МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

В. М. Бутенко, Є. П. Павленко, О. В. Головко

ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ. WEB-ПРОГРАМУВАННЯ

Навчальний посібник

Харків – 2019

УДК 004.4:681

Б 93

Рекомендовано вченою радою Українського державного університету залізничного транспорту як навчальний посібник (витяг з протоколу № 7 від 27 вересня 2018 р.)

Рецензенти:

професор, д-р техн. наук В. О. Філатов (ХНУРЕ),
професор, д-р техн. наук Г. Ф. Кривуля (ХНУРЕ)

Бутенко В. М., Павленко Є. П., Головко О. В.
Б 93 Інженерія програмного забезпечення. WEB-програмування: Навч. посібник. – Харків: УкрДУЗТ, 2019. – 127 с., табл. 14.

ISBN

У навчальному посібнику розглянуто: основи мови розмітки HTML версії 4.0, за допомогою яких реалізовується форматування WEB-сторінок; базові обчислювальні алгоритми сучасною мовою програмування JavaScript, яка дає змогу реалізовувати на WEB-сторінках інтерактивні елементи, а також основи мови програмування PHP, яка дозволяє обробляти інформацію з бази даних, висвітлюючи її на WEB-сторінках. Реалізація всіх завдань виконується з використанням інтегрованого середовища програмування Eclipse (Kepler 2013) версії 4.3, яке безоплатно розповсюджується для викладачів та студентів.

Навчальний посібник призначено для студентів усіх форм навчання спеціальності 123 – Комп'ютерна інженерія першого рівня вищої освіти – бакалавр з можливим застосуванням для споріднених курсів.

УДК 004.4:681

ISBN

Навчальний посібник

Бутенко Володимир Михайлович,
Павленко Євген Петрович,
Головко Олександра Володимирівна

ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ. WEB-ПРОГРАМУВАННЯ

Відповідальний за випуск Павленко Є. П.

Редактор Буранова Н. В.

Підписано до друку 27.09.18 р.

Формат паперу 60х84 1/16. Папір писальний.

Умовн.-друк.арк. 6,0. Тираж 50. Замовлення №

Видавець та виготовлювач Український державний університет залізничного транспорту,
61050, Харків-50, майдан Фейєрбаха, 7.
Свідоцтво суб'єкта видавничої справи ДК № 6100 від 21.03.2018 р.

ЗМІСТ

ВСТУП

Дисципліна «Інженерія програмного забезпечення. WEB-програмування» вивчає засоби розробки Web-сторінок за допомогою мови розмітки HTML та мов програмування JavaScript та/або PHP.

HTML є стандартним засобом для подання інформації так, щоб її можна було побачити незалежно від типу використовуваного комп'ютера та операційної системи. Дві найважливіших властивості HTML − простота і платформна незалежність.

JavaScript − об'єктно-орієнтована інтерпретована мова програмування, призначена для створення сценаріїв, які динамічно змінюють зовнішній вигляд Web-сторінки.

При створенні мови PHP враховано вимогу високої інтеграції з базами даних. Це стало однією з причин того, що PHP є популярною при розробленні Web-додатків.

Навчальний посібник містить чотири розділи і додаток з практичними завданнями для виконання.

У першому розділі розглянуто основи мови розмітки HTML, які дають змогу виконувати форматування Web-сторінок.

Другий розділ присвячений динамічному HTML, включаючи каскадні листи стилів.

Третій розділ надає інформацію про основи мови програмування JavaScript, яка дозволяє реалізовувати на Web-сторінках інтерактивні елементи.

Четвертий розділ містить навчальний матеріал з основ мови програмування PHP, яка дає можливість обробляти інформацію з бази даних, висвітлювати її на Web-сторінках.

Автори розділів:

Бутенко В. М. – вступ, частина контрольних запитань і завдань до розділів навчального посібника у додатку.

Павленко Є. П. – другий, третій та четвертий розділи.

Головко О. В. – перший розділ та завдання у додатку.

1. МОВА РОЗМІТКИ ГІПЕРТЕКСТУ HTML

1.1. Поняття HTML

Всесвітня павутина WWW складається з Web-сторінок, які створюються за допомогою мови розмітки гіпертексту HTML (HyperText Markup Language). HTML – це не мова програмування, це мова розмітки документа. HTML-документ – це звичайний текстовий документ, що містить позначки HTML. Позначки в текстовому документі служать для вказівки форми подання інформації, що міститься в документі.

HTML було створено як стандартний засіб для подання інформації так, щоб її можна було побачити незалежно від типу використовуваного комп'ютера і ОС [1].

Дві найважливіші переваги HTML − простота і платформна незалежність.

Програми перегляду HTML-документів називаються браузерами. Ці програми служать для інтерпретації файлів, розмічених за допомогою HTML, і відображення їх вмісту на екрані комп'ютера.

Мова HTML містить директиви, або теги. Це символи, які керують відображенням тексту і при цьому самі не відображаються на екрані.

Усі теги мови HTML виділяються символами-обмежувачами (<I>), між якими записується ідентифікатор тега і, можливо, його параметри.

Приклад: <I> курсив </ I>

Теги коментарю записуються з більш складними обмежувачами <! - і ->).

Назви тегів, а також їхніх параметрів можна записувати на будь-якому регістрі.

Більшість тегів HTML використовується попарно, тобто для кожного тега, що відкриває, в документі є відповідний тег, що закриває. За правилами HTML тег, що закриває, записується так само, як і той, що відкриває, але із символом / (прямий слеш) перед ім'ям тега. Теги, що закривають, не використовують параметри.

Теги, які потребують відповідних завершальних тегів, називаються тегами-контейнерами. Все, що записано між тегом, що відкриває, і тегом, що закриває, називається вмістом тега-контейнера. Іноді завершальний тег можна опускати. Наприклад, для тега, що описує дані для елемента таблиці <TD>, відповідний закриваючий тег </ TD> можна опускати. Закінчення даних для елемента таблиці буде розпізнано за появою чергового тега <TD> або тега закінчення рядка таблиці </ TR>.

Сучасні браузери в багатьох випадках правильно форматують документи, якщо опущені деякі завершальні теги.

Ряд тегів не потребує завершальних. Прикладами можуть служити тег вставки зображень <IMG>, примусового переведення рядка
 та ін.

На відміну від мов програмування, в яких помилкові оператори призводять до видачі відповідних повідомлень на етапі компіляції програми, неправильно записаний тег або його параметр ігнорується браузером. Це загальне правило для всіх браузерів, під дію якого підпадають не тільки помилково записані теги, але і теги, які не розпізнаються цією версією браузера.

Теги можуть записуватися з параметрами або атрибутами. Набори допустимих параметрів індивідуальні для кожного тега. Вони записуються після імені тега і відокремлюються один від одного пробілами. Порядок проходження параметрів тега довільний. Багато параметрів потребують указування їх значень, проте деякі характеристики не мають значень або можуть записуватися без них, набуваючи значень за замовчуванням. Якщо параметр вимагає значення, то воно вказується після назви параметра через знак рівності. Значення параметра може записуватися як у лапках, так і без них. Лапки обов'язкові, коли в значенні параметра є пробіли.

Приклад запису тега з параметрами:

<TABLE BORDER = 0 ALIGN = "LEFT">

1.2. Специфікації HTML

В середині 90-х років постала потреба стандартизації мови, оскільки різні компанії, які розробляли програмне забезпечення

для доступу в Інтернет, пропонували свої варіанти інструкцій HTML, кількість яких щодалі збільшувалась. Настала пора дійти якоїсь єдиної угоди в частині застосування тегів мови HTML.

Роботу зі створення специфікації HTML взяла на себе організація World Wide Web Consortium (W3C). Її завданням було складання специфікації, що відбиває сучасний рівень можливостей мови з урахуванням різноманітних пропозицій компаній-розробників браузерів. У 1995 р. прийнято специфікацію HTML 2.0.

Схема затвердження специфікацій полягає у такому. Консорціум W3C випускає проект специфікації. Після обговорення його випускається «чорновий» варіант специфікації і пропонується до обговорення на певний період. Після періоду обговорення робочий варіант специфікацій може стати рекомендацією, тобто офіційно визнаним варіантом специфікації HTML.

У 1996 р. було випущено проект HTML 3.2. У 1997 р. ця версія стала офіційною рекомендацією.

У 1997 р. вийшла специфікація HTML 4.0. У цій специфікації ключовою ідеєю стало відділення опису структури документа від опису його подання на екрані. Досвід показує, що поділ структури та подання документа зменшує витрати на підтримку широкого спектра платформ, середовищ, а також полегшує внесення виправлень у документи. Відповідно до цієї ідеї слід ширше користуватися методами опису подання документа за допомогою таблиць стилів, замість того щоб розміщувати дані про форму подання упереміш зі змістом документа [2].

Нове в HTML 4.0 – це можливість для користувача змінювати і динамічно керувати виведенням на екран тексту і графіки без потреби оновлення сторінки. Тому HTML 4.0 назвали "динамічним".

DHTML для досягнення багатьох ефектів використовує мови сценарію JavaScript і VBScript, не покладаючись тільки на теги.

Для реалізації цієї ідеї у специфікації HTML 4.0 ряд тегів, що використовуються для безпосереднього задання форми подання HTML-елементів, скасовано. Наприклад, до скасованих з цієї причини тегів належать <CENTER>, <FONT>,

<BASEFONT>, <STRIKE>, <U>. Замість скасованих тегів пропонуються альтернативні варіанти реалізації відповідних можливостей.

Поняття скасованого тега полягає у такому. Якщо в даній специфікації мови тег названий скасованим, то це означає, що браузери мають поки продовжувати підтримку таких тегів, але їх використання не рекомендується. У наступних специфікаціях ці теги, можливо, будуть переведені в розряд застарілих. Застарілі теги можуть більше не підтримуватися браузерами.

Офіційна специфікація має забезпечувати однакову форму подання інформації різними браузерами. Постійно виникають нові ідеї, реалізовані компаніями-розробниками у своїх браузерах і пропаговані ними. Вдалі ідеї приживаються. Частина можливостей так і залишається специфічними особливостями окремого браузера. Успішні розробки в підсумку потрапляють у специфікацію і стають загальноприйнятними. Таким чином, процес удосконалення можливостей браузерів і уточнення специфікації йде безперервно, справляючи взаємний вплив одне на одного [3].

Специфікацію HTML 5 введено у 2014 р. Її мета – поліпшити підтримку мультимедіа-технологій і зберегти читаність коду. HTML 5 розроблено як заміну одночасно і HTML, і XHTML. У ній існує набагато більша кількість можливостей створення складних веб-додатків.

HTML 5 уводить кілька нових елементів і атрибутів, які відображають типове використання розмітки на сучасних веб-сайтах. Деякі з них − семантичні заміни для використання універсальних блокових ( ) і малих () елементів, наприклад, <nav> (блок навігації по сайту), <footer> (зазвичай відноситься до нижньої частини сторінки або останнього рядка HTML коду) або <audio> і <video> замість <object>. Деякі застарілі елементи, які можна було використовувати в HTML 4.0, було вилучено, включаючи оформлювальні елементи, такі як <font> і , чиї ефекти виконуються за допомогою каскадних таблиць стилів.

Специфікація HTML 5 висуває вимоги як до браузерів, так і до документів. Документи можуть не завжди містити коректний синтаксис, але HTML 5-сумісні браузери застосовують алгоритми

роботи над помилками розмітки в документах для побудови правильної об'єктної моделі (DOM). Чітке визначення вимог до браузерів виконується з метою досягнення сумісності між браузерами різних виробників, так само, як і вимоги до синтаксису розмітки документів з метою коректного відображення їх у різних браузерах. У старих версіях браузерів нові теги HTML 5 просто ігноруються.

1.3. Структура HTML-документа

Першим тегом, з якого слід починати опис документів HTML, є тег <HTML>. Він має завжди починати опис документа, а завершувати опис документа має тег </ HTML>. Ці теги позначають, що рядки, які розташувалися між ними, складають єдиний HTML-документ.

Найчастіше тег <HTML> використовується без параметрів.

Більшість сучасних браузерів можуть розпізнати документ і без тегів <html> і </ html>.

Документ зазвичай складається з двох розділів − розділу заголовка (що починається тегом <head>) і розділу змістової частини документа (що починається тегом <body>). Для документів, що описують фреймові структури, замість розділу BODY використовується розділ FRAMESET (з тегом <FRAMESET>).

Розділ документа HEAD визначає його заголовок і не є обов'язковим тегом. Завданням заголовка є подання необхідної інформації для програми, що інтерпретує документ.

Тег-контейнер <TITLE> служить для того, щоб дати документу назву. Вона зазвичай відображається в заголовку вікна браузера. Тег <TITLE> є текстовим рядком.

Оскільки тег <TITLE> розташовується практично на самому початку HTML-файлу, то після початку завантаження документа в першу чергу відображається саме заголовок. Далі виконується завантаження основного вмісту документа, при цьому браузер починає форматування документа у вікні. Цей процес залежить від вмісту і структури документа, а також швидкості з’єднання. Він може затягнутися. Протягом деякого часу користувач бачитиме порожній екран з назвою документа.

Назва має бути такою, щоб користувач не втратив бажання дочекатися завантаження всього документа.

Часто HTML-документи пов'язані між собою, тобто мають посилання один на одного. Посилання можуть бути як абсолютні, так і відносні. І ті, й інші мають недоліки. Абсолютні посилання можуть бути занадто громіздкими і переставати працювати, якщо переміщений молодший за ієрархією документ. Відносні посилання легше вводити і оновлювати, але цей зв'язок обривається, якщо переміщений старший за ієрархією документ. Обидва види зв'язків можуть порушитися при перенесенні документа з одного комп'ютера на інший.

Теги <BASE> і <LINK> включаються в заголовок для того, щоб зв'язок між документами не порушувався.

Тег <BASE> служить для вказівки повної базової URL-адреси документа. За її допомогою відносні посилання продовжують працювати, якщо документ переноситься в інший каталог або навіть на інший комп'ютер. Тег <BASE> працює аналогічно до команди path MS-DOS.

Тег <BASE> має один обов'язковий параметр <HREF>, після якого вказується повна URL-адреса документа.

Приклад:

<BASE HREF = "// www. google.com">

Для того щоб показати логічний зв'язок між документами, у HTML введено тег <LINK>. Він складається з URL-адреси та параметрів, які конкретизують відношення між документами. Заголовок документа може містити будь-яку кількість тегів <LINK>.

Параметри тега <LINK>:

HREF вказує на URL-адресу іншого документа.

REL визначає відношення між поточним і іншим документом.

REV визначає відношення між іншим документом і поточним (відношення, зворотне REL).

TYPE вказує тип і параметри таблиці стилів

Приклад:

<LINK REL = "CONTENTS" HREF = "... / serg.html">

Перший рядок вказує на зв'язок із файлом змісту документа serg. html прямим відношенням contents.

Між документами може існувати багато різних відношень. Приклади інших значень параметра REL: bookmark, copyright, glossary, help, home, index, next, previous. Параметр REV може набувати значень: author, editor, publisher, owner.

Тег <META> керує діями браузерів, серверів і використовується для розширення інформації, що видається звичайними заголовками.

Коли HTML-документ запитується браузером, сервер, на якому лежить даний текст, відповідно до вмісту тегів <META> вписує у відповідні поля HTTP-пакетів необхідні значення.

