea> приймає тільки один тип значення, атрибут type тут не застосовують, але атрибут name використовується як і раніше.

Можливе використання наступних атрибутів:

- **<rows>** − висота поля в рядках тексту.
- ***<wrap>*** − значення soft передає на сервер текст одним рядком, а hard враховує значення cols і автоматично додає переноси.
- ***<cols>*** − ширина поля в символах.

Як правило, розміри <textarea> задаються за допомогою стилів, але якщо одночасно задані атрибути rows і cols з шириною і висотою через CSS, то стилі мають перевагу і значення атрибутів ігнорується.

Поля множинного вибору і меню

Крім текстових полів, HTML також дозволяє користувачам вибирати дані, використовуючи множинний вибір та списки, які випадають. Є кілька різних варіантів і полів для цих елементів форми, кожен з яких має свої відмінності та переваги.

Перемикачі

Це простий спосіб, який дозволяє користувачам зробити швидкий вибір з невеликого списку варіантів. Перемикачі дають користувачеві вибрати тільки один варіант з кількох.

Щоб створити перемикач, використовується елемент <input> зі значенням ***radio*** атрибута type. Кожен перемикач повинен мати однакове значення атрибуту name, щоб усі вони в групі були пов'язані один з одним.

З текстовими полями їх значення визначається тим, що користувач в них набирає; з перемикачами користувач робить вибір з деякої множини значень. Таким чином, ми повинні визначити вхідні значення. Використовуючи атрибут value ми можемо встановити конкретне значення для кожного елемента <input>.

Крім того, для попереднього вибору деякого значення ми можемо використовувати логічний атрибут ***checked***.

Приклад: Перемикачі

```
<form action="radio.php" method="post">
  <input type="radio" name="day" value="Friday" checked> П'ятниця
  <input type="radio" name="day" value="Saturday"> Субота
  <input type="radio" name="day" value="Sunday"> Неділя
</form>
```

Прапорці

Прапорці дуже схожі на перемикачі. Вони використовують ті ж атрибути і шаблони, за винятком значення атрибута type. Різниця між ними полягає в тому, що прапорці дозволяють користувачам вибрати кілька значень і зв'язати їх усі з одним ім'ям, в той час як перемикачі обмежують користувачів одним значенням.

Приклад: Прапорці

```
<form action="checkbox.php" method="post">
  <input type="checkbox" name="day" value="Friday" checked> П'ятниця
  <input type="checkbox" name="day" value="Saturday"> Субота
  <input type="checkbox" name="day" value="Sunday"> Неділя
</form>
```

Списки, які випадають

Списки, які випадають, є ідеальним способом, щоб практично надати користувачам довгий список варіантів. Довгий стовпець перемикачів поряд зі списком різних варіантів не тільки візуально непривабливий, але крім того складний і важкий для розуміння, особливо на мобільному пристрої. З іншого боку, списки, які випадають, забезпечують ідеальний формат для довгого списку варіантів.

Для створення списку застосовують елементи ***<select>*** і ***<option>***. Елемент <select> огортає всі пункти меню, а кожен пункт меню розмічений за допомогою елемента <option>. Атрибут name розташовується в елементі <select>, а атрибут value розташовується в елементах <option>, вкладених в елемент <select>. Таким чином, атрибут value у кожному елементі <option> пов'язаний з атрибутом name елемента <select>.

Кожен елемент <option> огортає текст, який видно користувачам, окремого пункту в списку.

Подібно логічному атрибуту ***checked*** у перемикачів і прапорців, для випадаючого меню можна використовувати логічний атрибут ***selected***, щоб попередньо виділити пункт для користувачів.

Приклад: списки, які випадають

```
<form action="menu.php" method="post">
  <select name="day">
    <option value="Friday" selected>П'ятниця</option>
    <option value="Saturday">Субота</option>
    <option value="Sunday">Неділя</option>
  </select>
</form>
```

Множинний вибір

Логічний атрибут multiple при додаванні до елементу <select> для стандартного списку дозволяє користувачеві вибрати більше одного варіанта зі списку одночасно. Крім того, за допомогою логічного атрибута selected, доданого до більш ніж одного елемента <option>, в меню можна заздалегідь вибрати кілька варіантів.

Розміром елемента <select> можна керувати за допомогою CSS і він повинен бути скорегований відповідним чином для множинного вибору. Можливо, є сенс інформувати користувачів, що для вибору декількох варіантів вони повинні утримувати клавішу Shift під час клацання мишкою, щоб зробити вибір.

Приклад: Множинний вибір

```
<form action="menu.php" method="post">
  <select name="day" multiple>
    <option value="Friday" selected>П'ятниця</option>
    <option value="Saturday" selected>Субота</option>
    <option value="Sunday">Неділя</option>
  </select>
</form>
```

Кнопки

Кнопки є одним з популярних елементів інтерфейсу і часто застосовуються всередині форм, наприклад, для відправки введеної інформації в формі на сервер.

Кнопки на веб-сторінці можна створити декількома способами − за допомогою елемента ***<input>*** або елемента ***<button>***.

Розглянемо спочатку додавання кнопки через <input> і його синтаксис.

```
<input type="button" value="Текст на кнопці">
```

Атрибути кнопки <input>:

- ***Name*** − ім'я кнопки, призначене для того, щоб обробник форми міг її ідентифікувати.
- ***Disabled*** − блокує кнопку і не дозволяє на неї натискати.
- ***Form*** − ідентифікатор форми для зв'язування кнопки з елементом <form>.
- **Type** − для звичайної кнопки значенням є button.
- ***Value*** − напис на кнопці.
- ***Autofocus*** − кнопка отримує фокус після завантаження документа.

Другий спосіб створення кнопки заснований на використанні елемента <button>. Він за своєю дією нагадує результат, одержаний за допомогою <input>. Але, на відміну від нього, пропонує розширені можливості по створенню кнопок. Наприклад, на подібній кнопці можна розміщувати будь-

які елементи HTML, включаючи зображення і таблиці. Синтаксис створення такої кнопки наступний.

```
<button>Напис на кнопці</button>
```

Атрибути, перераховані для кнопки <input>, діють і у цьому випадку, але атрибут value визначає тільки значення, яке відправляється на сервер, а не напис на кнопці. Якщо потрібно вивести на кнопці зображення, то <img> додається всередину <button>, як показано в наступному прикладі.

Для відправки даних на сервер призначена спеціальна кнопка submit. Її вигляд нічим не відрізняється від звичайних кнопок, але при натисканні на неї відбувається перехід до обробника форми за адресою, вказаною атрибутом action елементу <form>. Програма-обробник, отримує дані, введені користувачем в полях форми, проводить з ними необхідні маніпуляції, після чого повертає результат у вигляді HTML-документа. Що саме робить оброблювач, залежить від автора сайту, наприклад, така технологія застосовується при створенні опитувань, форумів, тестів тощо.

Синтаксис створення кнопки submit залежить від використовуваного елемента <input> або <button>.

```
<input type="submit">
<button type="submit">Напис на кнопці</button>
```

Кнопка reset

При натисканні на кнопку ***reset*** дані форми повертаються до первинних значень. Як правило, цю кнопку застосовують для очищення введеної в полях форми інформації. Для великих форм від використання кнопки reset краще взагалі відмовитися, щоб помилково на неї не натиснути, адже тоді доведеться заповнювати форму заново.

Синтаксис створення зазначеної кнопки простий і схожий на інші кнопки.

```
<input type="reset">
<button type="reset">Напис на кнопці</button>
```

Значення кнопки reset ніколи не пересилається на сервер. Якщо напис на кнопці опустити, іншими словами, не ставити атрибут value, на кнопці за замовчуванням буде додано текст «Скинути».

Організація елементів форми

Знати, як отримати дані з полів форми, це лише половина справи. Інша половина − це організація елементів форми і полів в зручному порядку. При взаємодії з формами користувачі повинні зрозуміти що від них вимагається і як надати запитувану інформацію.

За допомогою ***<label>, <fieldset>*** і ***<legend>*** ми можемо краще організувати форми і направляти користувачів правильно їх завершувати.

<label>

Елемент <label> містить підписи або заголовки для управління елементами форми, однозначно пов'язуючи їх, створюючи тим самим доступну форму для всіх користувачів і допоміжних технологій. <label> повинні включати в себе текст, який описує поля, до яких вони належать.

Елементи <label> можуть включати в себе атрибут for, його значення має бути таким же, як значення атрибута id у елемента, з яким пов'язаний <label>. Відповідність значень атрибутів for і id пов'язує два елементи разом, що дозволяє користувачам натиснути на <label> і передати фокус потрібному полю форми. У такому випадку синтаксис наступний:

```
<form method="post">
  <label for="user_name">Ім'я користувача</label>
  <input type="text" name="username" id="user_name">
</form>
```

У браузері така форма буде виглядати наступним чином:

При бажанні, у <label> можна огорнути поля форми, такі як перемикачі або прапорці. Це дозволяє опустити атрибути for і id.

<fieldset>

<fieldset> групує поля форми в організовані розділи подібно <section> або іншим структурним елементам. Однак <fieldset> є блочним елементом, який огортає пов'язані елементи в <form>, для їх кращої організації. Елементи <fieldset> за замовчуванням також включають в себе границі контуру, які можуть бути змінені за допомогою CSS.

<legend>

Елемент <legend> додає підпис або заголовок для елемента <fieldset>. Елемент <legend> огортає текст, який описує елементи управління форми, які знаходяться всередині <fieldset>. Розмітка повинна включати в себе елемент <legend> відразу після відкриваючого тега <fieldset>. На сторінці підпис з'явиться в лівому верхньому кутку рамки <fieldset>.

Питання для самоконтролю

1. Як задати гіперпосилання в документі HTML.
2. Для чого використовується графіка в HTML.
3. Що таке форма в документі HTML?

4. Для чого призначена форма в документі HTML.
5. Як створити таблицю?
6. Як об’єднати клітинки у таблиці?
7. Як задати товщину зовнішньої границі таблиці?
8. Як задати внутрішні поля для клітинок таблиці?
9. Чим відрізняються теги th і td?
10. Для чого призначений тег caption? Які його атрибути ви знаєте?
11. Як поводить себе клітинка із заданими розмірами, якщо її вміст перевищує ці розміри?
12. Який тег використовують для створення форм?
13. Назвіть відмінності для тегів <textarea> та <input>.
14. Як організувати підказки для текстового поля?
15. Які типи текстового поля ви знаєте?
16. Який атрибут не дозвляє вводити дані у текстову форму?
17. За допомогою якого тегу створюються списки?
18. Як створити ненумерований список?
19. Як створити список визначень?
20. За допопогою якого атрибуту визначається маркер або позначення нумерації списку?
21. Як почати нумерацію із заданої позиції?
22. Як створити пермикач?
23. Як створити список з вибором декількох варіантів?
24. За допомогою яких тегів організовують меню зі списком, який випадає?
25. Який атрибут допомагає помітити наперед заданий варіант для перимикачів та прапорців? Для меню?
26. За допомогою якого атрибуту організовують множинний вибір у меню?
27. Які елементи використовують для організації елементів форми?
28. Для чого використовують елемент <fieldset>?
29. За допомогою якого тегу додають напис до логічно об'єднаних полів форми?
30. За допомогою якого тегу їх об'єднують?
31. Який елемент використовують щоб зв'язати поле форми і текст, який ним управляє?
32. Яка різниця існує між атрибутами placeholder і value?
33. Який атрибут забороняє введення даних у поле форми?
34. За допомогою якого атрибуту поле форми позначають як обов'язкове для заповнення?

Тема. Технологія каскадних таблиць стилю CSS

План

1. Теоретичні відомості про CSS.
2. Призначення CSS.
3. Способи застосування CSS.

1. Теоретичні відомості про CSS та їх призначення

Дизайн Web-вузлів — це точне розміщення компонентів HTML-сторінок відносно один одного в робочій області вікна браузера.

Недоліки такого визначення Web-дизайну очевидні. У ньому не враховані ні колір, ні форма, ні інші властивості компонентів HTML-сторінок. Головне в цьому визначенні — показати обмеженість можливостей HTML-розмітки. Позиціонування компонентів на сторінці є одним із самих слабких місць HTML.

До компонентів сторінки відносяться: блоки тексту, графіка й вбудовані додатки. Розмір і границі кожного з цих компонентів у рамках HTML-розмітки задаються з різним ступенем точності. Розмір графіки і додатків можна задати з точністю до пікселя. Розміри текстових блоків у HTML задати не можна: вони визначаються браузером виходячи з відносного розміру шрифту за замовчуванням.

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

Каскадні таблиці стилів призначені усунути протиріччя між точністю визначення розмірів картинок і додатків, з одного боку, і точністю визначення розмірів блоків тексту і його накреслення — з іншої.

Таблиці стилів також дозволяють визначити колір і накреслення текстового фрагменту, змінювати ці параметри всередині текстового блоку, виконувати вирівнювання текстового блоку щодо інших блоків і компонентів сторінки.

Існування подібних можливостей дозволяє говорити про CSS як про засіб поділу логічної структури документа і форми його представлення. Логічна структура документа визначається елементами HTML-розмітки, у той час як форма представлення кожного з цих елементів задається CSS-описом елемента.

CSS дозволяє цілком перевизначити форму представлення елемента розмітки за замовчуванням.

Розглянувши в такий спосіб роль і призначення CSS серед різноманіття Web-технологій, ми переходимо безпосередньо до обговорення застосування каскадних таблиць стилів.

2. Способи застосування CSS

Під способами застосування CSS в даному розділі розумітимемо форму декларування стилю на HTML-сторінці і форму зв'язування опису стилю відображення елемента розмітки із самим елементом. Мова йде про те, де й у якій формі автор сторінки (або дизайнер) описує стиль, і як і в якій формі його використовує.

Отже, розрізняють чотири способи застосування стилів:

- ***вбудовування*** у теги документа – дозволяє змінити форматування конкретних елементів сторінки;
- ***впровадження*** – дозволяє задавати всі правила таблиці стилів в самому документі;
- ***зв'язування*** – дозволяє використовувати одну таблицю стилів для форматування багатьох сторінок;

- ***імпортування*** – дозволює вбудовувати у документ таблицю стилів, розміщену на сервері.

Вбудовування

```
<H1 STYLE="font-weight:normal; font-style:italic; font-size:10pt;">

Заголовок першого рівня
</H1>
```

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

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

Очевидно, що не всі параметри стилю можна встановити для конкретного елемента розмітки. Про типи елементів і відповідні їм параметри стилів ми поговоримо в пункті "Поняття блочного і стрічкового елементів".

Необхідно відзначити наступне: стилі розроблені, в першу чергу, для керування відображенням тексту. Не слід захоплюватися стилями для керування відображенням нетекстових елементів HTML-розмітки.