Параметри тега <META>:

HTTP - EQUIV визначає властивість для тега.

NAME - додатковий опис тега. Якщо цей параметр опущений, він вважається еквівалентним параметру HTTP-EQUIV.

URL визначає адресу документа.

CONTENT визначає значення, що повертається для властивості.

Розглянемо можливі значення параметра HTTP-EQUIV.

Значення Pragma застосовується для контролю кешування для протоколу HTTP. Значення CONTENT може бути тільки одне − «no-cache» (не кешувати даний документ).

<META HTTP-EQUIV = "Pragma" CONTENT = "no-cache">

Значення Content-Type використовується для вказівки кодування тексту.

Приклад:

<META HTTP-EQUIV = "Content-Type" CONTENT = "text / html; charset = Windows-1251">

Однак якщо текст документа в кодуванні Windows має значення charset = KOI8-r, то користувач не зможе змінити encoding, щоб побачити нормальний текст.

Значення Content-language використовується для значення мови документа. Може використовуватися пошуковими

машинами при індексуванні сторінок. Приклад встановлення мови-діалекту «Англійська-Великобританія»:

<META HTTP-EQUIV = "Content-language" CONTENT = "en-GB">

Значення refresh встановлює час затримки в секундах, після якої браузер автоматично оновлює документ.

Приклад:

<META HTTP - EQUIV = refresh CONTENT = "120" URL = "www. google.com ">

Після завантаження документа браузер чекатиме 120 секунд, а потім буде завантажуватися новий документ. Така конструкція часто використовується при зміні місця розташування документів. Невеликий документ з поданим рядком може бути залишений на старому місці розташування документа для автоматичного посилання на його нове місце розташування.

Приклад:

<META HTTP - EQUIV = refresh CONTENT = "120">

Браузер буде перезавантажувати сторінку кожні 120 секунд.

Значення Cache-Control визначає дії кеша по відношенню до даного документа. Можливі значення:

public − документ кешується в усіх кешах;

private − тільки в приватному кеші;

no-cache − не може кешуватися.

Розглянемо можливі значення параметра NAME.

Значення keywords використовується для вказівки ключових слів, що використовуються пошуковими системами. Цей спосіб дає можливість включати в індекс документа додаткові слова, які можуть явно не входити в його зміст.

Приклад:

<META NAME = "keywords" CONTENT = "user">

Значення Document-state використовується для управління індексацією сторінки для пошукових роботів. Визначає частоту індексації − або один раз індексувати, або реіндексувати документ регулярно. Можливі значення:

Static

Dynamic

<META NAME = "Document-state" CONTENT = "Dynamic">

У розділі заголовка документа можуть бути наявні ще два теги − <STYLE> і <SCRIPT>. Їх призначення пов'язане з використанням таблиць стилів у документі і записом скриптів [47].

1.4. Розділ документа BODY

У цьому розділі документа розташовується його змістова частина.

Розділ документа BODY має починатися тегом <BODY> і завершуватися тегом </ BODY>, між якими розташовується весь зміст даного розділу. Наявність цих тегів не є обов'язковою, оскільки браузери можуть визначити початок змістової частини документа за контекстом.

Параметри тега <BODY> (вони не є обов'язковими).

BACKGROUND вказує URL зображення, яке використовується як фонове.

BGCOLOR визначає колір фону документа.

BGSOUND − звуковий фон документа.

BGPROPERTIES − якщо встановлено значення FIXED, фонове зображення не прокручується.

ALI NK визначає колір активного посилання.

LINK визначає колір ще не переглянутого посилання.

VLINK визначає колір вже переглянутого посилання.

SCROLL встановлює наявність або відсутність смуг прокрутки вікна браузера.

TEXT визначає колір тексту.

LEFTMARGIN встановлює межу лівого поля документа в пікселах (відстань у пікселях між краєм тексту і лівим краєм вікна).

RIGHTMARGIN встановлює межу правого поля документа.

BOTTOMMARGIN встановлює межу нижнього поля документа.

TOPMARGIN встановлює межу верхнього поля документа.

Приклад:

<BODY BACKGROUND = "NAVYCON.GIF" BGPROPERTIES = FIXED SCROLL = YES TEXT = YELLOW LINK = "# FFAA00" BGCOLOR = GREEN RIGHTMARGIN = 10>

У мові HTML кольори визначаються цифрами в шістнадцятковому коді. Колірна система базується на трьох основних кольорах − червоному, зеленому і синьому і позначається RGB. Для кожного кольору задається шістнадцяткове значення в межах від 00 до FF, що відповідає діапазону 0-255 у десятковому численні. Потім ці значення об'єднуються в одне число, перед яким ставиться символ #.

Замість 16-кової цифри можна користуватися назвами кольорів. Браузери розпізнають 140 назв кольорів.

Крім BACKGROUND, треба задавати і BGCOLOR, оскільки користувач може заборонити завантаження фонового рисунка.

Фонове зображення для HTML-документа заповнює все вікно перегляду. Якщо розмір зображення менший за вікно перегляду, воно розмножується.

Зазвичай як фонове береться невелике зображення, для завантаження якого по мережі не потрібно багато часу.

1.5. Форматування тексту

Логічне і фізичне форматування.

Для форматування тексту HTML-документів передбачено групу тегів, яку можна розділити на теги логічного і фізичного форматування.

Теги логічного форматування позначають структурні типи текстових фрагментів, такі, наприклад, як програмний код (тег <CODE>), цитата (тег <CITE>) і т. ін. Мова йде про структурну

розмітку, яка не впливає на конкретне екранне подання фрагмента браузером. Тому така розмітка називається логічною. Фрагменти з логічним форматуванням браузери відображають на екрані певним чином, заданим за замовчуванням. Вид відображення жодним чином не пов'язаний зі структурним типом фрагмента.

Теги фізичного форматування визначають формат відображення вказаного в них фрагмента тексту у вікні браузера. Наприклад, для відображення фрагмента курсивом можна використовувати тег курсиву < I >.

Між розробниками HTML-документів тривалий час точилися суперечки про переваги і недоліки того чи іншого підходу. З виходом специфікації HTML 4.0 ці суперечки завершилися на користь застосування логічного форматування, оскільки було проголошено принцип відділення структури документа від його подання.

Проте на сьогодні може вільно використовуватися і фізичне форматування. У специфікації HTML 4.0 деякі теги фізичного форматування не рекомендуються для застосування, однак поки вони все ще підтримуються всіма браузерами. Деякі теги логічного форматування, покликані замінити теги фізичного форматування, розпізнаються не всіма браузерами, що робить їх застосування незручним. Приклад − логічний тег <DEL>, який рекомендується використовувати замість фізичного тега <STRIKE>.

Розглянемо теги логічного форматування.

Тег <CITE> використовується для позначення цитат або назв книг і статей. Браузерами такий текст зазвичай виводиться курсивом.

Приклад:

< CITE > Цитата </ CITE>

Тег <CODE> зазначає фрагмент тексту як невеликий фрагмент програмного коду. Відображається дрібним шрифтом.

Тег < DEL > зазначає текст як віддалений. Цей елемент корисно використовувати для оцінки змін, що вносяться до документа від версії до версії.

Текст, позначений тегом <DEL>, зазвичай відображається перекресленим текстом. У специфікації HTML 4.0 цьому тегу

віддається перевага перед тегом фізичного форматування < STRIKE > , що означає перекреслений текст.

Тег < INS > зазначає текст як вставку. Цей елемент корисно використовувати для оцінки змін, що вносяться до документа від версії до версії.

Текст, позначений тегом < INS >, зазвичай відображається підкресленим текстом.

Тег < DFN > зазначає свій текстовий фрагмент як визначення. Наприклад, цим тегом можна зазначити будь-який термін, коли він трапляється в тексті вперше. Тег < DFN > відображується за замовчуванням курсивом.

Тег < EM > використовується для виділення важливих фрагментів тексту. Браузери зазвичай відображають такий текст курсивом. Застосування цього тега є кращим, ніж застосування тега фізичного форматування < i >.

Тег <STRONG> використовується для виділення важливих фрагментів тексту. Браузери зазвичай відображають такий текст напівжирним шрифтом. Тегом < STRONG > зазвичай розмічають більш важливі фрагменти тексту, ніж ті, що розмічені тегом <EM>.

Тег <KBD> зазначає текст, який вводиться користувачем з клавіатури. Зазвичай відображається дрібним шрифтом. Застосування цього тега є кращим, ніж застосування тега фізичного форматування <TT>.

Тег <Q> зазначає короткі цитати в рядку тексту. Зазвичай відображається курсивом.

Тег < SAMP > зазначає текст як зразок. Звичайне використання цього тега − відмітка тексту, що видається програмами. Використовується також для виділення декількох символів моноширінним шрифтом.

Використання цього тега є кращим, ніж застосування тега фізичного форматування <TT>.

Деякі теги дають однаковий результат. Вони призначені для розстановки логічних наголосів, виділення логічних частин та підкреслювання суті висловлювань. Їх використання є вельми актуальним. Застосовуючи таблиці стилів, можна задати бажане відображення для будь-якого з тегів, перевизначивши значення за замовчуванням.

Розглянемо теги фізичного форматування тексту.

Тег <B> відображає текст напівжирним шрифтом. У більшості випадків рекомендується замість тега <B> використовувати тег логічного форматування < STRONG >.

Тег <I> відображає текст курсивом. Замість цього тега рекомендується використовувати теги <EM>, <DFN > або < CITE >, оскільки останні краще відображають призначення тексту, що виділяється.

Тег <TT> відображає текст шрифтом друкарської машинки. Замість цього тега краще використовувати теги < CODE > або < SAMP >.

Тег <U> відображає текст підкресленим. Скасований тег. Замість нього рекомендується використовувати теги < STRONG > або < CITE >.

Тег <STRIKE> відображає текст, перекреслений горизонтальною лінією. Скасований тег. Замість нього слід використовувати тег <DEL>.

На сьогодні тег <DEL > підтримується не всіма браузерами, тому поки рекомендується використовувати <DEL> у поєднанні з тегом <STRIKE>: а саме, всередину тега-контейнера <DEL> можна вкласти пару тегів

< STRIKE >. .. </ STRIKE >.

Тег <BIG> виводить текст шрифтом більшого (ніж інша частина тексту) розміру. Замість цього елемента краще використовувати < STRONG > або теги заголовків, наприклад, <H3>. Більшість браузерів підтримують вкладені теги < BIG >.

Тег <SMALL> виводить текст шрифтом меншого розміру. Більшість браузерів підтримують вкладені теги <SMALL> .

Тег <SUB> зсуває текст нижче рівня рядка і виводить його шрифтом меншого розміру. Використовується для відображення нижнього індексу.

Тег <SUP> зсуває текст вище рівня рядка і виводить його шрифтом меншого розміру. Використовується для виведення верхнього індексу .

Теги форматування можуть бути вкладеними один в одного. При цьому треба стежити, щоб один контейнер містився цілком у другому контейнері.

< B > < I > напівжирний і курсивний </ I > текст </ B >

Тег <FONT> вказує параметри шрифту.

Призначення параметрів шрифту безпосередньо в тексті документа порушує основну ідею поділу змістової частини документа і опису форми подання документа. Тому у специфікації HTML 4.0 даний тег, а також тег <BASEFONT> віднесені до скасованих. Їх подальше застосування не рекомендується.

Незважаючи на це, для простих документів фізичне форматування можна вважати допустимим. Крім того, починати навчання основ форматування найпростіше саме з правил безпосереднього зазначення форматів елементів.

Для тега можуть задаватися такі параметри: FACE, SIZE і COLOR.

Приклад:

<FONT FACE = "ARIAL", "TAHOMA", "VERDANA" SIZE = 4 COLOR = "RED"> ... </ FONT >

Параметр FACE служить для вказівки типу шрифту, яким броузер буде виводити текст. Якщо такого шрифту не буде знайдено, то цю вказівку буде проігноровано і буде використано шрифт, встановлений за замовчуванням.

Можна вказати як один, так і кілька назв шрифтів, розділяючи їх комами. У цьому полягає перевага використання шрифтів. Список шрифтів проглядається зліва направо. Якщо на комп'ютері користувача немає шрифту, вказаного у списку першим, то робиться спроба знайти наступний шрифт.

Параметр SIZE служить для вказівки розмірів шрифту в умовних одиницях від 1 до 7. Конкретний розмір шрифту залежить від використовуваного браузера.

Налаштування розмірів шрифту, використовуваних за замовчуванням, залежать від браузерів. Можна переналаштувати їх у браузері.

Розмір шрифту вказується як абсолютною величиною (SIZE = 5), так і відносною (SIZE = + 1). Останній спосіб часто використовується в поєднанні із заданням базового розміру шрифту за допомогою тега < BASEFONT >.

При вказівці розмірів шрифтів записи типу "2" і "+2" дають принципово різний результат.

Параметр COLOR встановлює колір шрифту, який може задаватися за допомогою стандартних імен або у форматі #RRGGBB.

Тег <BASEFONT> використовується для вказівки розміру, типу і кольору шрифту, який використовується в документі за замовчуванням. Ці значення є обов'язковими для всього документа, однак можуть у потрібних місцях перевизначатися за допомогою тега < FONT >. Після тега </ FONT > дія тега <BASEFONT> відновлюється. Значення параметрів шрифтів, використовуваних за замовчуванням, можуть неодноразово перевизначатися в документі, тобто тег <BASEFONT> може з'являтися в документі будь-яку кількість разів.

Тег <BASEFONT> може з'являтися також і в розділі <HEAD> документа. Для тега <BASEFONT> не існує тега, що закриває.

Як параметри можуть використовуватися такі самі параметри, що і для тега < FONT >, а саме: FACE , SIZE і COLOR . Призначення і правила запису параметрів аналогічні.

Одним із перших правил складання будь-якого документа є розбиття його тексту на окремі абзаци. При створенні документів за допомогою текстових редакторів розбиття на абзаци виконується введенням символа переведення рядка (зазвичай <Enter>). У HTML-документах символи переведення рядка не утворюють нового абзацу.

HTML передбачає, що автор документа нічого не знає про комп'ютер свого читача. Читач має право встановити будь-який розмір вікна і користуватися будь-яким із наявних у нього шрифтів. Це означає, що місце перенесення в рядку визначається тільки програмою перегляду читача. Тому для задання місця переведення рядка існують спеціальні теги поділу на абзаци і примусового переведення рядка.

Перед початком кожного абзацу тексту слід помістити тег <P>. Закривати тег </ P> не обов'язково. Браузери зазвичай відокремлюють абзаци один від одного порожнім рядком.

Браузери зазвичай інтерпретують кілька тегів абзацу <P> як один. Тому задати кілька порожніх рядків за допомогою цих тегів не вдається.

Тег <P> може задаватися з параметром горизонтального вирівнювання ALIGN. Значення параметра:

LEFT − вирівнювання тексту по лівій межі вікна браузера (за замовчуванням).

CENTER − вирівнювання по центру вікна браузера.

RIGHT − вирівнювання по правій межі вікна браузера.

JUSTIFY − вирівнювання по ширині (по двох сторонах).

Вирівнювання по ширині (ALIGN = JUSTIFY) тривалий час не підтримувалося браузерами.