Впровадження

Застосування елемента STYLE — це основний спосіб впровадження каскадних таблиць стилів у HTML-документ. Крім керування відображенням елементів розмітки, елемент STYLE дозволяє описувати стильові властивості елементів, які можна змінювати при програмуванні на JavaScript.

Елемент STYLE дає можливість визначити стиль відображення для:

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

Стандартні елементи розмітки описуються в елементі STYLE наступним 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
елемента HTML-розмітки.
</P>
...
</BODY>
```

Тепер усі параграфи документа будуть відображатися стилем з елемента STYLE, якщо тільки стиль не буде яким-небудь способом перевизначений. У STYLE можна визначити стиль будь-якого елемента розмітки.

Зв’язування

Посилання на опис стилю, розташованого за межами документа, здійснюється за допомогою елемента LINK, що розміщають в елементі HEAD. Зовнішнім описом може бути файл, що містить опис стилів. Опис стилів у цьому файлі буде по синтаксису в точності збігатися зі змістом елемента STYLE.

Нижче наведений приклад посилання на зовнішній опис стилів:

<LINK TYPE="text/css" REL="stylesheet" REF="http://intuit.ru/my_css.css">

Тут важливі значення атрибутів REL і TYPE. Атрибут REL повинен мати значення stylesheet. Type може приймати значення text/css або text/javascript.

Атрибут HREF задає універсальний показчик ресурсу (URL) для зовнішнього файла опису стилів. Це може бути посилання на файл із будь-яким ім'ям, а не лише на файл із розширенням *.css.

Імпортування

Імпорт опису стилів у дечому нагадує вказівку на зовнішній опис стилю. Імпортувати стиль можна або всередину елемента STYLE, або всередину зовнішнього файлу, що являє собою опис стилю.

Оператор імпорта стилю повинен передувати всім іншим описам стилів:

```
<STYLE>
@import:url(http://intuit.ru/style.css)
A { color:cyan;text-decoration:underline; }
</STYLE>
```

Питання для самоконтролю

1. Для чого призначенні каскадні таблиці CSS.
2. Які ви знаєте способи застосування таблиць CSS?
3. Яким чином відбувається вбудовування CSS?
4. Яким чином відбувається зв’язування CSS?
5. Яким чином відбувається імпортування CSS?
6. Яким чином відбувається впровадження CSS?

Тема. Каскадні таблиці стилю CSS. Блокові і стрічкові елементи

План

1. Синтаксис CSS.
2. Блокові і стрічкові елементи.
3. Обтікання тексту.

1. Синтаксис CSS

Формально стиль відображення елементів розмітки задається посиланням в елементі розмітки на селектор стилю. Синтаксис опису стилів у загальному вигляді можна подати в такий спосіб:

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

або

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

У першому варіанті перераховані селектори, для яких діє даний опис стилю. Другий варіант задає ієрархію вкладеності селекторів, для сукупності яких визначений стиль. Нагадаємо, що мова в даному випадку йде про опис стилів у нотації text/css. Опис стилів розміщаються або всередині елемента STYLE, або в зовнішньому файлі.

При описі стилів вказується **селектор** - назва тега (чи список назв тегів) і у фігурних дужках опис стилю (пари **властивість:значення** записують через крапку з комою) .

Як селектор можна використовувати ім'я елемента розмітки, ім'я класу й ідентифікатор об'єкта на HTML-сторінці.

Атрибут (attribute) визначає властивість відображуваного елемента, наприклад лівий відступ параграфа (margin-left), а значення (value) — значення цього атрибута, наприклад, 10 типографських пунктів (10 pt).

Селектор — ім'я елемента розмітки

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

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

Зовнішній файл при цьому може виглядати наступним чином:

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

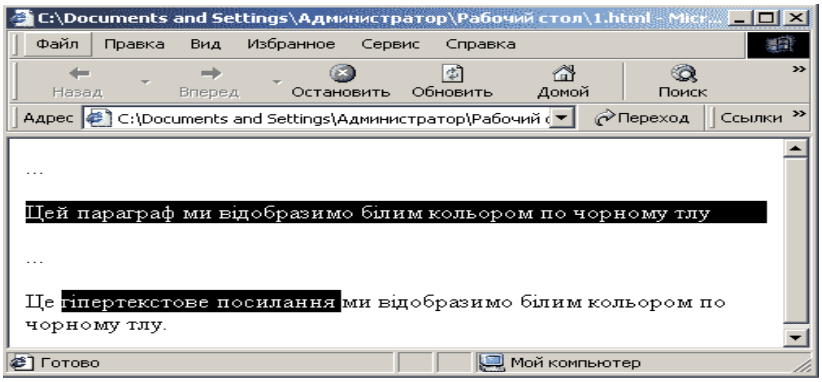
У першому рядку цього опису перераховані селектори-елементи, що будуть відображатися однаково:

Селектор — ім'я класу

Ім'я класу не є яким-небудь стандартним ім'ям елемента HTML-розмітки. Воно визначає опис класу елементів розмітки, що будуть відображатися однаково. Для того, щоб віднести елемент розмітки до того або іншого класу, потрібно скористатися його атрибутом CLASS:

```
<STYLE>
.test {color:white;background-color:black;}
</STYLE>
```

...
<P CLASS="test">
Цей параграф ми відобразимо білим кольором по чорному тлу
</P>
...
<P>
Це <A CLASS="test"> гіпертекстове посилання </A>
ми відобразимо білим кольором по чорному тлу.
</P>



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

Початкову крапку в імені класу можна опустити. Вона задається з розумінь 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 даному прикладі клас гіпертекстових посилань menu має один опис стилю, а клас гіпертекстових посилань paragraph — зовсім інше. При цьому кожен з цих класів не можна застосувати до інших елементів розмітки, наприклад, параграфові або спискові. Якщо ім'я елемента розмітки не задано, це означає, що клас можна застосувати до будь-якого елемента розмітки.

Селектор — ідентифікатор об'єкта

Об'єктна модель документа (Document Object Model) описує документ як дерево об'єктів. Об'єктами є: сам документ, його розділи (елемент DIV), зображення, параграфи, додатки і т.п. Кожному з цих об'єктів можна дати ім’я і звертатися до нього по імені. Дана можливість використовується при програмуванні сторінок на стороні клієнта.

Застосування ідентифікатора об'єкта виправдано ще й у випадку модифікації атрибута опису стилю для даного об'єкта в його CSS-описі. Замість двох описів класів, що відрізняються лише одним з параметрів, можна створити один опис класу й опис ідентифікатора об'єкта. Опис стилю для об'єкта задається рядком, у якому селектор являє собою ім'я цього об'єкта з передуючим символом "#":

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

2. Блокові і стрічкові елементи

В описі елементів розмітки мови HTML існує поняття стрічкового (in-line) елемента розмітки і блокового (block) елемента розмітки. Формально вони визначені в DTD SGML-описі мови HTML. Пояснити відмінності між блоковим і стрічковими елементами можна на прикладі:

- параграф — це блоковий елемент розмітки;
- виділення курсивом — це стрічковий елемент розмітки.

Блокові елементи можна вкладати один в одного, але вони не повинні перетинатися. Стрічкові елементи можна як вкладати, так і перетинати, але останнє робити не рекомендується.

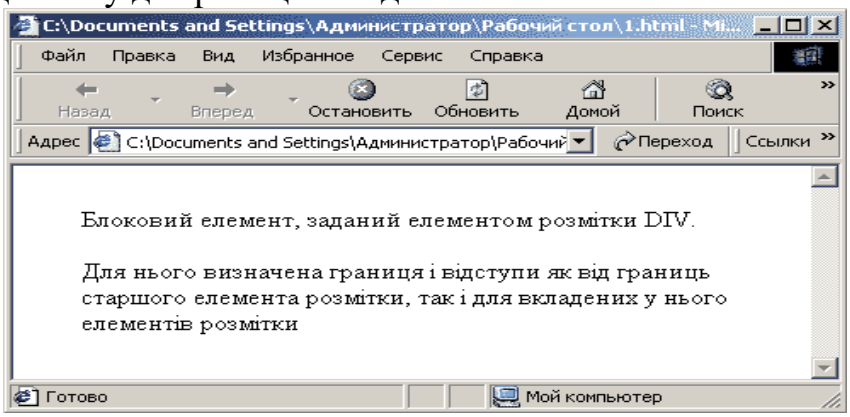
Очевидно, що по наборі атрибутів керування відображенням (атрибути опису стилю) стрічкові і блокові елементи відрізняються. Спрощено можна сказати, що атрибути опису стилю стрічкового елемента є підмножиною атрибутів опису стилю блокового елемента.

Узагальненнями блокового і стрічкового елементів, з точки зору стилів, є елементи DIV і SPAN, відповідно.

Елемент DIV

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

DIV дозволяє застосувати атрибути стилю, пов'язані з границею блоку і відступами блоку від границь старшого елемента, а також "набивка", тобто, відступ від границі блоку до границі вкладеного елемента:




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

У даному прикладі всередині вікна браузера розташований блоковий елемент (DIV), всередину якого поміщений ще один блоковий елемент (P). DIV має біле тіло і границю.

Елемент SPAN

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

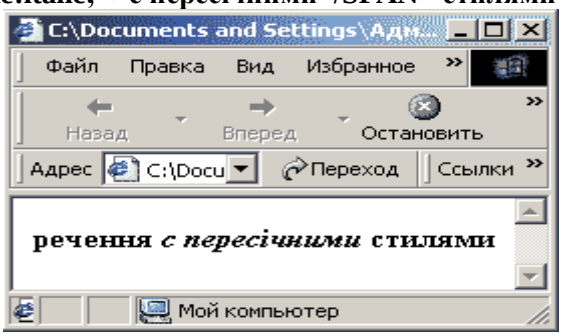
Таблиця 1

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>

Тег кінця елемента стрічкового типу закриває найближчий елемент, а не той, котрий відкритий тегом початку даного стрічкового стилю. Також і у випадку застосування елемента SPAN, де тег кінця можна співвіднести лише з найближчим тегом початку елемента SPAN:

<SPAN STYLE="font-weight:bold;">речення

<SPAN STYLE="font-style:italic;">с пересічними</SPAN> стилями</SPAN>



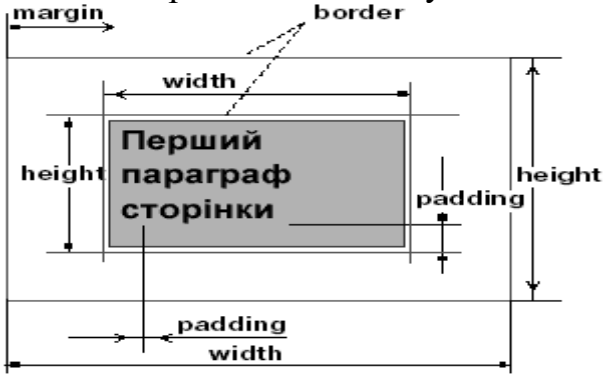
Застосування елемента SPAN обмежено браузерами, що підтримують CSS. При цьому не всі атрибути специфікації CSS підтримуються в браузерах.

Властивості блоків

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

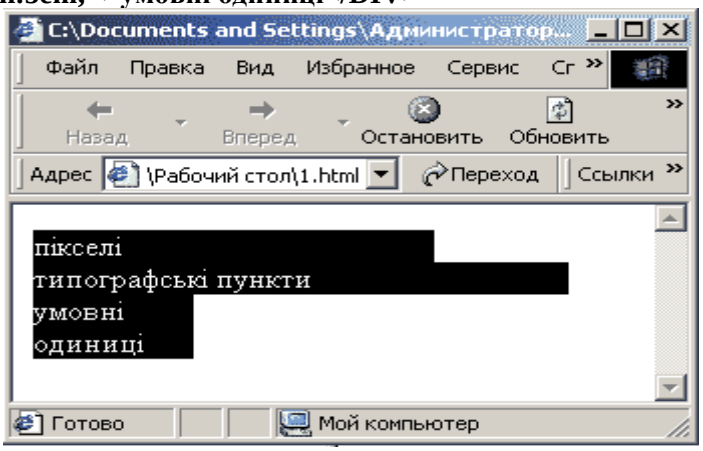
Блок тексту має такі властивості: висоту (height), ширину (width), границі (border), відступ (margin), набивка (padding), довільне розміщення (float), керування обтіканням (clear).

Графічно властивості можна представити наступним чином:



Із шириною і висотою блоку тексту усе більш-менш зрозуміло. Задаватися вони можуть у типографських пунктах (pt), пікселях (px) і умовних одиницях (em):

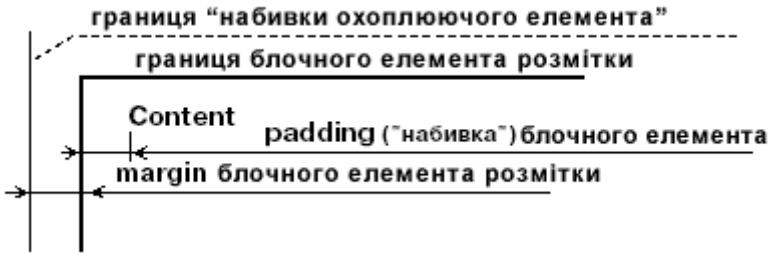
```
<DIV STYLE="width:200px;">пікселі</DIV>
<DIV STYLE="width:200pt;">типографські пункти</DIV>
<DIV STYLE="width:5em;">умовні одиниці</DIV>
```



Відстань від границі блокового елемента до границі вкладеного в нього блокового елемента називається padding. У рамках даного курсу лекцій для позначення цієї властивості використовується слово «набивка» або словосполучення «внутрішній відступ».

Відступ від «набивки» зовнішнього блокового елемента до границі вкладеного елемента називається margin. Для його позначення ми будемо вживати термін «відступ» або словосполучення «зовнішній відступ».

У такий спосіб padding і margin характеризують відступи блокового елемента відносного початку його змісту і щодо границі його елемента, що охоплює, розмітки, відповідно (див рис.).



Відступи і «набивки» можуть бути лівими, правими, верхніми і нижніми. CSS дозволяє змінювати будь-які з них.

При відображенні блоку тексту можна показати його видиму границю. CSS дозволяє визначити її стиль, ширину і колір. При використанні видимої границі варто враховувати специфіку браузерів. Одним з можливих способів застосування границі є видиме обмеження «плаваючих» блоків тексту.

«Плаваючий» текстовий блок дозволяє реалізувати можливість обтікання цього блоку текстом.

Притиснемо блок тексту вправо. Ліворуч його буде обтікати інший текст.

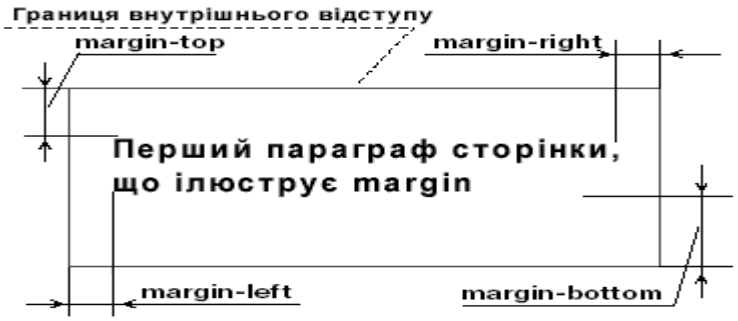
Обтікання одного тексту іншим відбувається аналогічно, що й обтікання текстом картинки або таблиці. Текст охоплюючого блоку, прагне «втиснутися» на вільне місце, залишене «плаваючим» блоком. Коли границя «плаваючого» блоку

закінчується, охоплюючий блок поширюється на всю ширину відведеного для тексту простору.

Таким чином, блок тексту з точки зору розміщення на сторінці рівноцінний картинкам або прямокутним областям додатків.

Відступи (margin)

При відображенні блоку тексту на папері довкола нього зазвичай залишають поля. Поля можна задавати або щодо границі сторінки, або щодо самого блоку тексту. У першому випадку ми маємо справу з «набивкою» (padding), а в другому — з відступом (margin). Власне, ширина поля буде визначатися сумою ширини «набивки» і ширини відступу.



Звичайно пунктирна лінія і границя блоку є невидимими лініями. Вони є уявними по вирівняному краю тексту. Вірніше, є уявною сумарна ширина полів. Стрілки вказують напрямок відліку відступу. Padding відраховується від зовнішньої границі блоку всередину блоку, у той час як margin — від зовнішньої границі блоку в область охопленого ним блоку (назовні).

Зовнішній відступ (margin) може відраховуватися за будь-яким напрямком щодо сторін блоку:

- **margin-left** — лівий зовнішній відступ. Визначає відстань від лівої границі блоку тексту до лівої границі внутрішнього відступу («набивки», padding) охопленого елемента;
- **margin-right** — правий зовнішній відступ. Визначає відстань від правої границі блоку тексту до правої границі внутрішнього відступу («набивки», padding) охопленого елемента;
- **margin-top** — верхній зовнішній відступ. Визначає відстань від верхньої границі блоку тексту до верхньої границі внутрішнього відступу («набивки», padding) охопленого елемента;
- **margin-bottom** — нижній зовнішній відступ. Визначає відстань від нижньої границі блоку тексту до нижньої границі внутрішнього відступу («набивки», padding) охопленого елемента;
- **margin** — задає загальний зовнішній відступ від усіх сторін блоку тексту. Застосовується в тому випадку, якщо блок тексту равновіддалений від усіх границь внутрішнього відступу охопленого елемента.

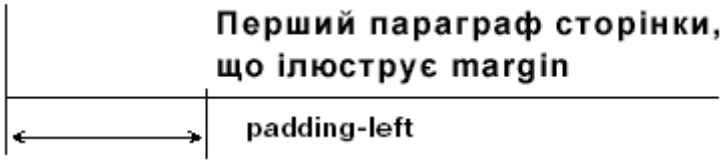
Якщо розмір усіх зовнішніх відступів однаковий, то можна просто скористатися атрибутом margin:

P { margin:5px; }

При застосуванні зовнішнього відступу варто пам'ятати, що він відраховується від границі елемента до границі внутрішнього відступу («набивки», padding) охопленого елемента. Якщо цей факт не враховувати, то загальна ширина видимих полів може виявитися більшою, ніж зазначено в зовнішньому відступі.

Набивка (padding)

Текст всередині блоку починається не від самої його границі. Між границею і змістом блоку є вільний простір. Він називається внутрішній відступ текстового блоку або padding. Разом із зовнішнім відступом (margin) текстового блоку padding утворить загальне поле відступу від границі охопленого блокового елемента розмітки.



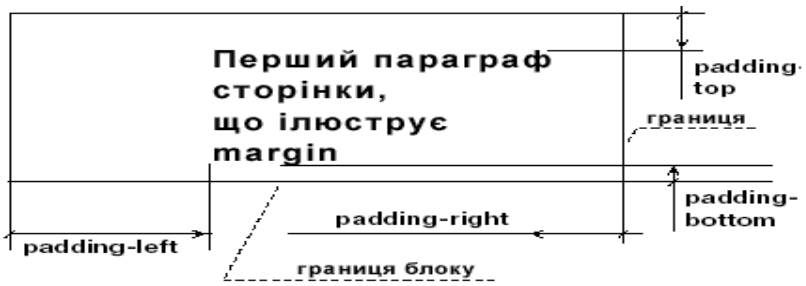
Для цього прикладу при описі параграфа використовувався стиль:

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

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

- **padding-left —** лівий внутрішній відступ, що визначає відстань від лівого краю блоку до його змісту;
- **padding-right** — правий внутрішній відступ, що визначає відстань від правого краю блоку до його змісту;
- **padding-top** — верхній внутрішній відступ, що визначає відстань від верхнього краю блоку до його змісту;
- **padding-bottom** — нижній внутрішній відступ, що визначає відстань від нижнього краю блоку до його змісту;
- **padding** — визначає єдиний розмір внутрішнього відступу блоку. Цей параметр задається у випадку однакового розміру відступу від усіх сторін блоку.

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



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

Границя (border)

У кожного блокового елемента розмітки є границя. Від границі відраховуються відступи (margin і padding). Вздовж границі «плаваючого» блоку його обтікає текст.

Для опису границь блоків застосовуються наступні атрибути:

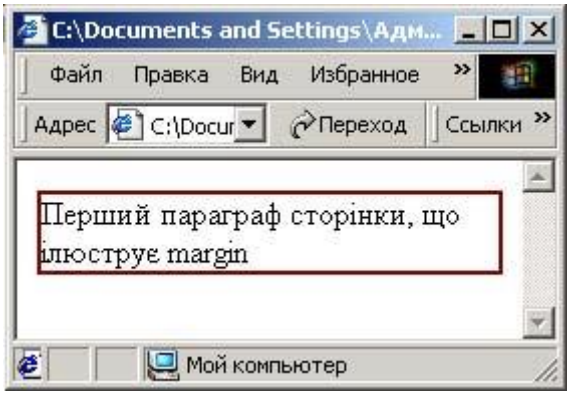
- **border-top-width** — ширина верхньої границі блоку;
- **border-bottom-width** — ширина нижньої границі блоку;
- **border-left-width** — ширина лівої границі блоку;
- **border-right-width** — ширина правої границі блоку;
- **border-width** — ширина границі блоку. Задається в тому випадку, якщо ширина границі блоку однакова по всьому периметрі блоку;
- **border-color** — колір границі блоку;
- **border-style** — тип лінії границі блоку. Може приймати значення: none, dotted, dashed, solid, double, groove, ridge, inset, outset.

Для опису границі немає необхідності вказувати в стилі всі атрибути. Існує скорочений запис атрибутів. Наприклад, для опису верхньої лінії границі можна використовувати запис типу:

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

атрибут: ширина_лінії тип_лінії колір_лінії код

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



У цьому прикладі ми використовували наступний опис стилю відображення границі:

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

3. Обтікання тексту

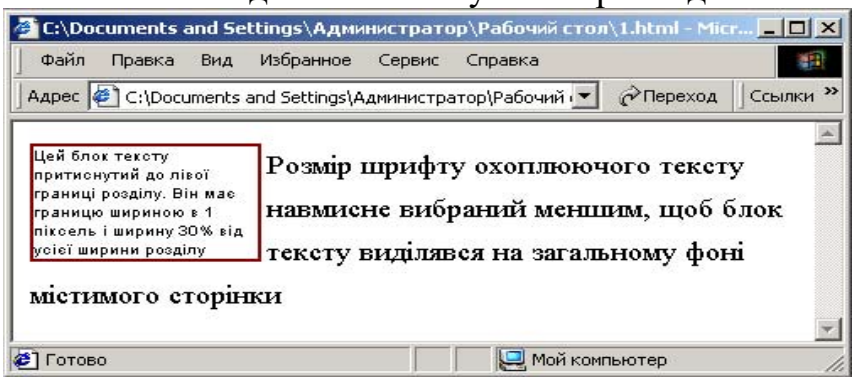
Під обтіканням блоку текстом розуміють той самий ефект, який можна досягнути для графіки, коли картинка не розриває блок тексту, а вбудовується в нього. Текст у цьому випадку «обтікає» картинку з одного боку — там, де є вільне поле між границею сторінки (елемента) і картинкою. Обтікання картинки текстом від звичайного вбудовування картинки в текст документа відрізняється тим, що вздовж вертикальної границі картинки розташовується кілька рядків тексту, а не одна.

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

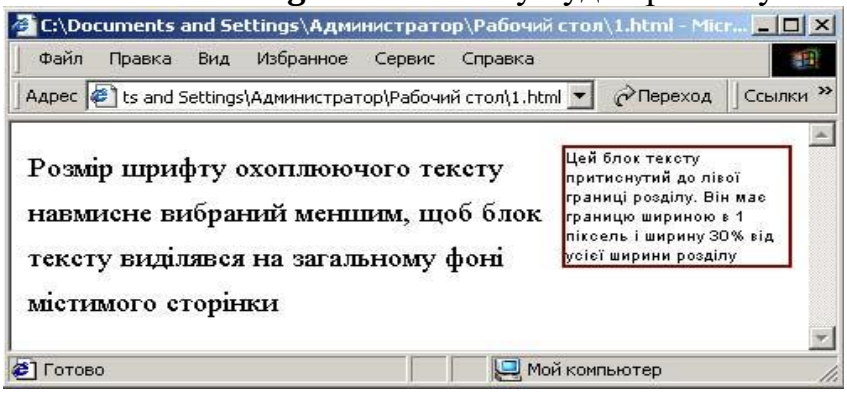
Атрибут **float** визначає «плаваючий» блок тексту. Він може приймати значення:

- **left** — блок притиснутий до лівої границі охоплюючого елемента;
- **right** — блок притиснутий до правої границі охоплюючого елемента;
- **both** — текст може обтікати блок по обидва боки.

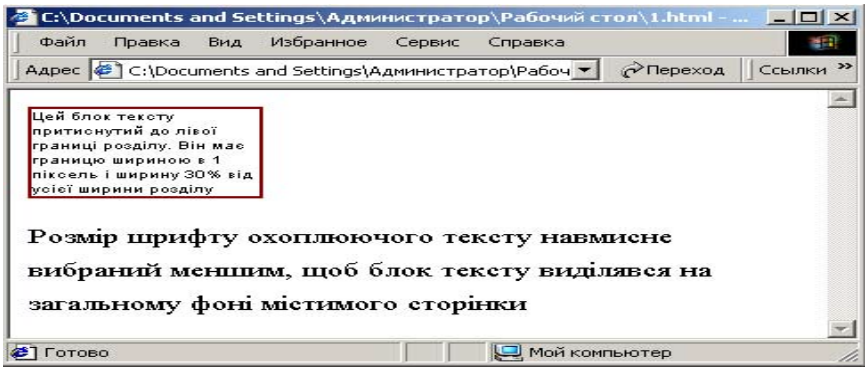
Проілюструвати обтікання дозволяє наступний приклад:



При використанні значення **right** блок тексту буде притиснутий вправо.



Другий атрибут опису стилів **clear** дозволяє керувати власне обтіканням. Він не допускає наявності «плаваючих» блоків біля блоку тексту. Атрибут може приймати значення: **right, left, none, both**:



Питання для самоконтролю

1. Для чого призначенні каскадні таблиці CSS?
2. Охарактеризуйте синтаксис таблиць CSS.
3. Що таке блокові елементи?
4. Що таке стрічкові елементи?
5. Назвіть властивості блоків?
6. Які Ви знаєте способи обтікання тексту?

<u>Тема.</u> Керування кольором і визначення шрифту у CSS

План

1. Керування кольором у CSS.
2. Керування шрифтами у CSS.
3. Характеристики тексту у CSS:
4. Списки у CSS.

1. Керування кольором у CSS

Каскадні таблиці стилів (CSS), у першу чергу, описують властивості тексту. Це стосується як текстових блоків, так і стрічкових елементів розмітки змісту сторінки. У даному розділі мова йтиме про керування відображенням кольору тексту (color) і кольору тіла (background-color), на якому відображається текст.

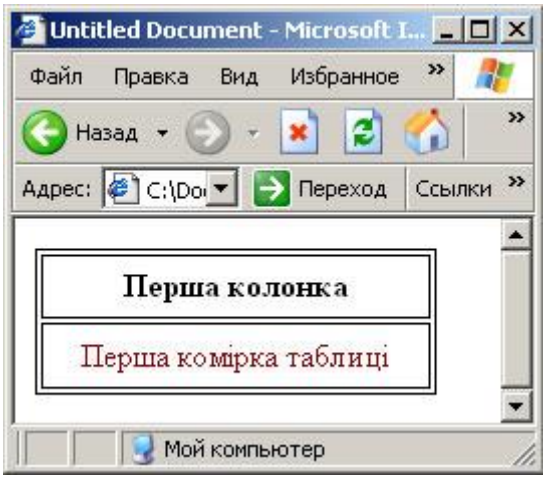
Крім кольору тексту і кольору тіла CSS дозволяє визначати колір границі текстового блоку (border-color).

Колір тексту

У HTML для керування кольором відображуваного тексту використовується елемент FONT. Його аналогом у CSS є атрибут color. Цей атрибут можна застосовувати як для блокових, так і для стрічкових елементів розмітки.

Розглянемо як блоковий елемент розмітки комірку таблиці:

TD{color:darkred;}



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

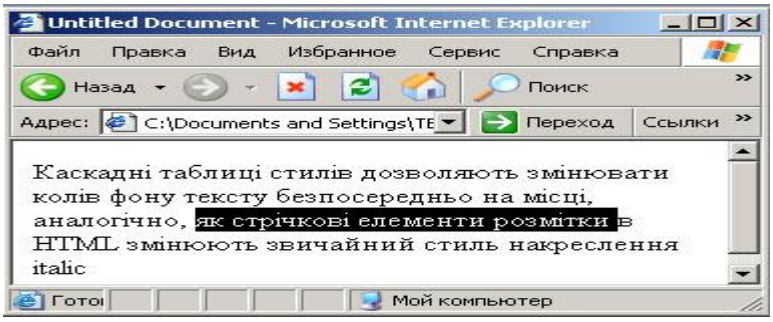
Колір тіла тексту

У HTML кольором тла можна керувати тільки для конкретного блокового елемента розмітки. Таким елементом може бути вся сторінка:

<BODY BGCOLOR=...>...</BODY>
Або, наприклад, таблиця:
<TABLE BGCOLOR=...>...</TABLE>

У наведеному нижче прикладі для виділення тексту застосоване інвертування кольору тіла і кольору тексту:

<SPAN STYLE=«background-color:black;color:white;»>
як стрічкові елементи розмітки
</SPAN>



2. Керування шрифтами у CSS

Шрифтам у комп'ютерній графіці завжди приділялося багато уваги, і World Wide Web не є виключенням. Але все багатство і розмаїтість існуючих шрифтів для української мови обмежено фактично трьома шрифтами: serif (звичайно Times або інший шрифт із засічками), sans-serif (Arial, Helvetica або інший шрифт без засічок) і monospace (Courier). Якщо бути точним, то тут перераховані сімейства шрифтів. Звичайно, кожне з цих сімейств представлено лише одним кириличним шрифтом.

Автор документу для керування відображенням літер може застосувати кілька атрибутів, що впливають на шрифт:

- **font-family** - сімейство накреслень шрифту (гарнітура);
- **font-style** - пряме накреслення або курсив;
- **font-weight** - «посилення» (насиченість) шрифту, «жирність» букв;

- **font-size** - розмір шрифту (кегль). Задається в пікселях (px) і типографських пунктах (pt);
- **font-variant** - варіант накреслення (звичайний або дрібні букви - капітель).

При використанні різних гарнітур (font-family) варто 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 в користувача шрифти.

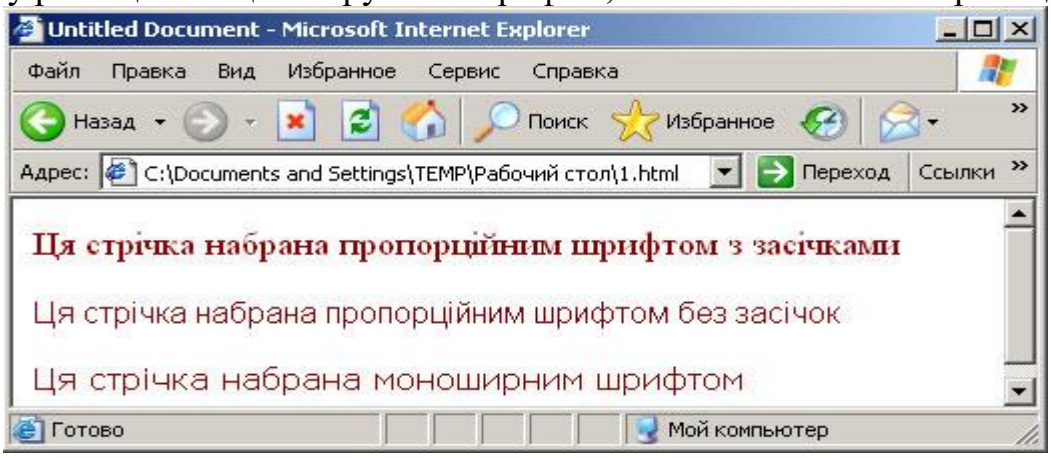
Саме неприємне, з чим можна зіштовхнутися при використанні шрифтів - це невідповідність моноширних шрифтів, що застосовуються в HTML-формах. Зворотний зв'язок з користувачем у цьому випадку неможливий.

Гарнітура (font-family)

Гарнітура шрифту - це набір накреслень одного шрифту. Шрифт може мати «пряме» накреслення (normal), курсив (italic), «скошене» (oblique), посилене по насиченості («жирне», bold), «дрібне» (капітель, small-caps) і т.п.

Найбільш розповсюджені гарнітури в російській і українській частинах Web - це Times, Arial, Courier. Причому усі вони належать до різних груп шрифтів. Times - це пропорційний шрифт «із засічками» (serif), Arial - це пропорційний шрифт «без засічок» (sans-serif), а Courier - це моноширний шрифт (monospace). У Unix замість Arial частіше застосовується Helvetica.

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



Кегль (font-size)

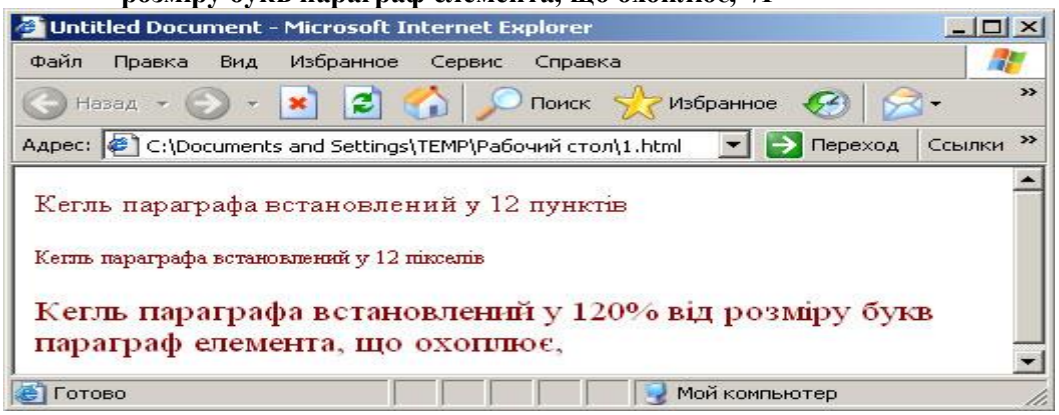
Кегль - це, якщо говорити спрощено, розмір шрифту. Більш докладне пояснення варто шукати в спеціальній літературі. Нам досить знати, що CSS через параметр font-size дозволяє керувати розміром літер.

Розмір шрифту можна задавати в типографських пунктах (pt, 0,35 мм) або пікселях (px). При установці кегля варто пам'ятати, що font-size задає не висоту літери, а розмір «прямокутника» під букву, що більше самої букви.

От кілька прикладів використання font-size:

<P STYLE=«font-size:12pt;»>Кегль параграфа встановлений у 12 пунктів</P>
<P STYLE=«font-size:12px;»>Кегль параграфа встановлений у 12 пікселів</P>
<P STYLE=«font-size:120%;»>Кегль параграфа встановлений у 120% від

розміру букв параграф елемента, що охоплює,</P>



Як видно з останнього прикладу, кегль можна задавати не тільки в абсолютних одиницях, але й у відносних. Крім відсотків, існує ще кілька умовних одиниць виміру 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 кегля x-small</P>
<P STYLE=«font-size:xx-small;»>Розмір кегля xx-small</P>

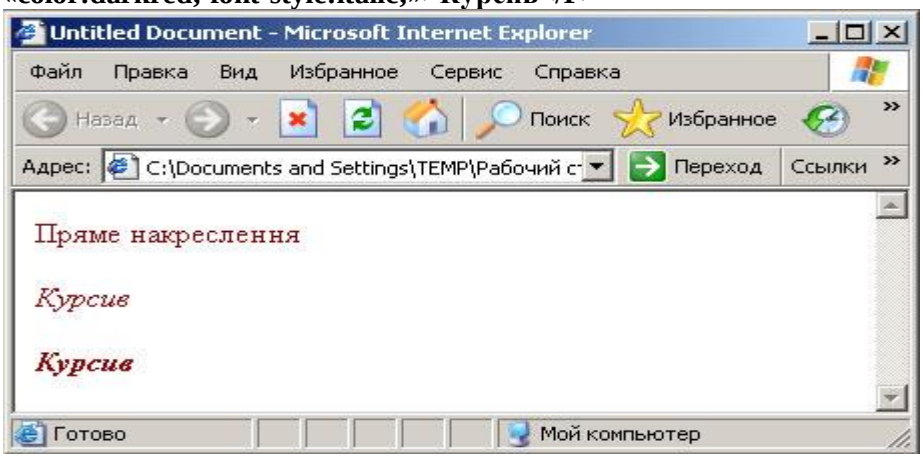
Аналогічно x-small і xx-small, існують розміри x-large і xx-large. Крім того, є larger, smaller і medium.

Накреслення

У кожної гарнітури (font-family) існує кілька накреслень. Кожне з них визначається в CSS трьома параметрами стилю: font-style, font-variant, font-weight.

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

<P STYLE=«color:darkred; font-style:normal;»>Пряме накреслення</P>
<P STYLE=«color:darkred; font-style:italic;»>Курсив</P>



Якщо хочеться підсилити насиченість («жирність») шрифту, то в описі стилю вказують атрибут font-weight, що приймає значення normal або bold:

<P STYLE=«color:darkred;font-style:italic;font-weight:bold;»>Курсив</P>

3. Характеристика тексту CSS

У цьому розділі ми розглянемо ті властивості текстового фрагменту, що залишилися без уваги в розділах, присвячених блокам тексту і шрифтам.

Під час обговорення властивостей блокових елементів розмітки мова йшла про параметри, що відносяться до блоку як цілого. Ми не розглядали внутрішні характеристики тексту.

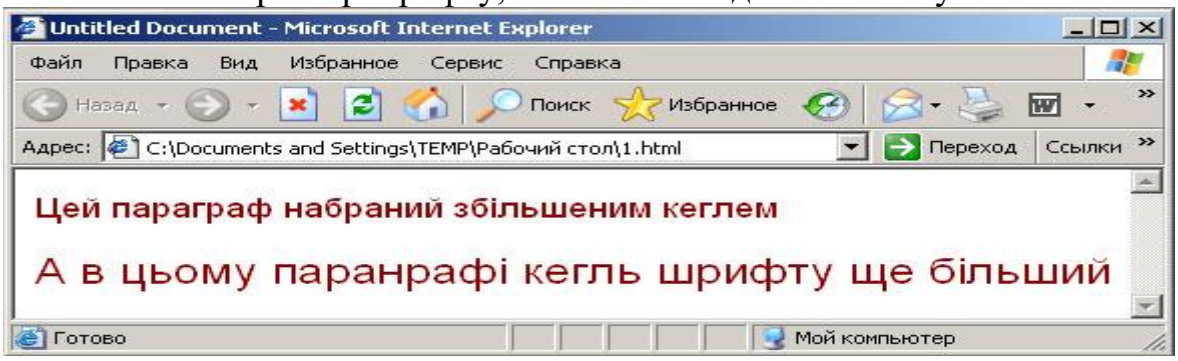
Розповідаючи про шрифти, ми акцентували увагу на накресленнях символів як таких, а не на їхньому співвідношенні.

Проте, осторонь залишилися такі важливі характеристики текстового фрагменту, як:

- міжбуквенні відстані;
- висота рядків;
- вирівнювання;
- відступ у першому рядку параграфа;
- перетворення накреслення.

Міжбуквенні відстані

Відстань між буквами автоматично регулюється розміром шрифту — кеглем. Чим більший розмір шрифту, тим більша відстань між буквами:

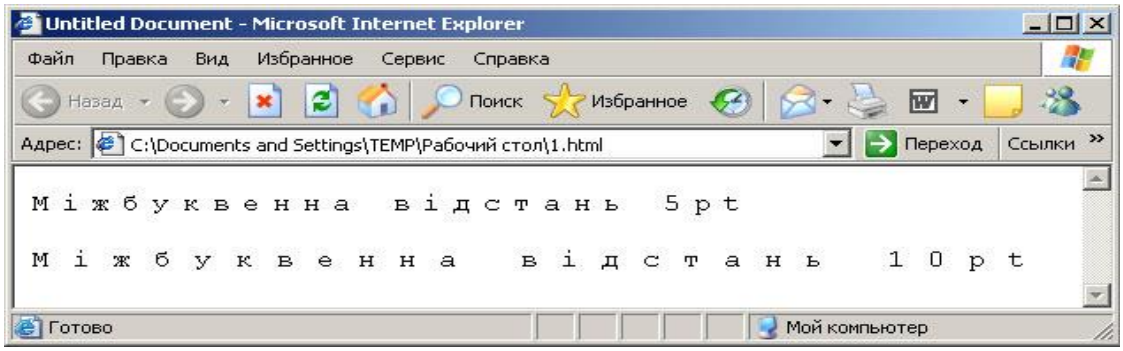


Придивившись уважно, неважко переконатися, що відстань між буквами в слові «параграф» першого прикладу і буквами слова «параграфа» другого прикладу різна. В другому випадку вона більша:

Моноширний шрифт обраний не випадково. На пропорційному шрифті міжбуквенна відстань залежить від накреслення букв і визначити її як відстань між буквами досить складно. У моноширного шрифту розмір символу фіксований, тому і відстань між буквами простежується чітко.

Однак, не завжди зручно керувати міжбуквенною відстанню через кегль (font-size). Бувають випадки, коли потрібно або ущільнити рядок, або збільшити відстані між буквами. Це можна зробити за допомогою атрибута ***letter-spacing***:

<P STYLE=«font-family:monospace;letter-spacing:5pt;color:black»>
Міжбуквенна відстань 5pt</P>
<P STYLE=«font-family:monospace;letter-spacing:10pt;color:black»>
Міжбуквенна відстань 10pt</P>



Вирівнювання

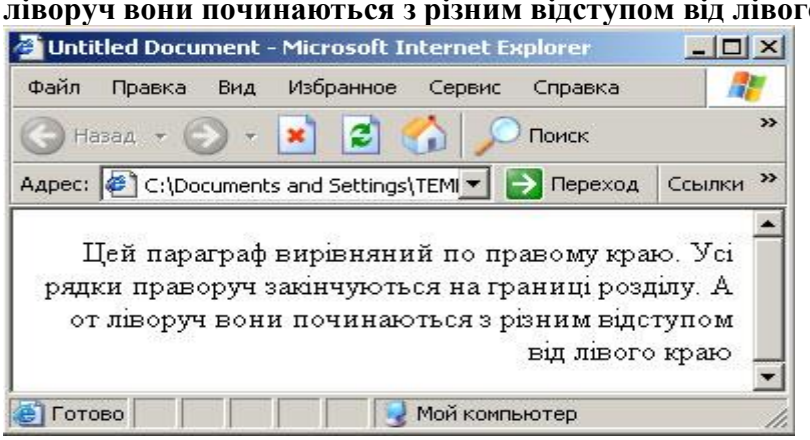
По замовчуванню усі слова в параграфі притиснуті вліво. Лівий край параграфа в такий спосіб виявляється вирівняним. Точно так само може бути вирівняний правий край параграфа або блоку тексту, і навіть обидва краї разом.

У звичайній HTML-розмітці такий ефект досягається за рахунок застосування атрибута ALIGN, як це зроблено на сторінках даного посібника:

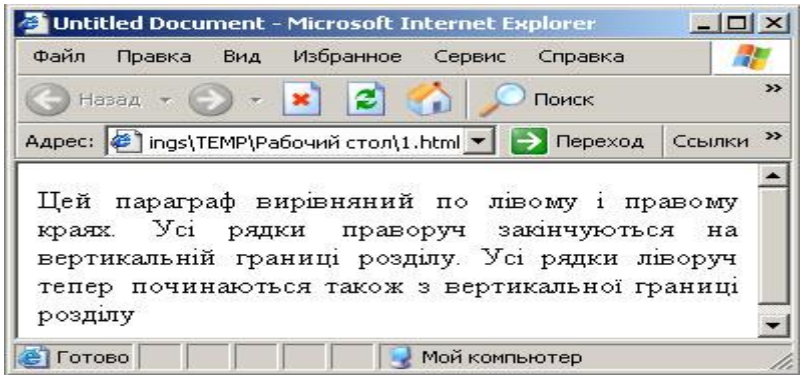
<P ALIGN=justify>...</P>

Аналогічний результат у CSS досягається за рахунок атрибута ***text-align***:

<P STYLE=«text-align:right; color:black;»>
Цей параграф вирівняний по правому краю. Усі рядки праворуч закінчуються на границі розділу. А от ліворуч вони починаються з різним відступом від лівого краю</P>



<P STYLE=«text-align:justify; color:black;»>
Цей параграф вирівняний по лівому і правому краях.
Усі рядки праворуч закінчуються на вертикальній границі розділу.
Усі рядки ліворуч тепер починаються також з вертикальної границі розділу</P>



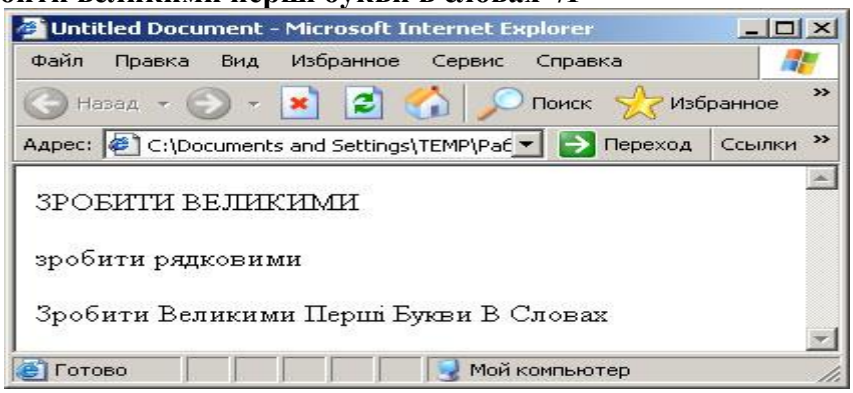
Вирівнювати текст можна в будь-якому блоковому елементі. Причому можна не лише вирівнювати текст по краях блокового елемента, але і центрувати його (<P STYLE=«text-align:center;»>...</P>).

Перетворення шрифту

Перетворення шрифту має на увазі капіталізацію слів, заміну всіх «великих» і «маленьких» літер у великі, або, навпаки, одержання одних рядкових літер.

Розглянемо кілька прикладів:

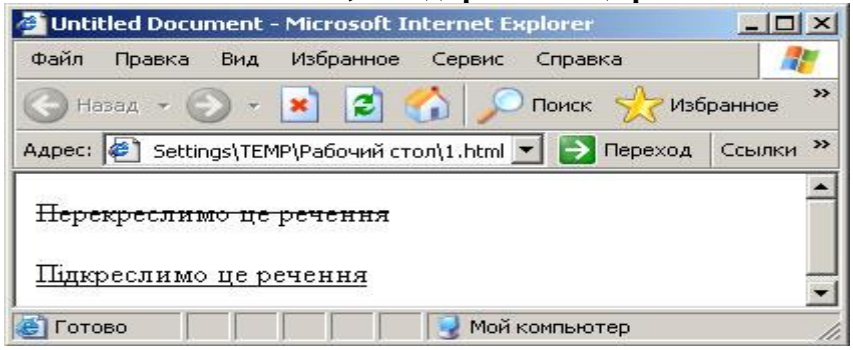
<P STYLE=«text-transform:uppercase;»>Зробити великими</P>
<P STYLE=«text-transform:lowercase;»>Зробити рядковими</P>
<P STYLE=«text-transform:capitalize;»>
Зробити великими перші букви в словах</P>



Зверніть увагу, що виконання перетворень залежить від алгоритму перетворення символів.

Ще один вид перетворення шрифту — це *підкреслення*, *перекреслення* або *надкреслення* слів. Виконується таке перетворення за допомогою атрибуту ***text-decoration:***

<P STYLE=«text-decoration:line-through;»>Перекреслимо це речення</P>
<P STYLE=«text-decoration:underline;»>Підкреслимо це речення</P>



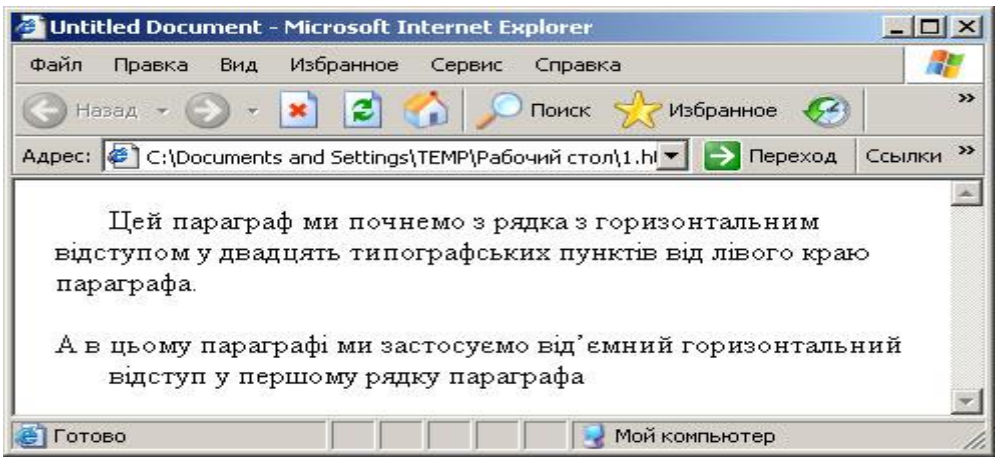
Для того, щоб перетворення працювало, необхідно відповідне накреслення (підкреслені або перекреслені накреслення букв). Дуже складно знайти гарнітури, у яких було б накреслення з надкресленими буквами. Скасування декору відбувається, якщо використовувати в text-decoration значення none.

Перший рядок параграфа

При оформленні параграфів у технології CSS автор може скористатися абзацним відступом, таку можливість надає йому атрибут ***text-indent***.

Мова йде про горизонтальний відступ у першому рядку параграфа щодо його лівого краю:

<P STYLE=«text-indent:20pt;»>Цей параграф ми почнемо з рядка з горизонтальним відступом у двадцять типографських пунктів від лівого краю параграфа.</P>
<P STYLE=«text-indent:-10pt;»>А в цьому параграфі ми застосуємо від'ємний горизонтальний відступ у першому рядку параграфа</P>



Крім text-indent у CSS для оформлення першого рядка параграфа зарезервований модифікатор стилю ***first-line***. Він дозволяє не лише задати горизонтальний зсув, але і визначити інші параметри параграфа:

P:first-line { color:red; }

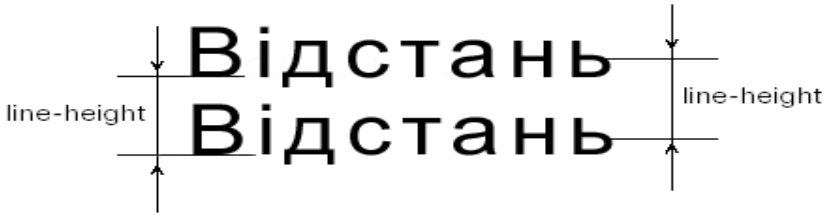
Ще один параметр, що впливає на відображення першого рядка параграфа — перша буква першого рядка. Її відображенням керує модифікатор first-letter:

P:first-letter { font-size:20pt; }

На жаль, обидва названих модифікатора реалізовані не у всіх версіях браузерів, тому для вірності застосовують елементи розмітки FONT і TABLE.

Міжстрічкова відстань

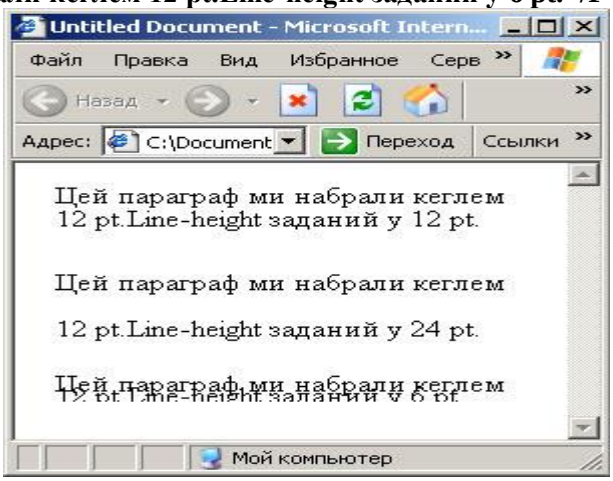
У CSS міжстрічкова відстань визначається параметром line-height. Він задає відстань не між рядками, а між базовими лініями рядків. Простіше кажучи, якщо, наприклад, взяти букву «н» і надрукувати її послідовно одну під одною, то line-height буде дорівнює відстані між двома однаковими точками букв.



Подивимося, як цей параметр впливає на взаємне розташування рядків:

<P STYLE=«line-height:12pt;font-size:12pt;color:black;»>

Цей параграф ми набрали кеглем 12 pt.Line-height заданий у 12 pt.</p>
<P STYLE=«line-height:24pt;font-size:12pt;color:black;»>
Цей параграф ми набрали кеглем 12 pt.Line-height заданий у 24 pt.</P>
<P STYLE=«line-height:6pt;font-size:12pt;color:black;»>
Цей параграф ми набрали кеглем 12 pt.Line-height заданий у 6 pt.</P>



Перший приклад набраний зі значенням line-height, рівним розмірові кегля. В другому прикладі значення line-height удвічі перевищує розмір кегля. У третьому прикладі значення line-height у два рази менше розміру кегля — рядка стали «наповзати» один на одного.

4. Списки у CSS

При відображенні списків CSS дозволяє керувати формою і зображенням «кульок» (bullets) списку. «Кулька» (bullet) — це символ, що розміщенний перед елементом списку. Наприклад, у невпорядкованому списку (unordered list) перед елементом списку ставиться «жирна» крапка:

- Перший елемент списку;
- Другий елемент списку;
- Третій елемент списку.

CSS дозволяє керувати формою «кульок» і заміняти «кульки» картинками.

Цікаво, що керування відображенням елементів списку віднесено до набору властивостей, у які входить атрибут display. У цього атрибута може бути лише одне значення — none. Якщо елемент у своєму описі має атрибут display, і цей атрибут дорівнює none, то він не відображається браузером взагалі:

```
<UL STYLE=«display:none;»>
        <LI>Перший елемент списку
        <LI>Другий елемент списку
        <LI>Третій елемент списку
</UL>
```

Атрибут display керує відображенням документа на дисплеї комп'ютера, але не поширюється на інші середовища відображення документа. Наприклад, при друці схований список повинен бути відображений.

Однак, насправді він не відображається і при друці.

Форма «кульок»

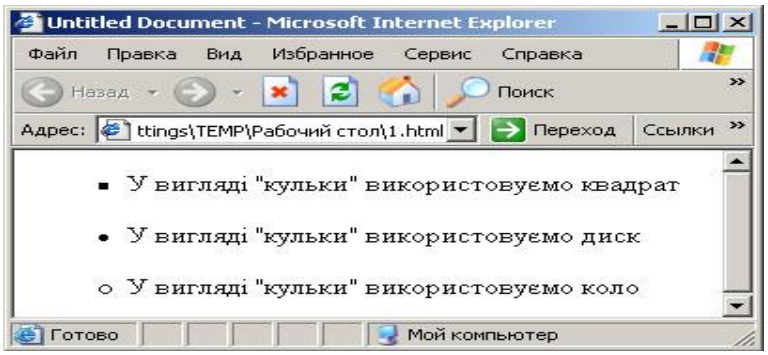
Форма «кульки» у вигляді «жирної» крапки трохи незвична. Звичайно в машинописних документах використовують риску. З іншого боку, у рекламних матеріалах часто в якості «кульки» застосовують квадрат або інший символ типографського набору, а також графічну картинку.

CSS дозволяє керувати формою «кульки» через атрибут ***list-style-type***:

```
<UL STYLE=«list-style-type:square;»>
        <LI>У вигляді «кульки» використовуємо квадрат
</UL>

<UL STYLE=«list-style-type:disk;»>
        <LI>У вигляді «кульки» використовуємо диск
</UL>

<UL STYLE=«list-style-type:circle;»>
        <LI>У вигляді «кульки» використовуємо коло
</UL>
```

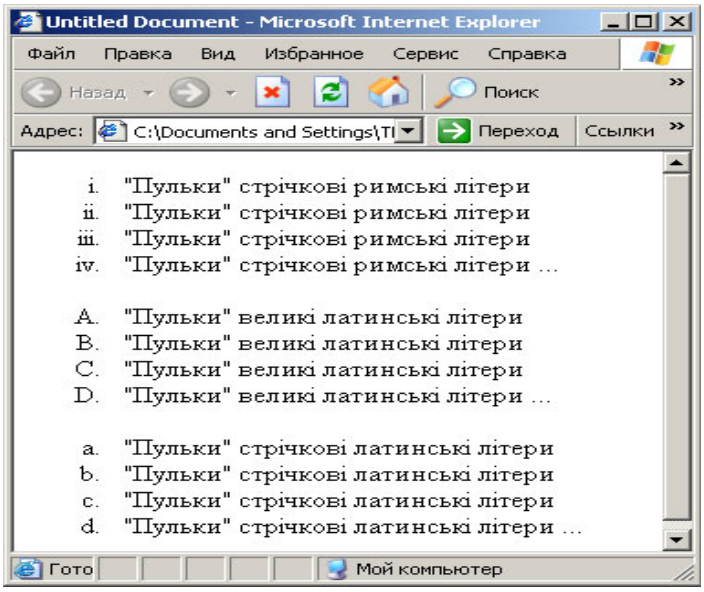


Дотепер ми говорили лише про невпорядковані списки (UL), але керувати відображенням «кульок» можна й в упорядкованих списках (OL):

```
<OL STYLE=«list-style-type:lower-roman; color:black;»>
        <LI>...
                ...
</OL>

<OL STYLE=«list-style-type:upper-alpha; color:black;»>
        <LI>...
                ...
</OL>

<OL STYLE=«list-style-type:lower-alpha; color:black;»>
        <LI>...
        ...
</OL>
```



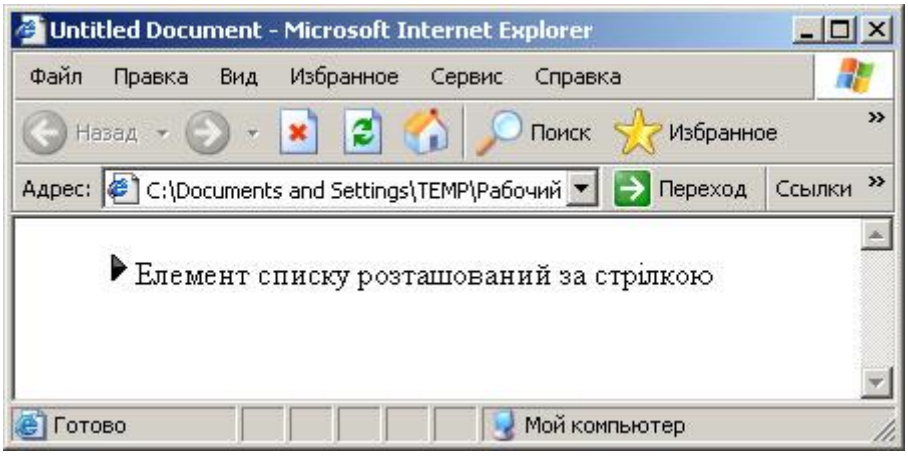
«Кульки»-картинки

Якщо стандартні форми «кульок» автора сторінки не влаштовують, він може використовувати нестандартні. Для цього йому доведеться «кульку» намалювати самому й у вигляді графічного файлу розмістити на Web-вузлі. У такої «кульки» є URL, що використовується в CSS для звертання до неї.