При відображенні текстових документів у браузері перенесення рядка може здійснюватися тільки за символами-розділювачами слів (наприклад, пробілів). Іноді потрібно задати примусове переведення рядка, що реалізується незалежно від налаштувань браузера. Для цього служить тег примусового переведення рядка
, який не має відповідного тега, що закриває.

На відміну від тега абзацу < P > при використанні < BR > не з'явиться порожній рядок.

Використання тега
 потребує обережності − можлива ситуація, коли браузер вже зробив новий рядок за одне-два слова до того, як зустрів тег
.

Бувають ситуації, коли потрібно виконати операцію протилежного призначення − заборонити новий рядок. Для цього існує тег-контейнер <NOBR>. Текст, розмічений цим тегом, буде гарантовано розміщуватися в одному рядку, незалежно від його довжини. Якщо при цьому рядок буде виходити за межі вікна перегляду браузера, то з'явиться горизонтальна смуга прокрутки.

Розмічаючи текст за допомогою тега <NOBR>, можна отримати дуже довгі рядки. Щоб цього уникнути, можна вказати браузеру місце можливого переведення рядка, що буде виконано тільки за потреби (так званий "м'який" перехід рядка). Це можна виконати за допомогою тега <WBR> , який так само, як і тег
, не потребує тега, що закриває.

Поряд із назвою всього документа на Web-сторінці можуть використовуватися заголовки для окремих частин документа. Ці заголовки можуть мати шість різних рівнів (розмірів).

Для розмітки заголовків використовуються теги <H1>, <H2>, <H3>, <H4>, <H5> і <H6>. Ці теги потребують тега, що закриває. Тема з номером 1 є найбільшим (заголовок верхнього рівня), а з номером 6 – найдрібнішим. При використанні тегів заголовків здійснюється вставка порожнього рядка до і після заголовка, тому тегів абзацу або переведення рядка тут не потрібно.

Теги заголовків можуть задаватися з параметром горизонтального вирівнювання ALIGN . Значення параметра − ті самі, що і для тега абзацу <P>.

Іншим методом поділу документа на частини є проведення горизонтальних ліній.

Тег <HR> дозволяє провести горизонтальну лінію. Цей тег не є контейнером, тому не потребує тега, що закриває. До і після лінії автоматично вставляється порожній рядок. Назвемо параметри тега <HR>.

ALIGN вирівнює по краю або центру; має значення LEFT , CENTER , RIGHT.

WIDTH встановлює довжину лінії в пікселях або відсотках від ширини вікна браузера.

SIZE встановлює товщину лінії в пікселях.

NOSHADE скасовує рельєфність лінії.

COLOR вказує колір лінії. Використовується формат RGB або стандартне ім'я.

Приклад :

<HR ALIGN = RIGHT WIDTH = 50% COLOR = "YELLOW">

Для розбиття тексту на абзаци і забезпечення примусового переведення рядка слід користуватися спеціальними тегами. Однак бувають випадки, коли в HTML-документ необхідно включити текст, що вже має форматування, виконане традиційним способом за допомогою символів переведення рядка, табуляції і т. ін. Для цього передбачений спеціальний тег-контейнер < PRE >, що визначає попередньо відформатований текст.

Текст, розмічений тегом < PRE >, буде відображатися в такому вигляді, який він має у звичайному текстовому редакторі. Для відображення завжди буде використовуватися дрібний шрифт.

Одним із варіантів використання цього тега є таблиці, побудовані без застосування спеціальних тегів розмітки таблиць. Іншим важливим застосуванням є виведення на екран великих блоків програмного коду, які браузер не повинен переформатувати.

Текст всередині контейнера < PRE > може містити деякі елементи форматування. Не можна ставити такі : <IMG>, <OBJECT>, <APPLET>, <BIG>, <SMALL>, <SUB>, <SUP>, <FONT>, <BASEFONT>. Не можна ставити теги заголовків. Тег абзацу буде реалізовувати перехід на новий рядок.

Існують ще кілька тегів, що застосовуються для відображення заздалегідь відформатованого тексту. До них належать теги <XMP>, <PIAINTEXT> і <LISTING>. Усі ці три теги у специфікації HTML 4.0 зазначено як застарілі. У майбутніх версіях браузери припинять їх підтримку. Замість цих тегів рекомендовано використовувати тег <PRE>.

Тег-контейнер <DIV> служить для виділення фрагмента документа. Метою цього виділення є управління параметрами даного фрагмента, яке зазвичай виконується за допомогою призначення стилів.

За допомогою цього тега можна задати вирівнювання тексту:

< DIV ALIGN = LEFT >

...

</ DIV >

Тег-контейнер < CENTER > призначений для центрування текстових та інших елементів посередині вікна браузера. Його корисно використовувати для центрування таблиць.

Для виділення довгих цитат існує тег <BLOCKQUOTE>. Він є контейнером і може містити будь-які теги форматування. Текст, розмічений даним тегом, при відображенні відділяється від основного тексту порожніми рядками і виводиться з невеликим відступом вправо.

Спеціальні символи

Деякі спеціальні символи не входять до базової частини таблиці кодів ASCII. До них належать букви алфавітів частини європейських мов, математичні і деякі інші символи.

Можна вводити потрібні символи в HTML-документ за допомогою спеціальних кодів. Ці коди складаються із символа амперсанда (&) і наступного за ним імені символа або його десяткового або шістнадцяткового значення. Закінчуватися спеціальний символ має крапкою з комою.

Зазначимо деякі символи, використання яких є актуальним і забезпечено підтримкою браузерів.

& Lt <
& Gt>
& Nbsp нерозривний пробіл
& Amp &
& Quot '

Символи можуть бути також задані своїми кодами. Наприклад, символ нерозривного пробілу має код 160. Він може записуватися в десятковому вигляді як #160;

1.6. Посилання на інші документи і файли

Посилання реалізовують можливість перейти від одного документа до іншого, нехай навіть розташованого на сервері в іншій країні.

Посилання складається з двох частин. Перша з них − це те, що видно на Web-сторінці; вона називається покажчиком посилання (anchor). Друга частина, що дає інструкцію браузеру, називається адресною частиною посилання (URL-адресу). При натисканні мишею по покажчику посилання браузер завантажує документ, адреса якого дається URL-адресою.

Покажчиком посилання може бути слово, група слів або зображення. зовнішній вигляд посилання залежить від способів створення і установок браузера читача. Покажчики бувають двох типів − текстові та графічні.

Текстові покажчики являють собою слово або кілька слів, виділених на екрані підкресленням. Колір текстового покажчика може регулюватися автором і установками програми перегляду.

Приклад текстового покажчика посилання:

<A HREF = "serg.html"> Домашня сторінка Сергія </ A>

Як посилання можна використовувати графічне зображення. За принципом дії графічні посилання нічим не відрізняються від текстових. Вони не підкреслюються і не виділяються кольором, а для їх виділення браузери зазвичай навколо такого зображення рисують рамку.

Приклад графічного покажчика посилання:

<A HREF = "serg. Html"> <IMG SRC = "photo. Gif"> </ A>

Адреса може бути відносною або абсолютною.

URL-адресу, яку повністю визначає комп'ютер, каталог і файл, називають абсолютною. Абсолютні покажчики можуть посилатися на файли, розташовані на інших комп'ютерах.

Якщо в URL-адресі не вказується повний шлях до файлу, то таке посилання називається відносним. Наприклад, якщо браузер завантажив сторінку, що розташована за адресою http: // www.serg.ua / page, то відносний покажчик /picture має на увазі адресу http: //www. serg.ua / page / picture, тобто підкаталог, розташований на тій самій машині.

Відносні покажчики зручні у використанні. Вони дають змогу переміщати файли в межах сервера без великих змін в адресації.

Відносний покажчик працює по-іншому, якщо в HTML-документі не використовується тег <BASE>. В цьому випадку покажчик дає адресу щодо URL-адреси, визначеної в тегу <BASE>.

Приклад посилання на адресу електронної пошти:

<A HREE = "mailto: serg @ i.ua "> Пишіть! </ A>

Внутрішні посилання

Крім посилань на інші документи, буває корисно організувати посилання на різні частини поточного документа. Наприклад, великий документ читається краще, якщо він має зміст із посиланнями на відповідні розділи.

Приклад внутрішньої посилання.

<A HREF = "# abzac"> Форматування абзаців </ A>

<A NAME = abzac> </ A>

Посилання на документи різних типів

Коли користувач натискає мишею на посиланні, що вказує на іншу Web - сторінку, вона виводиться безпосередньо у вікні браузера. А якщо посилання вказує на документ іншого типу, то:

1. Браузер розпізнає цей тип документа. Наприклад – посилання на графічний файл формату GIF. Браузер завантажить вказане зображення. У деяких випадках браузер може додатково використовувати програмний модуль (plug - in), без якого задачу не було б розв’язано.

2. Браузер не розпізнає тип прийнятого документа. У цьому випадку він звернеться до допоміжних програм, які є на машині користувача. Якщо відповідна програма знайдеться, браузер запустить її і передасть їй отриманий документ. Наприклад, за посиланням на відеофайл формату AVI браузер завантажить файл, знайде програму для демонстрації AVI-файлів і запустить її. Відеофайл буде показаний у додатковому невеликому вікні [5].

1.7. Робота зі списками

У мові HTML передбачено такі типи списків: маркований, нумерований і список визначень.

Маркований список

Для створення маркованого списку необхідно використовувати тег-контейнер <UL>, усередині якого розташовуються всі елементи списку.

Кожен елемент списку має починатися тегом <LI> (LI - List Item, елемент списку). Тег <LI> не потребує відповідного тега, що закриває, хоча його наявність не заборонено. Браузери

зазвичай при відображенні документа починають кожен новий елемент списку з нового рядка.

Приклад маркованого списку.

<UL>
<LI> П'ятниця
<LI> Неділя
<LI> П'ятниця
</ UL>

У тегу <UL> може бути вказаний параметр TYPE.

Параметр TYPE може набувати таких значень: disc, circle, square. Цей параметр використовується для примусового задання виду маркерів (забарвлені кола, незабарвлені кола, забарвлені квадратики).

За замовчуванням TYPE = disc. Для вкладених маркованих списків на першому рівні за замовчуванням використовується значення disc, на другому − circle, на третьому і далі − square.

Параметр TYPE з тими самими значеннями може вживатися для вказівки видів маркерів окремих елементів списку. Для цього параметр TYPE дозволено вказувати в тегу <LI>.

Графічні маркери

Як маркери списку можна використовувати графічні зображення, що застосовуються для створення красиво оформлених HTML-документів.

Тег списку <UL> виконує єдине завдання − вказує браузеру, що вся інформація, розташована після даного тега, має відображатися із зсувом вправо (відступом) на деяку величину. Теги <LI> забезпечують виведення стандартних маркерів елементів списку.

Щоб побудувати список із графічними маркерами, можна обійтися без <LI>, досить перед кожним елементом списку вставити графічне зображення.

<UL>
<IMG SRC = "Navicon. gif "> П'ятниця
<IMG SRC = "N avicon. gif "> Неділя
<IMG SRC = "Navicon. gif "> П'ятниця
</ UL>

Використання графіки може значно збільшити обсяг Web-сторінки. У цьому випадку рисунок невеликий і передається один раз.

Нумерований список

Для створення нумерованого списку використовується тег-контейнер <OL>. Як і для маркованого списку, кожен елемент нумерованого списку має починатися тегом <LI>.

У тегу <OL> можуть бути вказані такі параметри: TYPE і START.

Параметр TYPE використовується для задання виду нумерації списку. Може набувати таких значень:

TYPE = A − задає маркери у вигляді великих латинських букв;

TYPE = A − задає маркери у вигляді малих латинських букв;

TYPE = I − задає маркери у вигляді великих римських цифр;

TYPE = i − задає маркери у вигляді малих римських цифр;

TYPE = 1 − задає маркери у вигляді арабських цифр.

За замовчуванням TYPE = 1, тобто нумерація за допомогою арабських цифр. Це стосується і вкладених нумерованих списків.

Параметр TYPE може використовуватися для нумерації окремих елементів списку. Для цього параметр TYPE дозволено вказувати в тегу елемента списку <LI>.

Параметр START дозволяє почати нумерацію списку не з одиниці. Як значення параметра START завжди має зазначатися натуральне число, незалежно від виду нумерації списку.

Приклад:

<OL TYPE = A START = 5>

Тег <LI> для нумерованих списків дає можливість використовувати параметри TYPE і VALUE.

Значення параметра VALUE тега <LI> дозволяє змінити номер даного елемента списку. При цьому змінюється нумерація і всіх наступних елементів. Типовим застосуванням є списки з пропуском деяких елементів.

1.8. Вбудовування зображень у HTML-документи

Для вбудовування зображень у HTML-документ використовується тег <IMG>. Обов'язковий параметр SRC визначає URL-адресу файлу із зображенням.

Розглянемо інші параметри тега <IMG>.

Параметр ALIGN задає вирівнювання зображення. Розглянемо можливі значення цього параметра.

При використанні параметрів LEFT або RIGHT виходить "плаваюче" зображення.

LEFT − зображення притискається до лівого поля вікна. Текст обтікає зображення з правого боку.

RIGHT−зображення притискається до правого поля вікна. Текст обтікає зображення з лівого боку.

При використанні інших параметрів зображення вбудовується в рядок тексту, а параметри вирівнювання задають розташування зображення щодо рядка тексту. Таким чином, на відміну від плаваючих зображень, тут зображення є звичайним елементом рядка.

TOP − верхня межа зображення вирівнюється за найвищим елементом рядка.

TEXTTOP − верхня межа зображення вирівнюється за найвищим текстовим елементом рядка.

MIDDLE − вирівнювання середини зображення по базовій лінії рядка.

Базова лінія − це нижня частина лінії тексту, яка проводиться без урахування нижньої частини деяких символів, наприклад, букв j, q, y.

ABSMIDDLE − вирівнювання середини зображення посередині поточного рядка.

BASELINE або BOTTOM − вирівнювання нижньої межі зображення по базовій лінії поточного рядка.

ABSBOTTOM − вирівнювання нижньої межі зображення по нижній межі поточного рядка. Вирівнювання виконується за найнижчим елементом у рядку.

Якщо в документі використовуються плаваючі зображення, вирівняні зі значенням RIGHT або LEFT, то є можливість примусового припинення обтікання в заданому місці тексту. Це забезпечується застосуванням тега примусового переривання рядка
 з параметром CLEAR. Як значення параметра CLEAR можна використовувати такі: LEFT, RIGHT або ALL.

Приклад: <BR CLEAR = right>

Текст, що йде далі, розміститься нижче зображення з нового рядка.

Задання розмірів зображення, що виводиться

Параметри WIDTH і HEIGHT вказують розміри зображення. Значення параметрів можуть зазначатися як у пікселях, так і у відсотках від розмірів вікна перегляду.

Значення параметрів ширини і висоти зображення можуть не збігатися з істинними розмірами зображення. В цьому випадку браузери автоматично при завантаженні зображення виконують його перемасштабування. Неакуратне задання параметрів може призвести до зміни пропорцій рисунка і його спотворення.

Будь-який із цих параметрів може бути опущений. Якщо заданий тільки один з параметрів, то при завантаженні рисунка другий параметр буде обчислюватися автоматично з умов збереження пропорцій.

Якщо задати параметри ширини і висоти менші за реальні значення, ілюстрація буде виведена у зменшеному вигляді, проте це не скорочує час завантаження зображення.

Відокремлення зображення від тексту

Для тега <IMG> можна задавати параметри HSPACE і VSPACE, значення яких визначають відступи від зображення, відповідно, по горизонталі і вертикалі. Це гарантує, що між текстом і зображенням залишиться простір, потрібний для нормального сприйняття.

Приклад:

<IMG src = serg. gif ALIGN = LEFT HSPACE = 10 VSPACE = 15>

Параметр BORDER задає товщину рамки навколо зображення (в пікселях). За замовчуванням рамка навколо зображення не накреслюється. Винятком з цього правила є випадок, коли зображення є посиланням.

Параметр ALT визначає альтернативний текст. Його вказівка дає можливість користувачам, що працюють у режимі відключення завантаження зображень, отримати текстову інформацію про вбудовані зображення [6].

При відключеному зображенні замість нього на екрані з'явиться альтернативний текст, визначений значенням параметра ALT.

Браузери відображають альтернативний текст як підказку при переміщенні курсору миші на зображенні.

1.9. Створення таблиць

Таблиця має починатися тегом <TABLE> і завершуватися тегом </TABLE> .

Приклад:

<TABLE BORDER = 10 CELLSPACING = 5 CELLPADDING = 5 WIDTH = 50% BGCOLOR = BLUE>

Параметр BORDER управляє зображенням рамки навколо таблиці і ліній сітки таблиці. За замовчуванням рамки не накреслюються.

Чисельне значення параметра визначає товщину рамки навколо всієї таблиці в пікселях. На товщину рамок навколо кожної комірки це значення не впливає. Зазвичай товщина приймається рівною мінімальному значенню 1.

За відсутності параметра BORDER рамки не прорисовуються, але місце під них залишається. Можна задати BORDER = 0.

Параметр CELLSPACING визначає відстань між суміжними комірками (між рамками комірок) як по горизонталі, так і по вертикалі. За замовчуванням значення приймається

рівним 2. При заданні CELLSPACING = 0 рамки суміжних комірок зіллються і створять враження єдиної сітки таблиці.

Параметр CELLPADDING визначає розмір відступу між рамкою комірки і даними всередині комірки. За замовчуванням значення приймається рівним 1.

Параметри WIDTH і HEIGHT

При відображенні таблиць їх ширина і висота автоматично обчислюються браузером і залежать від багатьох факторів: значень параметрів таблиці, її рядків і комірок, вмісту комірок, а також параметрів, що задаються при перегляді документа в браузері, наприклад, типу і розмірів шрифту, відображення зображення та ін. При відображенні робиться спроба надати таблицю в найбільш зручному вигляді − розташувати таблицю так, щоб вона містилася у вікні перегляду.

Для примусового задання ширини або висоти таблиці використовуються параметри WIDTH і HEIGHT. Ширина і висота задаються в пікселях або у відсотках від усього розміру вікна. Припустимо задавати значення більше 100 %.

Параметр ALIGN має значення − LEFT (вирівнювання вліво) і RIGHT (вирівнювання вправо). За замовчуванням таблиці вирівняно по лівому краю. Наявність параметра ALIGN дає змогу виконати обтікання таблиці текстом з протилежного боку аналогічно до обтікання картинок. Обтікання таблиці текстом з двох боків не передбачено. Для переривання обтікання слід використовувати тег
 з параметром CLEAR.

Якщо параметр ALIGN опущений, то місце справа і зліва від таблиці завжди буде порожнім незалежно від її ширини. Домогтися розташування таблиці по центру вікна можна, якщо опис таблиці помістити всередині пари тегів <CENTER> і </ CENTER>.

Заголовок таблиці <CAPTION>.

Тег-контейнер заголовка таблиці <CAPTION> має розташовуватися відразу ж після тега <TABLE> і до першого тега <TR>.

Між тегами <CAPTION> і </ CAPTION> допустимо записувати будь-які HTML-елементи, що вживаються в розділі <BODY>.

Тег має параметр ALIGN, що набуває значення TOP (заголовок над таблицею) або BOTTOM (заголовок під таблицею). За замовчуванням ALIGN = TOP.

Інші значення параметра ALIGN: LEFT, CENTER і RIGHT– для горизонтального вирівнювання.

Якщо ALIGN застосований для горизонтального вирівнювання, то для вертикального вирівнювання застосовується VALIGN, який набуває значення TOP або BOTTOM.

Рядки і стовпці таблиці

Кожен рядок починається тегом <TR> і завершується тегом </ TR>. Окрема комірка у рядку обрамляється парою тегів <TD> і </ TD> або <TH> і </ TH>. Тег <TH> використовується для комірок-заголовків таблиці, а <TD> - для комірок-даних. Вміст комірок <TH> відображається за замовчуванням напівжирним шрифтом і розташовується по центру. Комірки, визначені тегом <TD>, за замовчуванням відображаються з вирівнюванням уліво (ALIGN = LEFT).

Кількість рядків у таблиці визначається числом тегів, що відкривають <TR>, а кількість стовпців − максімальною кількістю <TD> і <TH> серед усіх рядків. Якщо одна або кілька комірок, розташованих у кінці рядка, не містить даних, то їх опис може бути опущено, а браузер автоматично додасть необхідну кількість порожніх клітинок.

Форматування даних усередині таблиці

Всередині комірки допустимо використання всіх елементів HTML, які можуть з'являтися всередині розділу <BODY>, в тому числі теги <P>,
, <HR>, коди заголовків − <H1> - <H6>, теги форматування символів − <B >, **<I>,** <STRONG>, <FONT>, тег вставки графічних зображень <IMG>, гіпертекстових посилань <A>. Область дії тегів, заданих всередині окремої комірки, обмежується межами цієї комірки незалежно від наявності завершального тега. Наприклад, якщо всередині комірки вказано тег <EM>, то навіть за відсутності завершального коду </ EM> або його вказівки через кілька елементів таблиці текст наступної комірки буде відображений шрифтом за замовчуванням.

Для форматування даних усередині елементів таблиці передбачені наведені нижче параметри.

Параметри вирівнювання вмісту комірок – ALIGN і VALIGN. Можуть застосовуватися в тегах <TR>, <TD> і <TH>. Параметр горизонтального вирівнювання ALIGN може набувати значень LEFT, RIGHT і CENTER.

Задання параметрів вирівнювання в тегу <TR> визначає вирівнювання для всіх комірок цього рядка, при цьому в кожній окремій клітинці рядка можна визначити свої параметри.

Параметр NOWRAP відключає можливість автоматичного розбиття тексту комірки на рядки. Може застосовуватися в тегах <TR>, <TD>, <TH>.

Параметри WIDTH і HEIGHT можуть застосовуватися в тегах <TD> і <TH>. Значення можуть задаватися в пікселях або у відсотках від розмірів усієї таблиці. Задання ширини комірки впливає на ширину всієї колонки, в якій розташована комірка, а задання висоти впливає на весь рядок. Якщо в колонці значення ширини зазначено лише в одній комірці, то це значення стає шириною всієї колонки. Якщо таких вказівок кілька, то вибирається максимальне значення. Такі самі властивості характерні і для рядків.

Щоб об'єднати кілька сусідніх комірок в одну, застосовуються параметри COLSPAN і ROWSPAN , що задаються в тегах < TD > або <TH>. Форма запису: COLSPAN = num, де num − числове значення, що вказує, на скільки стовпців слід розширити комірку по горизонталі. Застосування параметра ROWSPAN є аналогічним. Припустимо одночасне встановлення значень обох параметрів для однієї комірки.

Параметр BGCOLOR задає колір фону таблиці, окремих рядків і комірок. Може задаватися в тегах < TABLE >, < TR >, < TD >, < TH >.

Вирівнювання даних у стовпцях таблиці

Для вирівнювання вмісту всіх комірок поточного рядка досить задати необхідні параметри в тегу <TR>. Однак найчастіше необхідно забезпечити однакове вирівнювання елементів одного стовпця, оскільки в більшості випадків у стовпці розташовуються однорідні дані.

Передбачено спеціальні теги для об'єднання стовпців у групи і задання вирівнювання − <COL> і <COLGROUP> . Ці теги мають розташовуватися відразу ж за < TABLE > перед першою появою тега < TR >.

Параметри тегів < COL > і < COLGROUP >.

SPAN визначає кількість суміжних колонок, на які поширюється дія значень параметрів.

ALIGN визначає горизонтальне вирівнювання даних в усіх комірках відповідного стовпця (або стовпців). Допустимими значеннями параметра ALIGN є LEFT , RIGHT і CENTER .

Тег < COLGROUP > додатково дозволяє задавати параметр VALIGN , який визначає вертикальне вирівнювання даних у комірках. Допустимими значеннями параметра VALIGN є MIDDLE , TOP і BOTTOM .

Різниця між тегами <COLGROUP> і <COL> полягає в тому, що перший з них, крім задання параметрів вирівнювання даних для стовпців, виконує умовне об'єднання декількох стовпців у групу. За замовчуванням всі стовпці таблиці вважаються однією групою. Тег <COL> має використовуватися тільки для визначення вирівнювання даних в окремих стовпцях у групі.

Приклад. Необхідно побудувати таблицю, яка містить 5 стовпців, причому дані в перших двох з них мають бути вирівняні вправо, а в інших трьох − посередині.

```
<TABLE>
<COLGROUP SPAN = 2 ALIGN = RIGHT>
<COLGROUP SPAN = 3 ALIGN = CENTER>
дані  для таблиці
</ TABLE >
```

Більш складний приклад. Нехай у таблиці перші дві колонки мають бути вирівняні вправо, а третя −посередині, причому всі три колонки необхідно об'єднати в групу. Наступні дві колонки також мають бути об'єднані в групу, 4-та колонка з вирівнюванням вправо, 5-та − посередині.

```
<TABLE>
<COLGROUP>
< COL SPAN = 2 ALIGN = RIGHT>
```

< COL ALIGN = CENTER>
<COLGROUP>
< COL ALIGN = RIGHT>
< COL ALIGN = CENTER >
дані для таблиці
</ TABLE >

Після тега < COLGROUP > задаються настройки окремих стовпців даної групи. При цьому в тегу < COLGROUP > могли бути задані параметри вирівнювання, значення яких поширюються на всі стовпці даної групи. Значення параметрів, задані в тегу < COL >, скасовують значення з тегом <COLGROUP>.

Задання кольору рамок таблиці

Кілька параметрів дозволяють вибирати колір рамок таблиць − BORDERCOLOR, BORDERCOLORLIGHT і BORDERCOLORDARK . Ці параметри можуть задаватися в тегах < TABLE >, < TD >, < TH > і < TR >. Як значення цих параметрів може використовуватися назва кольору або його шістнадцяткове значення.

Рамка навколо таблиці формується за допомогою трьох кольорів, що дозволяє створити враження об'ємності. Світлий колір визначається параметром BORDERCOLORLIGHT і використовується для освітленого боку, темний колір (BORDERCOLORDARK) − для затіненого боку, а колір заповнення (BORDERCOLOR) визначає заливку "гребеня" рамки.

Теги структурування таблиці

Є ряд додаткових тегів для структурування таблиць і управління прорисовуванням ліній сітки.

За допомогою цих тегів можна виділити комірки заголовка таблиці, основний вміст таблиці і підсумковий рядок.

Теги < THEAD > і < TFOOT > використовуються для опису верхнього і нижнього колонтитулів таблиці. Ці теги можуть траплятися в таблиці не більше одного разу. Завершальний тег для них можна опускати. Використання цих тегів є зручним при створенні великих таблиць, які виходять за межі однієї сторінки.

Тег < TBODY > може траплятися багаторазово в описі таблиці, при цьому потрібне використання завершального тега </ TBODY >. Цей тег виконує логічне групування даних так само, як і тег < COLGROUP > .

Керування прорисовуванням рамок навколо таблиці здійснюється параметром FRAME тега < TABLE >. Значення параметра FRAME :

BOX або BORDER - рамка рисується з усіх чотирьох боків;

ABOVE − тільки зверху;

BELOW − тільки з нижнього боку;

HSIDES − рисується нижній і верхній бік;

VSIDES − рисується лівий і правий бік;

LHS − тільки з лівого боку;

RHS − тільки з правого боку;

VOID − таблиця без зовнішніх рамок.

Параметр RULES керує прорисовуванням внутрішніх ліній сітки таблиці і може набувати таких значень:

ALL − рисуються всі внутрішні лінії;

GROUPS − рисуються тільки лінії, що розділяють групи;

ROWS − рисуються лінії, що розділяють рядки;

COLS − рисуються лінії, що розділяють стовпці;

NONE − внутрішні лінії не рисуються.

Приклад:

< TABLE BORDER FRAME = LHS RULES = GROUPS >

Прорисовування ліній сітки таблиці і рамок буде здійснюватися тільки за наявності параметра BORDER тега <TABLE>. За відсутності цього параметра або його нульового значення лінії сітки і рамки будуть відсутні при будь-яких значеннях параметрів FRAME і RULES .

1.10. Фрейми

Фрейми дозволяють розбити вікно перегляду браузера на кілька областей. У кожну з областей можна завантажити окремий HTML-документ, перегляд якого здійснюється

незалежно від інших. Між фреймами можна організувати взаємодію, яка полягає в тому, що вибір посилання в одному з фреймів може привести до завантаження потрібного документа в інший фрейм або вікно браузера [7].

Типове використання фреймових структур: один фрейм служить змістом документів, а інший використовується для завантаження їх вмісту.

Фрейми дуже схожі на таблиці: і ті і інші розбивають вікно браузера на прямокутні області. Однак за допомогою фреймів можна вирішити не тільки завдання форматування сторінок документа, але й організувати взаємодію між ними. Принципова різниця між фреймами і таблицями полягає в тому, що кожному фрейму має відповідати окремий HTML-документ. Крім того, сторінка, яка відображається у фреймі, може прокручуватися при перегляді незалежно від інших.

Тег <FRAMESET>

Фрейми задаються за допомогою тега FRAMESET, який використовується для сторінок, що містять фрейми, замість розділу BODY звичайного документа. Web-сторінки, що складаються з фреймів, не можуть містити розділ BODY. У свою чергу, сторінки з розділом BODY не можуть використовувати фрейми.

Оскільки для сторінок із фреймами не застосовується розділ BODY, то немає можливості поставити фонове зображення і колір фону для всієї сторінки в цілому. Однак це не заважає в кожен фрейм завантажувати документи, що мають свої параметри фону.

Параметри тега <FRAMESET>: ROWS (рядки) і COLS (стовпці).

<FRAMESET ROWS = "список_значень" COLS = "список_значень">

Можна визначити значення для ROWS або COLS, або обох разом. Необхідно визначити щонайменше два значення хоча б одного з цих параметрів.

Список значень параметрів ROWS і COLS можуть задаватися в пікселях, у відсотках або у відносних одиницях. Число рядків або стовпців визначається числом значень.

Приклад:

<FRAMESET ROWS = "200,150,150">

Задавати значення розмірів фреймів у пікселях не дуже зручно. Браузери запускаються в різних операційних системах і з різними дисплеями.