```
<UL STYLE=«list-style-image:url(bimage.gif);>
        <LI>Елемент списку розташований за рисою
</UL>
```

Картинка може бути і більш цікавішою. Це вже залежить від фантазії автора документа. Наприклад, можна створити картинку-стрелочку:

```
<UL STYLE=«list-style-image:url(barrow.gif);»>
        <LI>Елемент списку розташований за стрілкою
</UL>
```



Питання для самоконтролю

1. Як задати колір тексту у CSS?
2. Як задати колір документа у таблицях CSS?
3. Які Ви знаєте атрибути керування шрифтами у CSS?
4. Наведіть кілька прикладів використання шрифтів.
5. Як характеризується текстовий фрагмент у CSS.
6. Як задати міжбуквені відстані у тексті?

7. Як задати висоту рядків у тексті?
8. Як задається вирівнювання у тексті?
9. Яким чином відбувається перетворення рядків у тексті.
10. Яким чином задаються списки у CSS?

Тема. Основні положення мови сценаріїв Java Script

План

1. Основні теоретичні відомості.
2. Засоби введення та виведення даних.
3. Типи даних.

1. Основні теоретичні відомості

Мова програмування JavaScript призначена для створення інтерактивних HTML-документів. Основні області використання JavaScript такі:

- створення динамічних сторінок, тобто, сторінок, вміст яких може мінятися після завантаження документа;
- перевірка правильності заповнення користувачем форм до пересилання їх на сервер;
- розв'язання «локальних» задач за допомогою сценаріїв і деякі інші сфери.

JavaScript дозволяє створювати додатки, виконувані як на стороні клієнта, так і на стороні сервера. При розробці додатків обох типів використовується так називане ядро, у якому містяться визначення стандартних об'єктів. Клієнтські додатки виконуються браузером на машині користувача.

Програма (сценарій) мовою JavaScript обробляється вбудованим у браузер інтерпретатором. Треба прагнути до того, щоб написані сценарії коректно виконувалися в будь-якому браузері. На первісному етапі навчання домогтися задоволення цієї вимоги складно.

Програма (сценарій) мовою JavaScript являє собою послідовність операторів. Якщо кілька операторів розташовуються в одному рядку, то між ними потрібно поставити знак «крапка з комою» (;). Якщо кожен оператор розміщається в окремому рядку, то роздільник можна не писати. Один оператор може розташовуватися на декількох рядках.

Відповідно до принципів структурного програмування, програму рекомендується записувати таким чином, щоб у ній була відбита блокова структура. Це полегшує дослідження програми і пошук помилок.

Де писати програми і як їх запускати

Приклади, які наводяться, потрібно самостійно виконати на комп'ютері. Це можна зробити по-різному, але принаймні на першому етапі рекомендую скористатися найпростішим і доступним способом: в якості інтерпретатора (виконавчої системи) програм на JavaScript візьміть веб-браузер. Як редактор ваших програм виберіть який-небудь простий текстовий редактор. Відкрийте

текстовий редактор і створіть у ньому заготовку файлу, що ви будете потім редагувати, вводячи експериментальні вирази або навіть цілі програми. А саме введіть у робоче поле редактора наступний текст:

```
<HTML>
  <HEAD><TITLE> Приклади </TITLE></HEAD>
    <SCRIPT>
    </SCRIPT>
</HTHL>
```

Вирази мовою JavaScript потрібно записувати між тегами **<SCRIPT>** і **</SCRIPT>.** Домовимось записувати в кожному рядку не більш одного виразу мовою JavaScript. Закінчувати рядок натисканням на клавішу Enter, щоб перейти до нового рядка. Нижче приведений приклад програми:

```
<HTML>
<HEAD> <TITLE> Приклади </TITLE> </HEAD>
  <SCRIPT>
    x  = 5
    y = x + 3
    alert(y)
  </SCRIPT>
</HTML>
```

При виконанні навчальних прикладів звичайно потрібно вивести на екран остаточні або проміжні дані. Для цього можна використовувати метод **alert**, вказавши в круглих дужках те, що потрібно вивести на екран. У приведеному вище прикладі метод alert(y) виведе на екран діалогове вікно, у якому відобразить значення змінної y (у даному випадку — число 8). Якщо не вставляти рядок alert(y), то програма виконується без відображення результату. Якщо ви хочете виводити результати обчислень не в діалоговому вікні, а безпосередньо у вікні веб-броузера, то замість alert(y) напишіть наступний вираз:

```
Document.writeln(y)
```

Створювати програми на JavaScript можна і за допомогою програм, спеціально призначених для розробки веб-сайтів, — наприклад Microsoft FrontPage або Macromedia Dreamweaver.

2. Введення і виведення даних

У JavaScript передбачені досить слабкі засоби для введення і виведення даних. Це цілком виправдано, оскільки JavaScript створювалась в першу чергу як мова сценаріїв для веб-сторінок. Якщо ви пишете програму на JavaScript, що буде виконуватися веб-броузером Internet Explorer, то можете скористатися трьома стандартними методами для введення і виведення даних: alert, prompt і confirm.

1. alert

Даний метод дозволяє виводити діалогове вікно з заданим повідомленням і кнопкою ОК. Синтаксис відповідного виразу має такий вигляд:

```
alert(повідомлення)
```

Якщо ваше повідомлення конкретне, тобто являє собою цілком визначений набір символів, те його необхідно написати в подвійних або одинарних лапках. Наприклад, alert(«Привіт усім!») (рис. 1). Узагалі говорячи, повідомлення являє собою дані будь-якого типу: послідовність символів, укладену в лапкb, число (у лапках або без них), змінну або вираз.

Рис. 1. Вікно діалогу, створене методом alert («Привіт усім!»)

Діалогове вікно, виведене на екран методом alert(), можна забрати, клацнувши на кнопці ОК. Доти поки ви не зробите цього, перехід до раніше відкритих вікон неможливий. Вікна, що мають властивість зупиняти всі наступні дії користувача і програм, називаються модальними. Таким чином, вікно, створюване за допомогою alert(), є модальним. Метод alert() можна використовувати для виведення проміжних і остаточних результатів програм при їхньому налагодженні.

2. confirm

Метод confirm() дозволяє вивести діалогове вікно з повідомленням і двома кнопками — ОК і Скасування (Cancel). На відміну від методу alert() цей метод повертає логічну величину, значення якої залежить від того, на яку із двох кнопок клацнув користувач. Якщо він клацнув на кнопці ОК, то повертається значення true (істина, так); якщо ж він клацнув на кнопці Скасування, то повертається значення false (хиба, немає). Значення, що повертається, можна обробити в програмі і, отже, створити ефект інтерактивності, тобто діалогової взаємодії програми з користувачем. Вікно, створюване за допомогою confirm(), є модальним. Синтаксис застосування методу confirm() має такий вигляд:

confirm (повідомлення)

Якщо ваше повідомлення конкретне, тобто являє собою цілком визначений набір символів, то його необхідно записати в лапках, подвійних або одинарних. Наприклад, confirm(«Ви дійсно хочете завершити роботу?») (рис. 2).

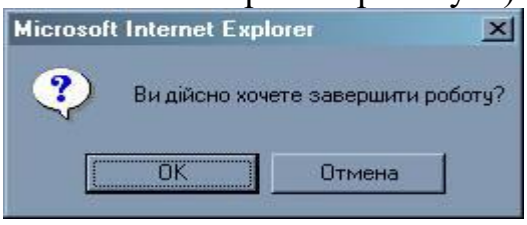


Рис. 2. Вікно діалогу створене методом confirm()

3. prompt

Метод prompt дозволяє вивести на екран діалогове вікно з повідомленням, а також з текстовим полем, у яке користувач може ввести дані (рис. 3). Крім того, у цьому вікні передбачені дві кнопки: ОК и Скасування (Cancel). На відміну від методів alert() і confirm() даний метод приймає два параметри: повідомлення і значення, що повинне з'явитися в текстовому полі введення даних за замовчуванням. Якщо користувач клацне на кнопці ОК, то метод поверне вміст поля введення даних, а якщо він клацне на кнопці Отмена, то повертається логічне значення false. Значення, що повертається, можна потім обробити в програмі і, отже, створити ефект інтерактивності. Синтаксис застосування методу prompt має такий вигляд:

prompt (повідомлення, значення_поля_введення_даних)

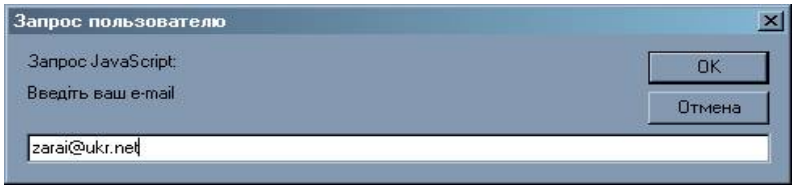


Рис. 3 Вікно діалогу створене методом Prompt()

Параметри методу prompt() не є обов'язковими. Якщо ви їх не вкажете, то буде виведене вікно без повідомлення, а в поле введення даних підставлене значення за замовчуванням — undefined (не визначене). Якщо ви не хочете, щоб у поле введення даних з'являлося значення за замовчуванням, то підставте як значення другого параметра порожній рядок «».

3. Типи даних

У будь-якій мові програмування дуже важливе поняття типу даних. Якщо не усвідомити його із самого початку, то потім прийдеться часто зіштовхуватися з дивним поводженням створеної вами програми (Табл. 1.1.).

Таблиця 1.1. Типи даних у JavaScript

Тип даних	Приклади	Описи значень
Рядковий або символьний (string)	«Привіт» д. т. 23-45-67»	Послідовність символів, записана в лапках, подвійних або одинарних
Числовий (number)	3.14 -567 +2.5	Число, послідовність цифр, перед якою може бути зазначений знак числа (+ або -); перед додатними числами не обов'язково ставити знак +; ціла і дробова частини чисел розділяються крапкою. Число записується без лапок
Логічний (булевий, boolean)	True false	Можливі тільки два значення true або false
Null		Відсутність якого б то не було значення
Об’єкт (object)		Програмний об'єкт, обумовлений своїми властивостями. Зокрема, масив також є об'єктом
Функція (function)		Визначення функції — програмного коду, виконання якого може повертати деяке значення

При створенні програм на JavaScript за типами даних стежить сам програміст. Якщо він переплутає типи, то інтерпретатор не зафіксує помилки, а спробує привести дані до деякого типу, щоб виконати зазначену операцію. Вам належить розібратися, до якого саме типу приводиться суміш даних різного типу. Багато мов програмування, у тому числі C і Pascal, не мають цю властивість, вони вимагають явної вказівки типу даних.

Функція **parseInt**(рядок, основа) перетворить зазначену в параметрі рядок у ціле число в системі числення по зазначеній основі (8, 10 або 16). Якщо основа не зазначена, то передбачається 10, тобто, десяткова система числення.

Приклади:

parseInt(«3.14») // результат = 3
parseInt(«-7.875») // результат = -7
parseInt(«435») // результат = 435
parseInt(«Вася») // результат = NaN, тобто не є числом
parseInt(«15»,8) // результат = 13
parseInt(«0xFF»,16) // результат = 255

Зверніть увагу, що при перетворенні в ціле число округлення не відбувається: дробова частина просто відкидається.

Функція **parseFloa**t(рядок) перетворить зазначений рядок у число з плаваючою точкою.

Приклади:

parseFloat(«3.14») // результат = 3.14
parseFloat(«-7.875») // результат = -7.875
parseFloart («435») // результат = 435
parseFloat («Вася») // результат = NaN, тобто не є числом

Задача перетворення чисел у рядки виникає рідше, ніж зворотне перетворення. Щоб перетворити число в рядок, досить до порожнього рядка додати це число, тобто, скористатися оператором додавання «+». Наприклад, обчислення вираз ««+3.14 дасть у результаті рядок «3.14».

Для визначення того, чи є значення виразу числом, служить убудована функція isNaNN (значення). Обчислення цієї функції дає результат логічного типу. Якщо зазначене значення не є числом, функція повертає true, інакше — false. Однак тут поняття «число» не збігається з поняттям «значення числового типу». Функція isNaNN() вважає числом і дані числового типу, і рядок, що містить тільки число. Логічні значення також ідентифікуються як числа. При цьому значенню true відповідає 1, а значенню false — 0. Таким чином, якщо isNaNN() повертає false, те це означає, що значення параметра має числовий тип, або є числом, перетвореним у рядковий тип, або є логічним (true або false).

І нарешті розглянемо так звані службові символи, які можна вставляти в рядки

Символ	Опис
\n	Новий рядок
\t	Табуляція
\f	Нова сторінка
\b	Забій
\r	Повернення каретки

Ці символи звичайно використовуються при формуванні рядкових даних для їхнього наступного відображення. Наприклад, якщо ми хочемо, щоб повідомлення, виведене на екран у браузері Internet Explorer за допомогою функції alert(), відображалося у виді декількох рядків, то варто використовувати службовий символ \n (рис. 4).

alert(«Прізвище – Іванів \nІм’я – Іван\nПо батькові – Іванович «)

Рис. 4. Вікно, створене функцією alert(), де рядок повідомлення містить два символи \n

Іноді потрібно відобразити символи, що мають службове призначення. Як, наприклад, відобразити лапки, якщо вони використовуються для завдання рядка символів? Для цієї мети використовується «\» (зворотна коса риска). Наприклад,

щоб відобразити рядок Акціонерне товариство «Роги та копита» разом з лапками, потрібно написати такий рядок:

«Акціонерне товариство \»Роги та копита\» «.

Зворотна коса риска вказує, що наступний безпосередньо за нею символ не потрібно інтерпретувати як символ синтаксису мови. У нашому прикладі вона показує, що лапки не є ознакою початку або закінчення рядкових даних, а є просто елементом цих даних. У Internet Explorer виконання виразу:

alert(«Акціонерне товариство \»Роги та копита\»«) дасть результат, показаний на рис. 5.

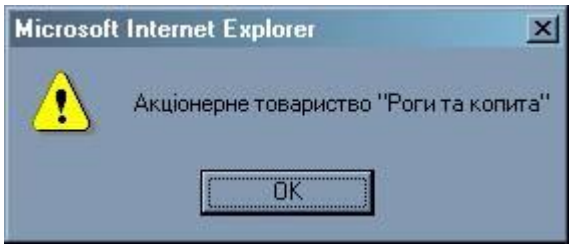


Рис. 5.

Відмітимо, що цю же задачу можна вирішити і трохи інакше, використовуючи лапки різних видів (подвійні й одинарні). По-перше, можна написати так: 'Акціонерне товариство «Роги та копита» '. У цьому випадку ми використовували одинарних лапки як ознаки початку і кінця всього рядка. По-друге, можна поміняти місцями лапки різних видів: «Акціонерне товариство 'Роги та копита' «. Однак, у цьому випадку назва акціонерного товариства буде відображатися в одинарних лапках. Нарешті, можна зовнішні лапки залишити подвійними, а внутрішні одинарні лапки продублювати: «Акціонерне товариство «Рогу і копита» «, Тоді при відображенні рядка внутрішні лапки будуть замінені на подвійні.

Зауваження: Лапки, що обрамляють рядкові дані, повинні бути одного виду і використовуватися парами. Інтерпретатор, знайшовши в тексті програми лапки, буде шукати ще лапки такого ж виду, вважаючи все, що знаходиться між ними рядковими даними.

Зауваження. Усередині рядка, написаного в лапках одного виду, можна використовувати лапки іншого виду (інакше інтерпретатор або видасть повідомлення про помилку, або неправильно сприйме дані).

Питання для самоконтролю

1. Охарактеризуйте мову програмування JavaScript.
2. Які Ви знаєте команди для введення даних?
3. Які Ви знаєте команди для виведення даних?
4. Охарактеризуйте типи даних.

Тема. Основні оператори мови сценаріїв Java Script

План

1. Змінні та оператор присвоєння.
2. Оператори:

2.1 арифметичні;
2.2 порівняння;
2.3 логічні.

1. Змінні та оператор присвоєння

Щоб зберігати дані у пам'яті й у той же час залишати їх доступними для подальшого використання, у програмах використовуються змінні.

Імена змінних

Змінну можна вважати контейнером для збереження даних. Дані, що зберігаються в змінній, називають значеннями цієї перемінної. Змінна має ім'я — послідовність букв, цифр і символу підкреслення без пробілів і розділових знаків, що починається обов'язково з букви або символу підкреслення.

При виборі імен змінних не можна використовувати ключові слова, тобто, слова, які визначають конструкції мови. Наприклад, не можна вибирати слова var, if, else, const, true, false, function, super, switch і ряд інших. Ім'я повинне відбивати зміст змінної. Якщо ім'я складається з декількох слів, то між ними можна вводити символ підкреслення або писати їх разом, починаючи кожне слово з великої букви. От приклади; my_first_adress, myFirstAdress. Іноді як перший символ імені використовують букву, що вказує на тип даних (значень) цієї змінної: с — рядковий (character), n - числовий (number), b — логічний (boolean), o — об'єкт (object), a — масив (array). Наприклад cAdress, nCena, aMonth. JavaScript є регістрозалежною мовою. Це означає, що зміна регістра символів (із прописних на рядкові і навпаки) в імені перемінної приводить до іншої змінної. Наприклад: variable, Variable і vaRiabLe – різні змінні.

Створення змінних

Створити змінну в програмі можна декількома способами:

1. За допомогою оператора присвоєння значень у форматі:

ім'я_змінної = значення

Оператор присвоєння позначається символом рівності **“=”.**

Приклад:

myName = «Іван»

2. За допомогою ключового слова **var** (від variable — змінна) у форматі:

var ім'я_змінної

У цьому випадку створена змінна не має ніякого значення, але може його одержати надалі за допомогою оператора присвоєння.

Приклад:

var myName

myName =«Іван»

3. За допомогою ключового слова var і оператора присвоєння у форматі:

var ім'я_змінної = значення

Приклад

var myName = «Іван»

Рядок коду програми з ключовим словом var задає так названу ініціалізацію змінної і для кожної змінної використовується один раз. Тип змінної визначається типом значення, що вона має. На відміну від багатьох інших мов програмування, при ініціалізації змінної не потрібно описувати її тип. Змінна може мати значення

різних типів і неодноразово їх змінювати. Наприклад, ви можете написати наступний код програми:

var x = «Іван»
// деякий код
x = «Ганна»
// деякий код
x = 2.5

Одне ключове слово var можна використовувати для ініціалізації відразу декількох змінних, як з оператором присвоєння, так і без нього. При цьому змінні і вирази з операторами присвоєння розділяються комами, наприклад:

var name = «Вася», address, x = 3,14

Існують додаткові оператори присвоєння, такі як **+=, -=** і т.п.

Оператор	**Приклад**	**Еквівалентний вираз**
+=	X+=Y	X=X + Y
-=	X-=Y	X=X-Y
*=	X*=Y	X=X*Y
/=	X/=Y	X=X/Y
%=	X%=Y	X=X%Y

Область дії змінних

Змінні, котрі створені в програмі за допомогою оператора присвоєння з використанням ключового слова var або без нього, є глобальними. Це означає, що вони доступні усюди в цій же програмі, а також у програмах, які викликаються з інших файлів. Ці змінні також доступні усередині коду функцій. Крім глобальних існують і локальні змінні. Вони можуть бути створені усередині коду функцій. Ви можете визначити змінні з однаковими назвами і в зовнішній програмі, і в коді функції. У цьому випадку змінні, визначені в коді функції за допомогою ключового слова var, будуть локальними: зміни їх значень усередині функції ніяк не відіб'ються на змінних з такими ж іменами, визначених у зовнішній програмі. При цьому значення локальних змінних не доступні з зовнішньої програми. Нерідко область дії називають областю видимості. Змінна може бути видима або не видима усередині програмної одиниці (функції, підпрограми). Область видимості, доступності або дії змінної — еквівалентні терміни. Крім них ще використовують поняття часу життя змінної. Час життя змінних у JavaScript визначається інтервалом часу від завантаження до вивантаження програми з пам'яті комп'ютера. Так, якщо програма (сценарій) записані в HTML-коді веб-сторінки, то після його вивантаження весь сценарій разом з визначеними в ньому змінними припиняє активне існування.

2. Оператори

Оператори призначені для складання виразів. Оператор застосовується до одного або двох даних, що у цьому випадку називаються операндами. Наприклад, оператор додавання застосовується до двох операндів, а оператор логічного заперечення — до одного операнду. Елементарний вираз, що складається з операндів і оператора, обчислюється інтерпретатором і, отже, має деяке значення. У цьому випадку говорять, що оператор повертає значення. Наприклад, оператор

додавання, застосований до чисел 2 і 3, повертає значення 5. Оператор має тип, що збігається з типом значення, що повертається їм.

Один оператор ми вже розглянули. Це оператор присвоєння “=”. Однак варто відмітити, що існує ще п'ять різновидів оператора присвоєння, що сполучають у собі дії звичайного оператора «=« і операторів додавання, віднімання, множення, ділення і ділення модулю.

Коментарі

Почнемо з операторів коментарю. Вони дозволяють виділити фрагмент програми, що не виконується інтерпретатором, а служать лише для пояснень змісту програми. У JavaScript припустимі два види операторів коментарю:

• // — один рядок символів, розташований праворуч від цього оператора, вважається коментарем;

• /*...*/ — усе, що укладено між /* і */, вважається коментарем; за допомогою цього оператора можна виділити кілька рядків як коментар.

Арифметичні оператори

Арифметичні оператори, такі як додавання, множення і т.д., у JavaScript можуть застосовуватися до даних будь-яких типів. Що з цього виходить, ми розглянемо небагато пізніше. А зараз просто перелічимо їх.

Арифметичні оператори	Назва	Приклад
+	Додавання	X + Y
-	Віднімання	X-Y
*	Множення	X*Y
/	Ділення	X/Y
%	Ділення по модулю	X%Y
++	Збільшення на 1	X++
--	Зменшення на 1	Y--

Формально ніщо не заважає нам застосувати ці оператори до даних інших типів. У випадку рядкових даних оператор додавання дає в результаті рядок, отриманий шляхом приписування праворуч другого рядка до першого. Наприклад, вираз «Вася» + «Маша» повертає в результаті «ВасяМаша». Тому для рядків оператор додавання називається оператором склейки або конкатенації. Застосування інших арифметичних операторів до рядків дає в результаті NaN. — значення, що не є числом. У випадку, коли оператор додавання застосовується до рядка і числа, інтерпретатор переводить число у відповідний рядок і далі діє як оператор склейки двох рядків.

У випадку логічних даних інтерпретатор переводить логічні значення операндів у числові (true у 1, false у 0), виконує обчислення і повертає числовий результат. Те ж саме відбувається в тому випадку, коли один оператор логічний, а іншої — числовий.

Якщо один операнд рядкового типу, а іншої - логічного, то у випадку додавання інтерпретатор переводить обидва операнди в рядковий тип і повертає

рядок символів, а у випадку інших арифметичних операторів він переводить обидва операнди в числовий тип.

На перший погляд, застосування арифметичних операторів до різнотипних даних може видатися досить заплутаним. Тому раджу, особливо спочатку, керуватися наступними простими правилами:

- застосовуйте арифметичні оператори до даних того самого типу;
- оператор додавання стосовно до рядків діє як оператор їхній склейки (приписування, конкатенації);
- у випадку застосування арифметичних операторів до логічних даних інтерпретатор розглядає значення true і false як числа відповідно 1 і 0, і повертає результат у числовому виді.

Крім цих правил потрібно пам'ятати, що в JavaScript є засоби для перетворення типів даних, тобто для перетворення даних одного типу в дані іншого типу.

Оператори порівняння

У програмах часто доводиться перевіряти, чи виконуються які-небудь умови.

Умови, що перевіряються, формуються на основі операторів порівняння, таких як «рівне», «менше», «більше» і т.д. Результатом обчислення елементарного виразу, що містить оператор порівняння та операнди, є логічне значення, тобто true або false.

Оператор	Назви	Приклад
==	Дорівнює	X==Y
!=	Не дорівнює	X!=Y
>	Більше, ніж	X>Y
>=	Більше або дорівнює (ме менше)	X>=Y
<	Менше, ніж	X<Y
<=	Менше або дорівнює (не більше)	X<=Y

Увага! Оператор «дорівнює» записується за допомогою двох символів «=» без пробілів між ними.

Порівнювати можна числа, рядки і логічні значення. Порівняння чисел відбувається за правилами арифметики, а рядків — шляхом порівняння ASCII-кодів символів починаючи з лівого кінця рядків. Логічні значення порівнюються так само, як і числа 1 і 0 (true відповідає 1, а false — 0).

Відмітимо, що оператори порівняння можуть бути застосовані і до різнотипних даних. Якщо порівнюються рядок і число, то інтерпретатор приводить операнди до числового типу. Те ж саме відбувається при порівнянні логічних даних і числа. Якщо порівнюються логічні дані і рядок, то справа трохи складніша. У цьому випадку результат не залежить від вмісту рядка. Якщо вона містить число (але не цифри з буквами), тільки пробіли або є порожній, то операнди приводяться до числового типу. При цьому порожній рядок («») або

рядок, який містить тільки пробіли перетвориться в число 0. В інших випадках всі оператори порівняння, крім <!=>, будуть повертати false (оператор <!=> — протилежний результат, тобто true).

Логічні оператори

Логічні дані, які отримують за допомогою елементарних виразів, що містять оператори порівняння, можна поєднувати в більш складні вирази. Для цього використовуються логічні (булеві) оператори — І чи АБО, а також оператор заперечення НЕ.

Оператор	Назва	Приклад
!	Заперечення (НЕ)	!X
&&	І	X&&Y
||	АБО	X||Y

Оператор заперечення «!» застосовується до одного операнду, змінюючи його значення на протилежне: якщо X має значення true, те !X повертає значення false, і навпаки, якщо X має значення false, те !X повертає значення true.

Нижче в таблиці зазначено, які значення повертають оператори І та АБО при різних логічних значеннях двох операндів.

X	Y	X&&Y	X||Y
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

Оператори “&&” і “||” ще називають логічним множенням і логічним додаванням відповідно. Якщо згадати, що значенню true можна зіставити 1, а значенню false — 0, то неважко зрозуміти, як обчислюються значення елементарних виражень з логічними операторами. Потрібно тільки врахувати, що в алгебрі, логіки 1 + 1 = 1 (а не 2). Аналогічно операторові заперечення відповідає вирахування з одиниці числового еквівалента логічного значення операнда.

Складні логічні вирази, що складаються з декількох більш простих, з'єднаними операторами І та АБО, виконуються за так званим принципом короткої обробки. Справа в тім, що значення усього виразу часом можна визначити, обчисливши лише одне або трохи більш простих виразів, не обчислюючи інші. Наприклад, вираз x && y обчислюється зліва на право; якщо значення x виявилося рівним false, то значення y не обчислюється, оскільки і так відомо, що значення усього вираження дорівнює false. Аналогічно якщо у виразі x || y значення x дорівнює true, те значення в не обчислюється, оскільки вже ясно, що весь вираз дорівнює true. Дана обставина вимагає особливої уваги при тестуванні складних логічних виразів. Так, якщо яка-небудь складова частина виразу містить помилку, то вона може залишитися невиявленою, оскільки ця частина виразу просто не виконувалася при тестуванні.

Питання для самоконтролю

1. Як записуються імена зміних?
2. Які Ви знаєте способи створення зміних?
3. Для чого призначені оператори у програмі?
4. Як записутються коментарі у програмі?
5. Охарактеризуйте арифметичні оператори.
6. Охарактеризуйте оператори порівняння.
7. Охарактеризуйте логічні оператори.

Тема. Галуженні програми та циклічні програми мовою сценаріїв Java Script

План

1. Оператори умовного переходу:
   1.1 Оператор if
   1.2 Оператор switch
2. Оператори циклу:
   2.1 Оператор for
   2.2 Оператор while
   2.3 Оператор do-while

1. Оператори умовного переходу

Обчислювальний процес можна направити тим чи іншим шляхом в залежності від того, чи виконується деяка умова чи ні. Для цієї мети служать оператори умовного переходу ***if і switch.***

Оператор if

Оператор умовного переходу if дозволяє реалізувати структуру умовного виразу якщо..., то ..., інакше...

Синтаксис оператора if переходу наступний:

```
if (умова)
        { код, котрий виконується в разі виконання умови }
  else
        { код, котрий виконується, якщо умова не виконана }
```

У фігурних дужках розташовується блок коду — кілька виразів. Якщо в блоці використовується не більш одного виразу, то фігурні дужки можна не писати. Частина цієї конструкції, обумовлена ключовим словом else (інакше), необов'язкова. У цьому випадку залишається тільки частина, визначена ключовим словом if (якщо):

```
if (умова)
        { код, що працює, якщо умова виконана }
```

Конструкція оператора умовного переходу допускає вкладення інших операторів умовного переходу. Умова зазвичай являє собою вираз логічного типу.

Приклади

1. Виводиться діалогове вікно з тим або іншим повідомленням у залежності від значення змінної age (вік).