Кращим варіантом буде встановлення значень у відсотках або у відносних одиницях, наприклад:

<FRAMESET ROWS = "25%, 50%, 25% ">

Якщо сума заданих відсотків не дорівнює 100 %, то значення будуть пропорційно відмасштабовані, щоб у результаті вийшло рівно 100 %.

Значення у відносних одиницях:

<FRAMESET COLS = "*, 2 *, 3 *">

Кожна зірочка є ще однією частиною цілого. Складаючи всі значення чисел, що стоять біля зірочок, отримаємо знаменник дробу. У цьому прикладі перший стовпець займатиме 1/6 частини від загальної ширини вікна, другий стовпець - 2/6, а останній - 3/6.

Можна одночасно використовувати всі три варіанти задання значень.

Контейнер FRAMESET може бути вкладений всередину іншого такого контейнера.

Приклад найпростішого задання фреймів:

```
<HTML>
<HEAD> </ HEAD>
<FRAMESET ROWS = "25%, 75%">
<FRAME SRC = "serg.html">
<FRAME SRC = "john.html">
</ FRAMESET>
</ HTML>
```

Параметр BORDER (FRAMEBORDER) визначає товщину рамок між фреймами в пікселях. При заданні BORDER = 0 рамка між фреймами не буде вирисовуватися.

Параметр FRAMESPACING визначає кількість пікселів між фреймами, що залишаються порожніми.

Тег <FRAME>

Тег <FRAME> визначає одиночний фрейм. Тег <FRAME> не є контейнером.

Необхідно записати стільки тегів <FRAME>, скільки окремих фреймів визначено при заданні тега <FRAMESET>.

Приклад тега <FRAME>:

<FRAME SRC = "serg.html" NAME = "serg" SCROLLING = YES MARGINWIDTH = 5 NORESIZE>

Параметр SRC визначає URL-адресу документа, який буде завантажений у даний фрейм. Без задання цього параметра у фрейм не можна завантажити документ.

Параметр NAME визначає ім'я фрейма, яке може використовуватися для посилання до даного кадру.

Приклад посилання:

<A HREF = "nick. html" TARGET = "serg"> Завантаження у фрейм serg </ A>

Параметр TARGET посилається на ім'я фрейма. Імена фреймів мають починатися з алфавітно-цифрового символа. Імена фреймів порівнюються з урахуванням регістру символів.

Якщо TARGET посилається на ім'я фрейма, якого не існує, то при активізації такого посилання буде утворено нове вікно браузера з цим ім'ям і в нього буде завантажений файл.

Параметри MARGINWIDTH і MARGINHEIGHT встановлюють ширину полів фрейма. Значення встановлюються в пікселях.

Параметр SCROLLING може набувати трьох значень: YES, NO або AUTO. Він керує наявністю або відсутністю смуг прокрутки. За замовчуванням діє AUTO (наявність смуг у разі

потреби). Значення YES викликає появу смуг прокрутки незалежно від потреби у цьому, а NO −забороняє їх появу.

Зазвичай користувач може мишею змінювати розмір фреймів при перегляді сторінки. Це може порушити структуру спроектованих фреймів. Для запобігання зміні розміру фреймів слід скористатися параметром NORESIZE.

Приклад HTML-документа з фреймами:

```
<FRAMESET ROWS = "25%, 50%, 25%">
<FRAME SRC = "header.html">
        <FRAMESET COLS = "25%, 75%">
        <FRAME SRC = "serg1.html">
        <FRAME SRC = "serg2.html">
        </ FRAMESET>
<FRAME SRC = "footer.html">
</ FRAMESET>
```

Взаємодія між фреймами полягає в можливості завантаження документів в обраний фрейм за командами з іншого фрейма. Для цього використовується параметр TARGET тега <A>. Цей параметр визначає ім'я фрейма, в яке буде завантажуватися документ, на який вказує дане посилання. За замовчуванням документ завантажується в поточний фрейм. Це замовчування може бути змінено заданням тега <BASE> з потрібним значенням параметра TARGET.

У розділі <HEAD> доцільно записати:

```
<BASE  TARGET = "serg">
```

Є чотири зарезервованих імені фреймів, при заданні яких виконуються спеціальні дії. Ці імена починаються із символа підкреслення. Будь-яке інше ім'я, яке починається із символа підкреслення, неприпустиме.

TARGET = "_ blank" − забезпечує завантаження документа в нове вікно. Це вікно не матиме імені, а отже, в нього неможливо буде завантажити інший документ.

TARGET = "_ self" − завантаження документа буде проведено в поточний фрейм. Такий запис слід використовувати для обходу замовчування, заданого тегом <BASE>.

TARGET = "_ top" − викликає завантаження документа в повне вікно.

TARGET = "_ parent" − викликає завантаження документа в область, яку займає фрейм-батько поточного фрейма.

Плаваючі фрейми

Тег <IFRAME> реалізує концепцію плаваючих фреймів. На відміну від звичайних фреймів, опис плаваючих фреймів може траплятися в тексті звичайного HTML-документа.

У тегу <IFRAME> застосовуються такі самі параметри, як і в тегу <FRAME>. Немає тільки параметра NORESIZE, застосовувати який немає сенсу, оскільки розмір плаваючих фреймів у будь-якому випадку не може бути змінено при перегляді документа.

Для задання розташування і розмірів плаваючого фрейма в документі можна використовувати такі додаткові параметри: WIDTH, HEIGHT, HSPACE, VSPACE, ALIGN. Їх призначення збігається з відповідними параметрами для вбудованих зображень, які задаються тегом <IMG>.

Приклад:

<IFRAME SRC = serg.html NAME = "A" HEIGHT = 200 WIDT H = 50% HSPACE = 10 SCROLLING = YES ALIGN = RIGHT>

Концепція плаваючих фреймів близька до концепції зображень. У потрібне місце HTML-документа цілком вбудовується інший HTML-документ [8].

1.11. Діалогові форми

Діалогові форми призначені для пересилання даних від віддаленого користувача до Web-сервера. За їхньою допомогою можна організувати найпростіший діалог між користувачем і сервером, наприклад, реєстрацію користувача на сервері або вибір потрібної відповіді з поданого списку.

Тег <FORM>

У HTML-документі для задання форми використовуються теги <FORM>, що є тегами-контейнерами. Документ може

містити декілька форм, але вони не можуть бути вкладені одна в одну.

Приклад:

<FORM METHOD = "POST" ACTION = "/ cgi-bin / data" ENCTYPE = "Multipart / form-data">

</ FORM>

Параметри тега <FORM>

Параметр ACTION є обов'язковим. Його значенням є URL-адреса CGI-програми, яка буде обробляти інформацію, витягнуту з даної форми.

Інший варіант: ACTION = "mailto: serg@i.ua "

Параметр METHOD визначає метод пересилання даних, що містяться у формі, від браузера до Web-сервера. Він може набувати двох значень: GET (за замовчуванням) і POST.

Взаємодія між клієнтом-браузером і Web-сервером здійснюється за правилами, заданими протоколом HTTP, і складається із запитів клієнта і відповідей сервера. Запит розбивається на три частини. У першому рядку запиту міститься HTTP-команда, що називається методом, URL-адреса запитуваного файлу і номер версії протоколу HTTP. Друга частина − заголовок запиту. Третя частина − тіло запиту, дані, що посилаються із сервера.

Метод повідомляє серверу про мету запиту. У протоколі HTTP визначено кілька методів. Для передачі даних форми в CGI-програму використовуються два методи: GET і POST.

При використанні методу GET дані форми пересилаються в першому рядку. Ці дані розташовуються після символа "?" у вигляді сукупності пар змінна = значення, розділених символом "&". Приклад першого рядка запиту:

GET / cgi-bin / data? Name = serg & surname = sergienko

Після виділення даних з URL сервер привласнює їх змінній QUERY _ STRING, яка може бути використана CGI-програмою.

При використанні методу POST дані форми пересилаються з Web-сервера в тілі запиту, після чого передаються в CGI-програму через стандартне введення.

Методи GET і POST мають свої переваги і недоліки. Метод GET забезпечує кращу продуктивність при пересиланні форм, що складаються з невеликого набору коротких полів. При пересиланні великого обсягу даних слід використовувати метод POST, оскільки браузер або сервер можуть накладати обмеження на розмір даних, переданих у складі URL, і відкидати частину даних, що виходить за межу. Метод POST є більш надійним при пересиланні конфіденційної інформації [9].

Значенням параметра ENCTYPE є медіа-тип, який визначає формат кодування даних при передачі їх від браузера до сервера. Можливі два значення цього параметра: application / x-www-form-urlencoded (за замовчуванням) і multipart / form - data.

Стандарт MIME (Multipurpose Internet Mail Extensions, багатоцільові розширення електронної пошти для Інтернету) визначає набір MIME-типів, що відповідають різним типам даних, і правила їх пересилання електронною поштою. Для позначення MIME-типу використовується запис вигляду тип / підтип, де тип визначає загальний тип даних, наприклад, text, image, application (тип application позначає специфічний внутрішній формат даних, який використовується деякою програмою), підтип − конкретний формат всередині типу даних, наприклад, image / gif, text / html.

У Web MIME-типи називаються також медіа-типами і використовуються для ідентифікації формату документів, переданих по протоколу HTTP. У HTML-формі параметр ENCTYPE визначає медіа-тип, який використовується для кодування і пересилання спеціального типу даних.

Форма відображається у вікні браузера у вигляді набору стандартних елементів керування, які використовуються для заповнення полів форми значеннями, які потім передаються Web-серверу. Значення вводиться в поле введення користувачем або призначається за замовчуванням. Для створення полів засобами мови HTML існують спеціальні теги, які вживаються тільки всередині тега <FORM>.

Тег <INPUT>

За допомогою цього тега можна створювати всередині форми поля для введення рядка тексту, пароля, імені файлу,

кнопки. Він має два обов'язкових параметри: TYPE і NAME. Параметр TYPE визначає тип поля: введення рядка тексту, пароля та ін. Параметр NAME визначає ім'я, що привласнюється полю. Воно використовується як ідентифікатор значення, переданого Web-серверу. Інші параметри міняються залежно від типу поля.

Розглянемо типи полів, що створюються за допомогою тега <INPUT>.

Рядок введення тексту

TYPE = TEXT

Додаткові параметри:

MAXLENGTH задає максимальну кількість символів, дозволених у текстовому полі. За замовчуванням вона не обмежена.

SIZE задає максимальну кількість відображуваних символів.

VALUE задає початкове значення текстового поля.

Приклад:

<INPUT TYPE = text NAME = user VALUE = Serg SIZE = 25 MAXLENGTH = 45>

Рядок введення пароля

TYPE = PASSWORD

Відрізняється від попереднього тим, що всі символи, що вводяться, замінюються на екрані символом *.

Поле PASSWORD не гарантує безпеки введеного тексту, бо на сервер він передається в незашифрованому вигляді.

Параметр VALUE може бути заданий, хоча це і не має сенсу.

Поле для введення імені локального файлу

TYPE = FILE

Це поле супроводжується кнопкою Browse. Обраний файл приєднується до вмісту форми при пересиланні на сервер. Файл можна ввести безпосередньо або, скориставшись кнопкою Browse, вибрати його з діалогового вікна.

Для коректної передачі приєднаного файлу слід встановити значення параметрів форми рівними ENCTYPE = "multipart / form - data" і METHOD = POST. У противному разі буде переданий введений рядок, тобто ім'я файлу, а не його вміст. Розширені можливості пошуку MAXLENGTH і SIZE мають те саме значення, що і для елементів TEXT і PASSWORD.

Прапорець

TYPI = CHECKBOX

Створює поле для установлення прапорця, який можна встановити або скинути. Елементи CHECKBOX можна об'єднати в групу, встановивши однакове значення параметра NAME. Додаткові параметри:

VALUE = рядок

Значення, яке буде передано із сервера, якщо цю кнопку вибрано. Якщо кнопку не вибрано, значення не передається. Це обов’язковий параметр.

CHECKED

Якщо вказано параметр CHECKED, елемент є вибраним за замовчуванням.

Якщо прапорці утворюють групу, то переданим значенням є рядок розділених комами значень параметра VALUE всіх установлених прапорців.

Радіокнопки

TYPE = RADIO

Створює радіокнопки, з яких може бути вибрано тільки одну. Всі елементи групи повинні мати однакове значення параметра NAME. Відображаються у вигляді круглої кнопки. Додаткові параметри:

VALUE = рядок

Обов'язковий параметр, значення якого передається серверу при виборі даної кнопки. Повинен мати унікальне значення для кожного члена групи.

CHECKED

Встановлює елемент таким, що вибрано за замовчуванням. Тільки один елемент у групі повинен мати цей параметр.

Кнопка передачі

TYPE = SUBMIT

Створює кнопку передачі, натискання якої викликає пересилку на сервер усього вмісту форми. Відображається у вигляді прямокутної кнопки з написом Submit.

Додатковий параметр.

VALUE = назва кнопки

Дозволяє змінити напис на кнопці. Параметр NAME для даного елемента може бути опущений. У цьому випадку значення кнопки не включається у список параметрів форми і не передається на сервер.

Всередині форми можуть існувати кілька кнопок передачі.

Кнопка скидання

TYPE = RESET

Створює кнопку скидання, натискання якої скасовує всі зроблені зміни, відновлюючи значення полів форми на той момент, коли вона була завантажена. Відображається у вигляді прямокутної кнопки з написом Reset. Напис можна змінити за допомогою додаткового параметра.

VALUE = назва кнопки

Значення кнопки RESET ніколи не пересилається на сервер, тому у неї відсутній параметр NAME.

Кнопка з графічним зображенням

TYPE = IMAGE

Створює кнопку з графічним зображенням, що діє аналогічно до кнопки Submit. Додаткові параметри:

SRC = URL зображення задає URL-адресу файлу з графічним зображенням елемента.

ALIGN задає тип вирівнювання зображення щодо поточного рядка тексту.

Прихований елемент

TYPE = HIDDEN

Створює прихований елемент, що не відображається користувачеві. Інформація, що зберігається в прихованому полі, завжди пересилається на сервер і не може бути змінена ні користувачем, ні браузером. Приховане поле можна

використовувати, наприклад, у такому випадку. Користувач заповнює форму реєстрації та відправляє її серверу. При повторному вході на сервер потрібно заповнити другу форму, яка частково використовує інформацію, що міститься в першій формі. Сервер не зберігає історію діалогу з користувачем, він обробляє кожен запит незалежно і при отриманні другої форми не знатиме, як вона пов'язана з першою. Щоб повторно не вводити вже введену інформацію, можна задати в CGI-програмі, яка обробляє першу форму, перенесення необхідних даних у приховані поля другої форми. Вони не будуть видимі користувачем і водночас доступними із сервера.

Значення прихованого поля визначається параметром VALUE.

Меню, що випадає

Тег <SELECT> призначений для того, щоб організувати всередині форми меню, що випадає. Тег має такі параметри.

NAME − обов'язковий параметр. При виборі одного або декількох елементів формується список обраних значень, який передається на сервер під ім'ям NAME.

SIZE = n встановлює число одночасно видимих елементів вибору. Якщо значення параметра дорівнює 1, то відображається меню, що випадає, якщо більше 1, то відображається список прокрутки з n одночасно видимими елементами.

MULTIPLE означає, що з меню або списку можна вибрати одночасно кілька елементів.