```
if (age<18)
        {alert(«Ви занадто молоді для перегляду цього сайта»)}
  else
        {alert(«Підтвердіть своє рішення заглянути на цей сайт»)}
```

2. Виводиться діалогове вікно з повідомленням, якщо тільки значення змінної пеклі менше 18.

```
if (age<18)
        {alert(«Ви занадто молоді для перегляду цього сайта»)}
```

Чи робити відступи при написанні операторів, де розташовувати фігурні дужки — справа смаку. Варто керуватися наочністю і ясністю структури, при якій легко перевірити правильність розміщення дужок.

Вище ми уже відзначали, що умова в операторі if звичайно є логічним виразом. Однак це може бути також і рядковий, і числовий вираз. У випадку рядкового виразу умова вважається виконаною, якщо його значенням є не порожній рядок. Нагадаємо, що порожній рядок не містить жодного символу, у тому числі і пробілу (рядок, що містить хоча б один пробіл, не порожній). У випадку числового виразу умова вважається виконаною, якщо його значенням є число, відмінне від нуля. У багатьох випадках ця багатозначність типу умови виявляється дуже зручною.

Допустимо, що змінна x містить дані, що ввів користувач, і нам потрібно перевірити, що він дійсно щось увів. У наступному прикладі ми перевіряємо, що значення змінної x не порожнє (не 0, не порожній рядок і не null). Якщо x порожнє, то виводиться відповідне повідомлення:

```
if (!x)
    {alert(«ви нічого не ввели»)}
```

Оператор switch

Для організації перевірки невеликої кількості умов цілком достатньо використовувати розглянутий вище оператор if. Однак у випадку декількох умов більш зручний і наочним виявляється оператор switch (перемикач). Він особливо зручний, якщо потрібно перевірити кілька умов, що не є взаємовиключними.

Синтаксис оператора ***switch*** *виглядає в такий спосіб:*

```
switch (вираз) {
  case варіант1:
    код
    [break]
  case варіант2:
    код
    [break]
...
  [default:
код]
}
```

Параметр вираз оператора switch може приймати рядкові, числові і логічні значення. Зрозуміло, що у випадку логічного виразу можливі тільки два варіанти. Ключові слова (оператори) break і default не є обов'язковими, про що свідчать

прямокутні дужки. Тут вони є елементами опису синтаксису, і при написанні операторів їхній указувати не потрібно. Якщо оператор break зазначений, то перевірка інших умов не здійснюється. Якщо зазначено оператор default, то наступний за ним код виконується, якщо значення виразу не відповідає жодному з варіантів. Якщо усі варіанти можливих значень передбачені в операторі switch, то оператор default можна не використовувати.

Оператор switch працює в такий спосіб. Спочатку обчислюється вираз, зазначений в круглих дужках відразу за ключовим словом switch. Отримане значення порівнюється з тим, що зазначене в першому варіанті. Якщо вони не збігаються, то код цього варіанта не виконується і відбувається перехід до наступного варіанта. Якщо ж значення збіглися, то виконується код, що відповідає цьому варіантові. При цьому, якщо не зазначений оператор break, то виконуються коди й інших варіанти, поки не зустрінеться оператор break. Це ж правило діє і для інших варіантів.

Приклад

```
x = 2
   switch (x) {
        case 1:
             alert(l)
        case 2:
             alert(2)
        case 3:
             alert(3)
     }
```

У приведеному прикладі спрацюють 2-й і 3-й варіанти. Якщо ми хочемо, щоб спрацював тільки один який-небудь варіант (тільки той, котрий відповідає значенню виразу), то потрібно використовувати оператор break.

Приклад

```
 x = 2
switch (x) {
     case 1:
        alert(l);
        break
     case 2:
        alert(2);
        break
     case 3:
        alert(3);
        break
}
```

У цьому прикладі спрацює тільки 2-й варіант.

2. Оператори циклу

Оператор циклу забезпечує багаторазове виконання блоку програмного коду доти, поки не виконається деяка умова, У JavaScript передбачені три, оператора циклу: for, while і do-while.

Оператор for

Оператор for також називають оператором з лічильником циклів, хоча в ньому зовсім не обов'язково використовувати лічильник.

Синтаксис цього оператора наступний:

```
for ( [початковий вираз] ; [умова] ; [вираз оновлення] )
    {
       код
     }
```

Тут квадратні дужки лише вказують на те, що вказані в них параметри не є обов'язковими.

Усе, що знаходиться в круглих дужках праворуч від ключового слова for, називається заголовком оператора циклу, а вміст фігурних дужок — його тілом.

Оператор циклу працює в такий спосіб. Спочатку виконується початковий вираз. Потім перевіряється умова. Якщо вона істинна, то виконується код, записаний у фігурних дужках. Після цього виконується вираз оновлення (третій параметр оператора for). У випадку неістинності умови оператор циклу припиняє роботу (при цьому код не виконується).

У наступному прикладі оператор циклу просто змінює значення свого лічильника, виконуючи 15 ітерацій:

```
for (i = 1; i <= 15;  i++) {}
```

Небагато модифікуємо цей код, щоб обчислити суму всіх цілих додатних чисел від1 до 15:

```
var s = 1
     for ( i = 1; i <= 15; i++) {
       s = s + 1
     }
```

Лічильник циклів може бути не тільки зростаючим, але й спадаючим.

Приклад:

```
<html>
     <head> <title> Приклад обчислення ступеня у числа x </title> </head>
       <script>
          var x=2; y=10; z=x
          for (i=2; i <=y; i++)
             {
                z=z*x
              }
          alert(z)
       </script>
   </html>
```

Для примусового (тобто не за умовою) виходу з циклу використовується оператор **break** (переривання). Якщо обчислювальний процес зустрічає цей оператор у тілі оператора циклу, то він відразу ж завершується без виконання наступних виразів коду в тілі і навіть виразі оновлення. Звичайно оператор break застосовується при перевірці деякої додаткової умови, виконання якої вимагає завершення циклу, незважаючи на те що умова в заголовку циклу ще не виконана.

Типова структура оператора циклу з використанням ***break*** *має такий вигляд*:

```
for ( [початковий вираз] ; [умова1] ; [вираз оновлення] }
     {
       код
```

```
            if (умова2) {
                код
                break
                    }
        код
    }
```

Для керування обчисленнями в операторі циклу можна також використовувати оператор **continue** (продовження). Так само, як і break, цей оператор застосовується в тілі оператора циклу разом з оператором умовного переходу. Однак, на відміну від break, оператор continue припиняє виконання наступного за ним коду, виконує вираз оновлення і повертає обчислювальний процес у початок оператора циклу, де відбувається перевірка умови, зазначеної в заголовку.

Типова структура оператора циклу з використанням ***continue*** *має такий вигляд:*

```
for ( [початковий вираз] ; [умова1] ; [вираз оновлення] )
  {
        код
        if (умова2) {
          код
          continue
          }
        код
   }
```

Оператор while

Оператор циклу while (доти, поки) має структуру більш просту, чим оператор for, і працює трохи інакше.

Синтаксис цього оператора наступний:

```
while ( умова )
    {
      код
     }
```

При виконанні цього оператора спочатку здійснюється перевірка умови, зазначеної в заголовку, тобто в круглих дужках праворуч від ключового слова while. Якщо вона виконується, то виконується код у тілі оператора циклу. У противному випадку код не виконується. При виконанні коду (завершенні першої ітерації) обчислювальний процес повертається до заголовка, де знову перевіряється умова і т.д.

Оператор do-while

Оператор do-while (роби доти, поки) являє собою конструкцію з двох операторів, використовуваних спільно.

Синтаксис цієї конструкції наступний:

```
do {
        код
  }
  while (умова)
```

На відміну від оператора while в операторі do-while код виконується хоча б один раз, незалежно від умови. Умова перевіряється після виконання коду, якщо вона істинна, то знову виконується код у тілі оператора do. У противному випадку робота оператора do-while завершується. В операторі while умова перевірялась, в першу чергу, до виконання коду в тілі. Якщо при першому звертанні до оператора while умова не виконується, то код не буде виконаний ніколи.

Питання для самоконтролю

1. Охарактеризуйте оператори умовного переходу.
2. Наведіть приклади задач.
3. Охарактеризуйте оператори циклу.
4. Наведіть приклади задач.

Тема. Технологія AJAX

План

1. AJAX – інноваційний підхід до розробки сайту.
2. Переваги та недоліки технологія AJAX.
3. Особливості технології AJAX.

1. AJAX – інноваційний підхід до розробки сайту

Використання AJAX (Asynchronous JavaScript and XML) у роботі сайту уже стало «фішкою», яка свідчить про його інноваційність і відповідність сучасним тенденціям. AJAX є величезним проривом у розвитку Інтернет-технологій, але водночас це етап їх розвитку, який давно назрівав, а тому був неминучим. AJAX – це підхід до веб-розробки, який дає можливість веб-сторінці довантажувати необхідну інформацію без перезавантаження самої сторінки.

AJAX почали використовувати у 2005 році. AJAX не є окремою технологією, а концепцією використання декількох технологій, що існували і раніше. До цих технологій належать javascript, HTML, CSS, PHP, DOM, dHTML. AJAX є одним з елементів концепції DHTML, використовує DHTML для динамічної зміни контенту, а XMLHttpRequest – для динамічних звернень до сервера (без перезавантаження сторінки).

2. Переваги та недоліки технологія AJAX

Переваги AJAX очевидні:

- підвищення інтерактивності і динамічності веб-сторінок за рахунок зменшення об’єму інформації, що завантажується;
- зменшення навантаження на сервер, що також дуже важливо, враховуючи постійне зростання потоків інформації в мережі «Інтернет».
- крім того, AJAX забезпечує покращення функціональності сайту.

Найбільш яскравим і відомим прикладом використання AJAX можна назвати Google Maps. Взагалі, використання AJAX на сайтах Google підтвердило його ефективність. Активно використовується AJAX в різноманітних форумах, чатах, соціальних мережах.

Який механізм роботи AJAX? Через запит до сервера генерується сторінка, яку буде бачити користувач. Запити користувача будуть звертатися до AJAX-модулю, який забезпечує роботу з сервером через динамічні звернення. Інформація з бази даних зберігається в XML-файлі, який формується динамічно і виводить інформацію на сторінку сайту. AJAX передбачає асинхронний зв'язок. Це означає, що події не наступають негайно після певної дії, а може пройти достатньо часу, перш ніж буде отримано відповідь. На деякі запити відповідь взагалі можна і не отримати.

До основних ***недоліків AJAX*** слід віднести:

- безпеку (можливість прочитати вихідний код у браузері),
- неможливість реєстрації браузерами в історії відвідування сторінок (не працюватиме кнопка «Назад»),
- проблеми індексації пошуковими системами (динамічно завантажений контент недоступний для пошукових роботів).
- тому доцільно використовувати AJAX тільки для окремих частин контенту Вашого сайту.

3. Особливості технології AJAX

AJAX (Asynchronous JavaScript And XML - асинхронний JavaScript і XML) представляє собою технологію, що дозволяє при необхідності у фоновому режимі (не перериваючи роботи користувача і непомітно для нього) виконувати запити до серверу і отримувати додаткові дані для відновлення окремих частин Web-сторінки, тим самим виключаючи необхідність повторної завантаження сторінки. Наприклад, виконувати на стороні сервера перевірку правильності заповнення даних користувачем по мірі їх введення.

Без використання технології AJAX для вирішення цієї задачи є такі можливості:

- виконувати перевірку на стороні сервера, але в цьому випадку необхідно формировать нову Web-сторінок, що збільшує завантаження мережі і збільшує час очікування клієнта;
- виконувати перевірку на стороні клієнта, але при цьому часто необхідно пхранить великий обсяг інформації на комп'ютері клієнта.

Для застосування AJAX необхідні наступні компоненти:

- ***JavaScript*** (основний компонент);
- об'єкт XMLHttpRequest;
- ***серверні технології*** (наприклад, PHP).

Спочатку технологію AJAX розробила фірма Microsoft як об'єкт керування ActiveX після браузера Internet Explorer. Потім фірма Mozilla створила об'єкт XMLHttpRequest з (майже) ідентичними API, який в даний час підтримується всіма сучасними браузерами.

Питання для самоконтролю

1. З якою метою використовується технологія AJAX?
2. Назвіть переваги технології AJAX.
3. Назвіть недоліки технології AJAX.
4. Охарактеризуйте особливості технології AJAX.

СПИСОК ЛІТЕРАТУРИ

1. Пасічник О. В., Пасічник В. В. Веб-дизайн: Підручник. – Львів: Магнолія 2006, 2017. – 570 с.

2. Пасічник О. В., Пасічник В. В., Угрин Д. І. Веб-технології: Підручник. – Львів: Магнолія 2006, 2015. – 336 с.

3. Web-технології та Web-дизайн [Текст]: частина друга: конспект лекцій для студентів 4-го курсу спеціальності 5.05010101 "Обслуговування програмних систем і комплексів" денної форми навчання / уклад. Л.В.Мелещук – Ковель: КПЕК Луцького НТУ, 2015. – 92 с.

4. Мержевич В. HTML и CSS на прикладах. – СПб., 2004. – 441 с.

5. Дари К., Бринзаре Б., Черчез-Тоза Ф., Бусика М. AJAX и PHP. Разработка динамических веб-приложений. – М.: Солон-Пресс, 2006. – 336 с.

6. Уилсон П. Основы JavaScript. - СПб.: Символ, 2002.

Web-технології та Web-дизайн [Текст]: Конспект лекцій для здобувачів освітньо-кваліфікаційного рівня фаховий молодший бакалавр IV курсу спеціальності комп'ютерні науки денної форми навчання. /уклад. Л.В. Мелещук. – Ковель : ВСП «КФПЕК Луцький НТУ», 2021. – 93.

Комп'ютерний набір і верстка: Л.В.Мелещук

Редактор: Л.В.Мелещук

Підп. до друку _____2021. Формат 60x84/16.Папір офс. Гарн. Таймс.
Ум. друк. арк.3,4.
Обл.-вид. арк. 7,4. Тираж _____прим. Зам.188.

ВСП «Ковельський промислово-економічний фаховий коледж
Луцького національного технічного університету»
45000 м. Ковель, вул. Заводська, 23
Друк – ВСП «КПЕФК Луцького НТУ»