Елементи меню задаються всередині тега <SELECT> за допомогою тега <OPTION>.

Параметр VALUE містить значення, яке пересилається серверу, якщо даний елемент вибраний з меню або списку. Якщо значення цього параметра не задано, то за замовчуванням воно встановлюється рівним змісту тега <OPTION>.

Параметр SELECTED відображає елемент як обраний.

Приклад:

```
<SELECT NAME = "var-name">
<OPTION VALUE = "Red"> Red
<OPTION SELECTED> Green
</ SELECT>
```

Область введення тексту

<TEXTAREA NAME = serg ROWS = 5 COLS = 10>

текст

</ TEXTAREA>

Тег <TEXTAREA> створює усередині форми поле для введення багаторядкового тексту, відображається у вікні браузера у вигляді прямокутної області з горизонтальною і вертикальною смугами прокрутки. Для пересилання на сервер кожний введений рядок доповнюється символами %0D%0A (ASCII-символи "Повернення каретки" і "Переведення рядка" з попереднім символом %), отримані рядки об'єднуються в один рядок, який і відправляється на сервер під ім'ям, заданим параметром NAME. Параметри:

NAME

Ім'я, під яким дані відправлені на сервер.

COLS задає число стовпців видимого тексту.

ROWS задає число рядків видимого тексту.

Між тегами < TEXTAREA > і </ TEXTAREA > можна помістити текст, який буде відображатися за замовчуванням.

Завдання і контрольні запитання

1. У чому суть гіпертекстової форми подання WWW-документів?
2. Особливості синтаксису мови гіпертекстових посилань (HTML).
3. Які групи директив HTML ви знаєте?
4. Як на основі HTML відображаються заголовки, шрифти, списки, заздалегідь відформатований текст, гіперпосилання?
5. Створити особисту Web-сторінку з такими елементами:

- вказати ключові слова для пошукових систем;
- визначити колір фону, вигляд фонового зображення, колір тексту і посилань, поля документа;
- застосувати теги логічного і фізичного форматування;
- задати шрифт для деяких ділянок тексту і шрифт за замовчуванням;
- здійснити вирівнювання абзаців різного виду;
- задати заголовки різних рівнів;
- в тексті провести горизонтальні лінії різної товщини і кольору;

- задати попередньо відформатований текст;

- задати посилання трьох типів: всередині поточного документа, на інший документ і на певне місце іншого документа.

6. Створити таблицю, що складається з 5 рядків і 4 стовпців. Вирівнювання даних: у 1-му і 4-му рядках – по центру, у 2-му і 3-му – направо. Лінії зовнішньої рамки мають бути наявними з усіх боків, внутрішня рамка − між 2-м і 3-м стовпцем, між 1-м і 2-м рядком, 4-м і 5-м рядком .

7. Включити в документ діалогові елементи: рядок введення тексту, рядок введення пароля, поле для введення імені файлу, прапорці, радіокнопки, кнопку передачі, кнопку скидання, кнопку з графічним зображенням.

2. Динамічний HTML

2.1. Можливості динамічного HTML

Динамічний HTML − це HTML-сторінки з динамічно змінним вмістом.

Динамічний HTML об'єднує: безпосередньо HTML, таблиці стилів (Cascade Style Sheets − CSS) і мови сценаріїв JavaScr ipt або VBScript. Ці три компоненти DHTML пов'язані об'єктною моделлю документа.

За допомогою каскадних таблиць стилів визначається зовнішній вигляд HTML-документа: колір шрифту і фону документа, тип шрифту і багато іншого. Для кожного тега HTML можна визначити свій стиль відображення. Наприклад, заголовки першого рівня будуть відображатися шрифтом Arial 14 пунктів синього кольору, основний текст − Times New Roman 10 пунктів чорного кольору з одинарним інтервалом між рядками. Можна створити таблицю стилів і використовувати її для всіх документів, розташованих на сервері, що додасть однаковість всьому Web-сайту.

Можливості динамічного HTML [10]:

1. Усі елементи сторінки (теги, графіка, текст та ін.) доступні для огляду і керування.

Об'єктна модель документа робить всі елементи сторінки програмованими об'єктами. За її допомогою через мови сценаріїв можна отримати доступ і керувати всім, що є в документі. Кожен елемент HTML доступний як об'єкт, і, як наслідок, документ стає динамічним. Будь-яка дія користувача (клацання кнопкою миші, переміщення миші у вікні браузера, натискання клавіші клавіатури) об'єктною моделлю документа трактується як подія, яку можна обробити сценарієм.

Різні розробники браузерів можуть реалізовувати власну об'єктну модель документів. Тому розробникам динамічних сторінок доводиться писати кілька варіантів своїх додатків для всіх браузерів окремо.

2. Абсолютне позиціонування елементів, включаючи третю координату (z-індекс), і поява 2,5-вимірності сторінки.

Одним з недоліків HTML була складність керування параметрами сторінки. Текст відображався шрифтом і кольором, визначеними за замовчуванням, і його положення у вікні броузера не було визначено. Можна було визначити тільки положення початку і кінця абзацу, а також вирівнювання графіки та інших елементів: по лівому, по правому краю або по центру сторінки.

Проблемою було те, що, коли користувач змінював розміри вікна перегляду, все знову починало рухатися. Почасти допомагало вирішити цю проблему застосування таблиць і фреймів. У динамічному HTML розмір і положення елементів у вікні броузера може бути таким, як захоче автор. Вони можуть перетинатися, що досягається деякими способами використання таблиць стилів.

3. Динамічне перерисовування будь-якої частини сторінки дозволяє контролювати процес зміни. Не потрібно повністю оновлювати сторінку для появи її зміненого вигляду.

Браузер може перерисувати всю сторінку або її частину, використовуючи збережені копії елементів, а не вихідний текст сторінки.

4. Графічні фільтри дозволяють додавати ефекти мультимедіа − вертикальне чи горизонтальне відображення картинки, створення рядка, що біжить, рух плям по картинці та ін.

Існує кілька елементів керування мультимедіа, два з них доступні за допомогою властивостей каскадної таблиці стилів – це Visual Filter і Transitions Filter. Це дає змогу додавати до сторінки безліч ефектів, які раніше були доступні хіба що в графічних пакетах, таких як CorelDraw.

2.2. Таблиці стилів

Таблиця стилів – це опис правил, які задають параметри подання окремих елементів на екрані мовою HTML.

Переваги таблиць стилів [11]:

1. Універсальність застосування. Можна створити таблицю стилів і застосувати її до документа або групи документів, встановлюючи в них посилання на неї. Тим самим можна змінювати вигляд усіх сторінок, змінюючи таблицю стилів.

2. Можна керувати текстом у більшій мірі, ніж зазвичай. Існує ряд властивостей, за допомогою яких можна накладати один текст на інший, відкидати тінь та ін.

3. Таблиці стилів дозволяють зберігати в різних місцях текст і інформацію про те, якого вигляду він має бути. Це дає змогу зменшити розміри файлів і полегшує написання сторінок, зменшуючи «концентрацію» тегів.

Оголошення CSS записуються так: тег {характеристика: величина}

Приклад:

H1 {color: blue}

Можна привласнити одну властивість кільком тегам, групуючи ці теги в селекторі присвоювання.

Приклад:

H1, H2, H4 {color: black}

Так само, як і теги, можна групувати властивості. Потрібно узяти ці визначення в дужки, розділивши крапкою з комою:

```
H2 {color: red;
font-size: 12pt;
font - family: Arial  }
```

CSS надає можливість одному тегу успадковувати властивості іншого. Це означає, що не потрібно описувати властивості всіх можливих тегів. Якщо ми нехтуємо заданням властивостей для тега <EM>, він успадкує властивості тегів, в які він вкладений.

Приклад:

<НЗ> Таблиці <EM> листів / EM> стилів </ НЗ>

Якщо таблиця стилю для всіх заголовків <НЗ> визначає зелений колір, але нічого не говорить про <EM>, тоді слово «листів» буде зеленим, як і всі інші. Якщо спеціально поставити для <EM> блакитний колір, слово буде блакитним. Зазвичай параметри за замовчуванням задаються за допомогою тега <BODY> і успадковуються всіма іншими.

Можна задати властивість, що є функцією від батьківської:

P {font-size: 14pt}

P {line-height: 120%}

У цьому випадку висота рядка визначена відношенням до розміру шрифту. Це зручно, коли потрібно мати можливість перевизначити стилі пізніше. При зміні розміру шрифту в такому разі висоту рядка буде автоматично змінено.

Можна також задати колір <EM> червоним у випадках розташування цієї частини тексту між тегами <НЗ> і </ НЗ> − і тільки в них.

НЗ EM {color: red }

Тут таблиця стилів визначає, що будь-який зміст <EM>, вкладений у теги <НЗ>, буде показано жовтим. Це не позначається на стилі <EM> в інших місцях сторінки.

Можна визначити значення для різних змінних в одному рядку, розділивши змінні комами.

Приклад: НЗ EM, Н2 I {color: yellow}

Існує можливість одночасного використання різних таблиць стилів для одного документа.

У кожного браузера є свій стандартний стиль. Якій таблиці стилів буде віддано перевагу, задається деякою системою правил,

і іноді «виграє» стиль автора, іноді − стиль браузера за замовчуванням. Вибір із запропонованих стилів ґрунтується на таких правилах:

1. Визначається, чи немає протиріч у заданні параметрів елемента. Якщо є, то використовуються батьківські параметри. Якщо немає їх, то будуть використовуватися установки, задані за замовчуванням.

2. Якщо є конфлікт між авторським стилем і стилем, визначеним за замовчуванням, перевага віддається авторським параметрам.

3. Якщо конфліктують два стилі і один застосовується тільки в цій ситуації, а інший − в усіх випадках, перевага віддається першому.

Така система пріоритетів дозволяє одному документу посилатися на кілька таблиць стилів. Це дає змогу створювати універсальні таблиці стилів, щоб потім їх застосовувати у різноманітних комбінаціях.

Як використовувати таблиці стилів у документі HTML? Є кілька способів.

Для використання листів стилів існує кілька способів.

Перший з них називається приєднанням (зв'язуванням) зовнішніх листів стилів і полягає у визначенні правил стилів в окремому файлі з розширенням *.css і створенні зв'язку нашого HTML-документа з цим файлом за допомогою тега <LINK>. Тег <LINK> міститься в заголовку документа і використовується для визначення зв'язків між документами. Він має кілька атрибутів − HREF, REV, REL, TYPE,− які можуть набувати різних значень. Найбільш часто використовуються взаємозв'язки між документами, розташованими в певному порядку − next (наступний) і previous (попередній). Приклад тега <LINK> буде мати такий вигляд:

```
<HTML>
<HEAD>
<TITLE> Приклад документа </ TITLE>
<LINK REL = stylesheet HREF = "http://www.serg.com/serg.css" TYPE = "text/css">
<BODY>
```

Другий спосіб називається впровадженням листів стилів і полягає у визначенні стилів всередині HTML-документа за допомогою спеціальних тегів <STYLE>. Цей контейнер також міститься в заголовку документа:

```
<HTML>
<HEAD>
<TITLE> Приклад документа </ TITLE>
<STYLE TYPE = "text / css">
H1 {color: red}
</ STYLE>
<BODY>
```

Третім способом є імпортування листів стилів, яке виконується в такий спосіб:

```
<HTML>
<HEAD>
<TITLE> Приклад документа </ TITLE>
<STYLE TYPE = "text / css">
@ import url (http: // www. serg.com/serg.css);
</ STYLE>
<BODY>
```

2.3. Позиціонування елементів на Web-сторінці

У DHTML можна задати координати x, y і z кожного елемента сторінки, забезпечуючи тривимірність зображення і задаючи можливі перекриття елементів.

Властивість left (x-координата) використовується для задання в пікселях відстані елемента від лівого краю вікна.

Властивість top (y-координата) задає відстань у пікселях до елемента від верхнього краю вікна.

Властивість z-index (z-індекс) вказує, в якому порядку елементи будуть перекривати один одного. Елемент з більш високим z-індексом буде з'являтися над елементами з більш низьким.

Так на сторінках створюється подоба тривимірного простору. Насправді це лише плоскі шари, для яких задається порядок перекривання. Тому таке нововведення названо 2,5-вимірністю.

Основний текст має нульовий рівень. Це означає, що його z-індекс дорівнює нулю. Заданням додатного z-індексу ми поміщаємо елемент поверх тексту, від'ємного − під текстом. Можна задавати відносний рівень різних елементів за допомогою визначення для них різних значень z-індексу.

Браузер задає z-індекс елементів автоматично, якщо він не заданий спеціально. Коли у двох елементів один рівень, як, наприклад, у двох графічних тегів, то елементу, який розташовується в тексті пізніше, автоматично присвоюється більше значення z-індексу. Отже, наступний за текстом елемент з'явиться поверх попереднього.

Властивість position використовується для визначення місця, в якому браузер розмістить елемент.

Елементи HTML відображаються браузером послідовно, в тому порядку, як вони визначені в тексті HTML-файлу з урахуванням розташування попередніх відображених елементів і елементів-контейнерів, в яких вони можуть міститися. При компонуванні сторінки використовуються установки браузера для визначення положення кожного елемента. Наприклад, два послідовних абзаци йдуть один за одним, причому кожен починається з нового рядка.

Властивість position елемента визначає спосіб його позиціонування на сторінці: статичний, відносний або абсолютний. Значення властивості − static, relative, absolute.

Абсолютно позиціонований елемент вилучається з потоку відображення елементів і позиціонується незалежно, причому елемент може перекривати раніше відображені елементи.

Щоб визначити точку відліку розташування елемента, слід знайти його найближчого батька, позиціонованого абсолютно. Положення лівої верхньої вершини блоку цього елемента і буде точкою відліку для абсолютно позиціонованого елемента. Якщо процес пошуку батька дійде до елемента <BODY>, то тіло документа і буде тим елементом, відносно якого позиціонується вихідний елемент.

Зміна розмірів вікна браузера не змінить становище абсолютно позиціонованого елемента. Тому при зменшенні розмірів вікна такий елемент може виявитися за межами вікна.

Властивості width і height задають ширину і висоту блоку елемента. Якщо вони не задані, то по горизонталі блок поширюється до правого краю вікна браузера, а по вертикалі − на стільки, на скільки необхідно для відображення елемента. Якщо елемент не поміщається в блок заданого розміру, то його частина не буде видимою користувачу.

Після вилучення з початкового тексту документа всіх абсолютно позиціонованих елементів утворюють безперервний потік зображення, в якому кожний наступний елемент позиціонується відносно кінця попереднього.

Якщо для будь-якого елемента визначено значення властивостей top і left, то цей елемент зміщується вниз і вправо на задані величини відносно правого верхнього кута блоку попереднього елемента в потоці.

Статично позиціоновані елементи поводяться аналогічно до позиціонованих. Відмінність полягає в тому, що для них не можна встановити значення властивостей top і left і змістити їх, наприклад, з рядка абзацу вгору або вниз. За замовчуванням елементи будуть позиціоновані статично.

Приклад:

<DIV STYLE = "position: absolute; top: 200; left: 0; width: 400">

<H3> текст </ H3>

<IMG SRC = "serg.gif" STYLE = "position: absolute; top: 0; 1eft: 50; z-index: -l">

</ DIV>

2.4. Фільтри

Фільтри −це мультимедійні ефекти для зміни зовнішнього вигляду графіки і тексту на Web-сторінці. Ці ефекти підтримуються за допомогою каскадної таблиці стилів. Візуальні ефекти різноманітні: поступовий "прояв" зображення або тексту, зміна контрастності графічного зображення, "світіння" букв тексту та ін.

Доступні два види фільтрів. Перший −статичні фільтри. Вони змінюють зовнішній вигляд об'єкта (тексту або графіки), і ця картинка залишається нерухомою. Друга група фільтрів − це

динамічні фільтри. Вони дозволяють змінювати графічний елемент динамічно, із заданою користувачем швидкістю.

Існує кілька статичних фільтрів, які задаються таким чином:

filter: ім’я_фільтра ([параметри]);

Приклад застосування фільтра fliph, який перевертає картинку навколо вертикальної осі.

```
<STYLE>
.effect {filter: fliph ()}
</ STYLE>
...
<IMG CLASS = effect SRC = serg.gif>
```

Фільтри застосовуються не до всіх HTML-елементів, а тільки до тих, які містяться в прямокутному блоці і не є вікнами, як, наприклад, елемент <IFRAME>. Фільтри можуть застосовуватися до тегів BODY, IMG, INPUT, MARQUEE, TABLE, TD, TEXTAREA. До елемента DIV фільтр можна застосувати, якщо задані ширина, висота або елемент абсолютно позиціонується.

Фільтри не застосовуються до таких елементів HTML: IFRAME, SELECT, OPTION, P, EM, STRONG і до всіх заголовків H1, H2 та ін.

До елемента можна застосувати кілька фільтрів одночасно.

Приклад:

```
<style>
.effect {filter: blur (strength = 50) fliph ()}
</ style>
```

До графічного зображення застосовуються два фільтри. Перший (blur) розмазує зображення на глибину 50 пікселів, а другий (fliph) дзеркально його відображає в горизонтальному напрямку.

Завдання і контрольні запитання

1. Додати пов'язану таблицю стилів, в якій визначаються стилі заголовків H1-H6 і тега <P>.

2. Додати впроваджену таблицю стилів, в якій визначаються стилі тегів логічного форматування.

3. Імпортувати таблицю стилів, в якій визначаються стилі списків: встановити графічне зображення як маркер невпорядкованого списку; встановити нумерацію впорядкованого списку римськими цифрами.

4. Вбудувати визначення стилів у теги деяких абзаців і елементів логічного форматування.

5. Задати класи і підкласи властивостей для тегів <CITE> і <SAMP>.

6. Визначити стилі за допомогою ідентифікаторів.

7. Виконати накладення зображення і тексту в декількох рівнях.

8. Обрізати зображення. Задати стиснення зображення і додати до нього смуги прокрутки.

9. Застосувати до різних зображень і ділянок тексту 3 фільтри.

10. Які Ви знаєте способи підключення листів стилів?

11. Пріоритети правил CSS.

12. Які властивості керують видимістю елементів на сторінці?

13. Для чого призначені фільтри?

3. ПРОГРАМУВАННЯ МОВОЮ JAVASCRIPT

3.1. Призначення мови JavaScript

JavaScript − об'єктно-орієнтована інтерпретована мова програмування, призначена для створення сценаріїв, які динамічно змінюють зовнішній вигляд Web-сторінки. Браузер виконує кожний рядок сценарію послідовно, після виконання попереднього. Зміну сценарію JavaScript провести так само просто, як і редагування звичайного документа HTML. Усі проведені зміни набувають чинності безпосередньо після завантаження Web-сторінки у вікні браузера.

За допомогою JavaScript вирішуються такі завдання:

− відображення повідомлень (як на самій сторінці, так і у вигляді діалогового вікна);

– перевірка достовірності заповнених полів форм HTML до передачі їх на сервер;

– визначення використовуваного браузера і налаштування відповідно до нього Web-сторінки;

– створення динамічних HTM-сторінок спільно з каскадними таблицями стилів і об'єктною моделлю документа.

JavaScript розроблена компанією Netscape. JavaScript – це перша розроблена мова підготовки сценаріїв, що посідає сьогодні одне з перших місць за популярністю.

Мова JavaScript відповідає стандартам ЄСМА (Європейської асоціації стандартизації). Метою організації є розроблення специфікацій, яким мають задовольняти мови програмування.

Хоча синтаксис JavaScript схожий на синтаксис Java, по суті ці мови програмування різні. Їх відмінності:

– апплети Java компілюються в байт-коди, потім інтерпретуються Web-браузером. У JavaScript використовуються прості текстові команди, які безпосередньо вставляються в документ HTML;

– JavaScript застосовується для створення простих додатків і додавання інтерактивних елементів на Web-сторінку, а Java розрахований на розроблення складних додатків;

– JavaScript швидша за Java при вирішенні простих завдань, оскільки при запуску апплета Java завжди виникає затримка, пов'язана з інтерпретацією байт-кодів.

Для того щоб додати сценарій JavaScript на Web-сторінку, використовують тег <SCRIPT>.

<SCRIPT LANGUAGE = "JavaScript">

...

</ SCRIPT>

Існує чотири способи розміщення сценаріїв:

1. У тілі документа <BODY>. У цьому випадку результат сценарію відображається на Web-сторінці при її завантаженні в браузері.

2. У заголовку документа між тегами <HEAD>. Сценарій, розміщений у заголовку, не виконується одразу ж при завантаженні сторінки, а використовується іншими сценаріями як функція.

3. У тегу HTML, де можна розмістити обробник подій , що дозволяє виконувати сценарій JavaScript разом з тегом. Обробник подій є окремим типом сценарію, який не потребує використання тега <SCRIPT> для його позначення.

4. В окремому файлі. JavaScript дозволяє створювати власні файли з розширенням *. js. Такий файл не повинен містити тегів HTML. Зв'язується з документом таким чином:

<SCRIPT LANGUAGE = "JavaScript" SRC = serg. js>
оператори
</ SCRIPT>

Оператори всередині тега виконуються, якщо сталася помилка при завантаженні файлу.

3.2. Синтаксис JavaScript

JavaScript не потребує обов'язкового оголошення змінних. Повідомляти змінні можна за допомогою ключового слова var. Існує два типи змінних: глобальні змінні, якими можна користуватись в усіх сценаріях документа HTML, і локальні змінні, якими можна користуватись тільки як аргументами однієї функції. Вони застосовуються тільки в тій функції, в якій були створені.

Щоб створити глобальну змінну, потрібно оголосити її в головному сценарії, а не у функції. Приклад:

var st = 25;

Якщо цей оператор використовувати поза функцією, то буде створена глобальна змінна, а якщо всередині, то локальна.

У JavaScript не треба при оголошенні змінних визначати їх тип даних. Він розуміється за контекстом. Типи даних: числовий, логічний, рядковий.

Можна змінити тип даних змінної.

Приклад:

t = 35;
t = "Serg";

Тепер змінній t визначено рядковий тип даних. При виконанні останнього оператора повідомлення про помилку не виводиться.

Для перетворення даних у JavaScript використовуються дві функції:

- parseInt(). Перетворює текстовий тип даних у цілочисельний;

- parseFloat(). Перетворює текстовий тип даних у числовий з плаваючою точкою.

Обидві функції зчитують число у вигляді тексту і перетворюють його в числовий тип даних. Числа шукаються тільки на початку рядка тексту. Нечислова частина ігнорується.

Приклад:

```
total = parseInt ( "10 дисплеїв");
```

У JavaScript текст зберігається в об'єктах String. Існує два способи створення об'єктів String:

```
t = "Serg";
t = new  String ( "Serg");
```

Другий оператор містить команду new, використовувану для створення об'єктів. Цей оператор дає вказівку браузеру створити рядковий об'єкт, що містить текст "Serg" і присвоїти його змінній t.

Кількість символів, що міститься в рядковій змінній, визначається властивістю length об'єкта String. Наприклад, властивість t.length визначає довжину значення об'єкта t.

Приклад:

```
document.write (t.length);
```

Для зміни регістру символів тексту об'єкта String використовуються два методи.

toUpperCase(). Перетворює символи тексту у великі.

toLowerCase(). Перетворює символи тексту у малі.

Приклад:

```
document.  write (t. toLowerCase ());
```

З'явиться повідомлення:

serg

При цьому значення змінної t залишається колишнім. Якщо потрібно змінити значення рядкової змінної, використовується такий оператор:

```
t = t.  toLowerCase ();
```

Оголошення масиву:

sc = new Агга (30);

Індексування елементів масиву починається з 0, тому елементи масиву мають індекси 0 − 29. Звернення до елемента масиву:

sc [0] = 39;

Подібно до рядкових змінних, масиви мають властивість length. Вона визначає кількість елементів, з яких складається масив.

Методи об'єкта Array:

concat − об'єднує два масиви в один;

join − з'єднує всі елементи масиву в один рядок;

pop − видаляє останній елемент з масиву і повертає його значення;

push − додає один або кілька елементів у кінець масиву і повертає останній доданий елемент;

reverse − переставляє елементи масиву у зворотному порядку;

shift − видаляє перший елемент масиву і повертає його значення;

sort − сортує елементи масиву;

unshift − додає один або більше елементів у початок масиву і повертає нову довжину масиву.

Умовні оператори

Приклад:

if (phone = = "") window. alert ("Помилка!") else window. alert ("ОК!");

Умовний вираз може мати таку конструкцію:

Змінна = (умова)? якщо виконується: якщо не виконується;

Цей вислів дозволяє визначити змінній одне з двох значень. Одне в разі виконання умови, друге − при її невиконанні.

Приклад:

e = (a = = 1)? 1: 0;

Перемикач switch

Приклад. Фрагмент сценарію, що викликає в браузер одну з трьох сторінок.

```
w = window. prompt ( "Введіть ім'я фірми");
switch (w) {
case "Netscape":
window. location = "www. netscape.com";
break;
case "Intel":
window.location = "www.intel. com ";
break;
default:
window.location = "www.microsoft. com ";
}
```

3.3. Обробка подій

Події − це дії, що відбуваються в браузері, − клацання мишею, натискання клавіш та ін.

Сценарії, які дозволяють визначати події і виконувати відповідні їм дії, називаються оброблювачами подій. Оброблювач подій − це найпотужніший засіб у JavaScript. Одночасно він є і найпростішим у використанні засобом. Для використання обробника подій, як правило, задається лише один рядок програмного коду.

Кожна подія має власне ім'я. Наприклад, подія onMouseOver відбувається при наведенні покажчика миші на об'єкт Web-сторінки. При цьому подія onMouseOver активізується і запускається обробник цієї події.

Перелік обробників подій:

– onAbort − користувач відмовляється від завантаження зображення. Застосовується до об'єкта image;

– onChange − користувач змінює значення елемента. Застосовується до text, textarea, select;

– onClick−клацання на об'єкті. Застосовується до button, radio, checkbox, submit, link;

– onError − помилка при завантаженні документа або зображення. Застосовується до image, window;

– onKeyDown − натискання клавіші. Застосовується до document, link, image, textarea;

– onKeyPress − натискання і утримання клавіші;

– onKeyUp − відпускання клавіші;

– onLoad − завантаження документа. Застосовується до body;

– onUnLoad − закриття документа. Застосовується до body;

– onMouseDown−натискання кнопки миші. Застосовується до document, button, link;

– onMouseUp − відпускання кнопки миші;

– onMouseOver − наведення миші на об'єкт. Застосовується до link;

– onMouseOut − зняття миші з об'єкта. Застосовується до link;

– onMove − переміщення вікна;

– onReset − натискання кнопки Reset форми;

– onResize − зміна розмірів вікна. Застосовується до window;

– onSelect− користувач вибирає поле введення елемента форми. Застосовується до text, textarea.

Для визначення події не потрібно використовувати тег <SCRIPT>. Обробник подій вставляється в тег HTML. Ось як задається обробник подій onMouseOver:

<A HREF = "serg. Html"

onMouseOver = "window. alert ('Ви вказуєте на посилання');"> Клацніть тут </ A>

Для виділення тексту повідомлення використані одинарні лапки. Це тому, що подвійними лапками виділений обробник подій.

У простих обробниках подій можна використовувати кілька операторів, розділивши їх крапкою з комою.

Якщо необхідно задати в обробнику подій більш ніж один оператор, тоді краще створити додаткову функцію. Потрібно визначити функцію в заголовку документа, а потім викликати її замість обробника подій:

<A HREF = "# abzac onMouseOver = "Nick ();"> Наведіть покажчик на посилання </ A>

Використання функції є кращим, оскільки в ній можна ставити більш складні оператори, ніж в обробнику подій.

Найбільш використовуваний обробник подій − onMouseOver. OnMouseOut − це протилежний обробник. Він викликається при видаленні покажчика миші з потрібної області.

За допомогою обробника onMouseOver можна, при наведенні покажчика на об'єкт, скористатися додатковими функціями, наприклад, відобразити повідомлення в рядку стану. За потреби скасувати проведені дії після переміщення покажчика із зазначеного об'єкта використовується обробник onMouseOut.

Оброблювач onClick викликається після клацання мишею на об'єкті.

Якщо обробник подій onClick повертає значення false, то сторінка, на яку вказує посилання, завантажуватися не буде. Приклад задання посилання, що дозволяє завантажувати сторінку тільки після клацання на кнопці ОК. Якщо клацнути на кнопці Cancel, сторінку завантажено не буде:

<A HREF = "file. Html" onClick = "return
(window.Confirm('Впевнені?'));"> Клацніть тут </ A>

Функція return визначає поведінку обробника подій. Вона повертає значення false після клацання на кнопці Cancel.

Оброблювач подій onDblClick виконується подібним чином. Подвійне клацання реєструється не тільки на посиланні, але і на рисунку та іншому об'єкті документа. Можна використовувати один об'єкт для завдання двох різних посилань. Одне посилання буде активізованим при подвійному натисканні на об'єкті, а друге − при одинарному.

Ще два обробники onMouseDown і onMouseUp використовуються для визначення натискання і відпускання кнопки миші.

У JavaScript кожна подія породжує асоційований з нею об'єкт event. Цей об'єкт містить всю інформацію про подію, і його можна передати процедурі обробки події.

Інформація залежить від типу події. Наприклад, об'єкт event події onMouseDown містить інформацію про використану кнопку миші: властивість which. Для лівої кнопки миші ця властивість дорівнює 1, а для правої − 3. Ця властивість визначається як для onClick, onDblClick, так і для onMouseDown і onMouseUp.

Властивість modifiers містить інформацію про те, яку клавішу − Alt, Shift або Ctrl, натиснено при виникненні події. Властивості screenX і screenY містять значення координат курсору миші в момент виникнення події.

У JavaScript можлива не тільки обробка події, згенерованої користувачем, наприклад, натискання кнопки. В JavaScript допустимою також є програмна генерація подій. Реалізовує це група методів − blur(), click(), focus(), select() та ін. Наприклад, подію onClick можна викликати методом click().

3.4. Динамічні рисунки

Зображення на Web-сторінках подано у вигляді масиву images. Змінюючи властивості елементів масиву, можна викликати заміну одного рисунка іншим. Це дає змогу створювати динамічні картинки, без оновлення вмісту Web-сторінки цілком. Цей метод не потребує використання шарів. Можна тільки змінювати рисунок на Web-сторінці.

Кожен об'єкт image має такі властивості:

– border. Відповідає атрибуту BORDER тега <IMG>, визначає межі рисунка;

– complete. Визначає ступінь завантаженості рисунка. Набуває булевих значень (true або false);

– height і width. Задають розміри рисунка;

– hspace і vspace. Визначають відступ і вигляд рисунка;

– name. Ім'я рисунка. Воно визначається атрибутом NAME при визначенні рисунка;

– src. Джерело рисунка, яке визначається адресою URL. При створенні динамічних рисунків це значення обов'язково змінюється.

Об'єкт image не володіє жодним методом, але дозволяє застосовувати обробники подій.

onLoad. Запускається після закінчення завантаження рисунка. Краще використовувати обробник onLoad для об'єкта image, а не для всього документа, оскільки останній запускається після завантаження всіх рисунків.

onAbort. Запускається при скасуванні користувачем завантаження сторінки, на якій ще не відображені рисунки.

onError. Запускається, якщо файл рисунка пошкоджено або не знайдено.

Зміна рисунків, як правило, використовується для виділення посилань на Web-сторінці. При наведенні покажчика миші змінюється весь рисунок або тільки певна його частина (наприклад, додається рамка).

Існує можливість динамічно керувати як розташуванням елемента на екрані, так і його розмірами. Це досягається за допомогою об'єкта style, що повертається властивістю style відповідного елемента документа. Цей об'єкт дозволяє контролювати всі параметри елемента, які задаються за допомогою таблиць стилів. Зокрема його властивості left, top, width, height встановлюють місце розташування елемента на екрані і його розміри. Повернені ними значення мають тип String. Те саме виконують властивості pixelLeft, pixelTop, pixelHeight, pixelWidth, але повернені ними значення мають тип int.

Завдання і контрольні запитання

Виконати за допомогою мови сценаріїв JavaScript такі дії:

1. Додати опис посилання в рядок стану браузера. Забезпечити очистку рядка стану при знятті миші з посилання.

2. Виконати перевірку правильності заповнення текстових полів форми, в які вводиться номер телефону і адреса E-mail. Повідомлення про помилки виводити на сторінку червоним кольором поруч з помилково заповненим полем. Встановити курсор у помилково заповнене поле.

3. Розробити спливаюче меню з трьох пунктів, відповідних зображенням на сторінці. При виборі деякого пункту меню забезпечити заміну відповідного зображення на інше.

4. Задати список з трьох пунктів, нумерований арабськими цифрами. Забезпечити заміну нумерації на римські цифри при наведенні миші на посилання.

5. Назвіть основні можливості мови сценаріїв JavaScript.

6. Які типи даних і вбудовані об'єкти існують у JavaScript?

7. Що таке обробник подій? Що відбувається при поверненні ним значення true і false?

8. Які об'єкти, властивості і методи дозволяють керувати елементами діалогових форм?

4. МОВА ПРОГРАМУВАННЯ PHP

4.1. Основні характеристики і історія PHP

Першу версію PHP було створено в 1994 р. програмістом Расмусом Лердорфом. Вона являла собою набір макросів для полегшення створення Web-сторінок. Цей набір називався Personal Home Page Tool.

У 1995 р. автор випустив другу версію мови, назва її − PHP/FI. При цьому автор перейшов на більш швидку компільовану мову C. Саме тоді в PHP було додано підтримку основних БД, що посилило її популярність.

Розширення є одним з основних способів збільшення функціональності PHP. Існують сотні розширень (для роботи з БД, графікою, PDF-файлами), будь-який розробник може самостійно створити розширення для своїх потреб. У стандартному оснащенні PHP містяться лише найбільш популярні і перевірені розширення.

Популярність PHP неухильно зростала. У 1997 р. над проектом вже працювала ціла команда програмістів. В результаті в 1998 р. виникла наступна версія − PHP 3. Вона містила новий лексичний аналізатор Zend. Було також внесено деякі зміни в синтаксис мови і додано нові функції. Нова версія виявилася найкращою на той час мовою програмування щодо сервера, і популярність продукту стала просто вражаючою.

PHP 4.0, заснована на Zend, вийшла у 2000 р. Покращилася продуктивність, з'явилися ще кілька нововведень - підтримка сесій, буферизація виведення. Виникла можливість створення об'єктно-орієнтованих програм.

П'яту версію PHP було випущено 2004 р. Зміни включають оновлення ядра Zend (Zend Engine 2), що збільшило ефективність інтерпретатора. Введено підтримку XML. Доопрацьовано функції ООП, які стали схожі на модель, використовувану в Java. Зокрема введено деструкцію, відкрито, закрито і захищено члени і методи, інтерфейси і клонування об'єктів.

План щодо створення 6-ї версії PHP було прийнято у 2006 р. Одним з основних нововведень мала стати підтримка Юнікоду. Зміни було вирішено вводити не революційно, а поступово, додаючи їх у нові випуски PHP 5.1, 5.2, 5.3. Одночасно зміни додаються у версію PHP 6.0, яка поки не стабільна і призначена тільки для тестування через складнощі з підтримкою Юнікоду.

Характеристики PHP:

- підтримка платформ Win 32, UNIX, OS/2, QNX, MacOS;
- сумісність із серверами: Apache (Win 32, UNIX), phttpd, fhttpd, ISAPI, NSAPI;
- переносимість. PHP може працювати на багатьох операційних системах і майже на всіх серверах;
- підтримка технологій COM, XML, Java, Flash;
- розвинено функціональність для роботи з мережевими з'єднаннями;
- підтримка понад 20 БД і розвинено функціональність для роботи з ними.

PHP працює на найрізноманітніших платформах. Це може бути Windows, багато версій UNIX, у тому числі Linux і Macintosh. PHP підтримує широкий набір серверів, серед яких − Apache (сам по собі є відкритим програмним продуктом), Microsoft Internet Information Server, WebSite Pro, iPlanet Web Server і Microsoft Personal Web Server. Останній сервер особливо корисний тоді, коли потрібно перевірити свої програми під Windows.

При створенні мови PHP враховувалася вимога високої інтеграції з базами даних. Це стало однією з причин того, що PHP є такою популярною при створенні Web-додатків. Багато баз даних безпосередньо підтримуються мовою PHP, і серед них такі, як Adabas D, InterBase, Solid, dBase, mSQL, Sybase, Empress, MySQL, Velocis, FilePro, Oracle, UNIX dbm, Informix та ін. Крім того, PHP підтримує стандарт ODBC.

4.2. Взаємодія HTML і PHP

Для написання PHP-програми потрібно відкрити текстовий редактор. Як і HTML-документи, PHP-програми складаються з простого тексту, тому писати їх можна за допомогою будь-якого текстового редактора.

Приклад першої PHP-програми:

```
<?  php
print "hello  Web! ";?>
```

У цього файлу має бути правильне розширення, тому що на підставі цього сервер розпізнає файл як PHP-програму і запустить інтерпретатор. За замовчуванням розширення файлів програм має бути php. Однак це може бути виправлено за допомогою файлу конфігурації.

Функцію print() призначено для виведення даних. У більшості випадків все, що виводиться за допомогою цієї функції, з'являється у вікні браузера.

Попередня програма складається тільки з команд PHP. Однак можна створити змішаний документ, просто додавши HTML-текст перед тегом, що відкриває, і після тега, що закриває.

Приклад:

```
 <Html> <head>
<Title> Документ, що містить PHP-команди і HTML-текст
</ title> </head> <body> <b> <?  php
print "Hello world!";
?>
</ b>
</ body>
</ Html>
```

Інтерпретатор ігнорує все, що міститься поза тегами PHP. Якщо відобразити цей документ на екрані браузера, то можна побачити слова hello world, виділені напівжирним шрифтом.

У документ можна включати стільки блоків PHP-команд, скільки буде потрібно для формування Web-сторінки. Кілька блоків команд в одному документі утворюють єдину програму. Це означає, що все, що ви визначите в першому блоці (змінні, функції або класи), буде доступно програмі в наступних блоках.

PHP автоматично створює набір глобальних змінних, що описують і сервер, і клієнта. Тому ці змінні називають змінними середовища, або змінними оточення. Залежно від того, з яким сервером ви працюєте, яка у вас операційна система і як вони сконфігуровані, набір цих змінних може змінюватися. У табл. 1 подано список деяких змінних, що часто трапляються, до яких можна звернутися як через масив $GLOBALS, так і безпосередньо.

Таблиця 1

Змінна	Опис	Приклад
$_HTTP_USER_AGE NT	Назва і версія клієнта	Mozilla / 4.6 (X11; I; Linux 2/2 / 6-15 apmac ppc)
$_REMOTE_ADDR	IP-адреса клієнта	158.152.55.35
$_REQUEST_METHOD	Метод запиту, GET або POST	POST
$_QUERY_STRING	При запиті GET	name-matt & address-unknown
$_REQOEST_URL	Повна адреса клієнта, включаючи рядок запиту	/tt/php/forms/
$_HTTP_REFERER	Адреса сторінки, з якої було зроблено запит	

4.3. Змінні і типи даних PHP

Імена змінних PHP починаються зі знака долара, за яким йдуть літерно-цифрові символи та символи підкреслення. Перший символ не може бути цифрою. Імена змінних чутливі до регістру.

Мова PHP є слаботипізованою, тобто змінні не вимагають суворого задання типу при їх оголошенні, а в ході виконання програми тип змінної може бути майже завжди змінено неявним чином, без спеціальних перетворень.

Наприклад, змінна, оголошена рядком, може використовуватися далі в арифметичних операціях, а потім їй можна привласнити об'єкт. Типи даних подано в табл. 2.

Таблиця 2

Тип даних	Опис
boolean	Логічний тип
integer	Ціле число, розрядність якого (32 або 64 біти) залежить від ОС
double, float	Дійсне число
string	Рядковий тип, може зберігати рядок довільного обсягу (але не більше 8 Мб)
array	Масив
object	Об'єкт
resource	Дескриптор, що дозволяє оперувати тим чи іншим ресурсом (файлом, БД або ін.)
NULL	Спеціальний тип, який сигналізує про те, що змінна не ініціалізована

Приклад оголошення змінних:

```
$Num = 25;
$Num = 025;  // вісімкове число
$Num = 0 x 25;  // шістнадцяткове число

$F = 0.012;
$F = 1.2 E -2;

$B = true;
```

Видалення змінної

Змінну може бути видалено за допомогою виклику функції функції unset().

```
$User = "Alex";
$Num = 25;
unset ($User, $Num);
```

Після виклику функції unset() пам'ять, виділена під значення змінної, повертається системі, а змінній присвоюється значення NULL (неініціалізована). Функція unset() може застосовуватися до всіх типів даних, включаючи масиви та об'єкти.

Для перевірки існування змінної використовується функція isset(), аргументом якої є одна або кілька змінних.

Для перевірки, чи є рядок порожній чи ні, використовується функція empty(). Аргумент функції − одна змінна. Функція повертає true, якщо змінна дорівнює порожньому рядку, нулю, NULL, false, пустому масиву, неініціалізованій змінній.

```
$str = '';
if (isset ($ str))  // true
echo ‘Змінна  str  існує ';
if (empty ($ str)) // true
echo Змінна  str порожня';
```

Наведемо деякі функції, що дозволяють визначати тип змінної (табл. 3).

Таблиця 3

Ім'я функції	Опис
gettype ($var)	Повертає тип змінної $ var у вигляді рядка "integer", "double" та ін.
is_array ($var)	Повертає true, якщо $ var є масивом, false − в іншому випадку
is_double ($var)	Повертає true, якщо $ var є змінною типу double, і false − у противному випадку
is_int ($var)	Повертає true, якщо $ var є змінною типу int, і false − у противному випадку
is_object ($var)	Повертає true, якщо $ var є об'єктом, і false −у противному випадку
is_string ($var)	Повертає true, якщо $ var є змінною типу string, і false − у противному випадку

Неявне приведення типів

У сильнотипізованих мовах програмування (C ++) при оголошенні змінної потрібно вказувати її тип, а використання змінної неправильного типу призводить до помилки.

У слаботипізованих мовах програмування (PHP) непотрібно явно вказувати тип змінної, а спроба використання змінної в контексті, де очікується змінна іншого типу, призведе до спроби автоматично (неявно) перетворити змінну до потрібного типу.

Наприклад, якщо рядок містить число і використовується в арифметичному виразі, то він автоматично буде приведений до числового типу.

Приклад:

```
$str = "5.5";
$n = $str + 2;
echo $n;  // 7.5
```

Рядок може містити, крім чисел, інші символи. Спершу інтерпретатор PHP матиме спробу витягти з початку рядка найбільш повне значення, відповідне числу. Якщо витягти число з рядка не вдається, його значення розглядається як нульове.

Якщо очікується логічний тип, то числа, що дорівнюють нулю, порожні рядки, рядок «0», порожні масиви і об'єкти приводяться до значення False, всі інші змінні розглядаються як true.

Явне приведення типів

Можна явно вимагати від інтерпретатора PHP перетворити змінну до деякого типу. Для цього існує кілька способів.

Перший спосіб полягає у використанні круглих дужок.

Приклад:

```
$ f = 5.75;
$ n = (int) f;
echo $ n;  // 5
```

4.4. Масиви в PHP

Масив можна створити за допомогою функції array ().

Приклад:

```
$ arr = array ( "Приходько", "Кирило", "Романович");
echo $ arr [0];
```

```
echo $ arr [1];
echo $ arr [2];
```

Для перегляду вмісту масиву передбачено функцію print_r().

```
echo "<pre>";
print _ r ($arr);
echo "</ pre>";
```

Результат:

```
Array
(
[0] => Приходько
[1] => Кирило
[2] => Романович
)
```

Елементам масиву автоматично призначаються індекси, починаючи з нульового, як це прийнято в С-подібних мовах. Індекс початкового елемента, а також порядок проходження елементів можна змінювати.

Змінимо попередній приклад.

```
$ Arr = array (10 => "Приходько", "Кирило", "Романович");
```

Другий і третій елементи масиву отримують номери 11 і 12.

```
$ Arr = array (10 => "Приходько", 9 =>  "Кирило", "Романович");
```

Всі непронумеровані елементи набувають значень, що дорівнювало б максимальному і збільшеному на 1.

```
[ 10] => Приходько
[9] => Кирило
[11] => Романович
```

Ще один спосіб створення масиву − привласнення елементам масиву значень:

```
$ Arr [0] = "Приходько";
```

$ Arr [1] = "Кирило";
$ Arr [2] = "Романович";

Можна не вказувати індекс у квадратних дужках. Тоді автоматично будуть присвоюватися індекси, починаючи з нульового.

Якщо елементи масиву містять однакові значення, то для його створення можна використовувати функцію array _ fill (). Синтаксис:

array _ fill ($ start _ index, $ num, $ value);

Функція повертає масив з $ num елементів, що мають значення $ value. Нумерація індексів починається зі значення $ start _ index.

$ Arr = array _ fill (3, 3, "Кирило");
echo "<pre>";
pr int _ r ($ arr);
echo "</ pre>";

Результат:
Array (
[3] => Кирило
[4] => Кирило
[5] => Кирило
)
Ще одна функція створення масиву - range ().
range ($_low, $_high [, $_step]);

Функція повертає масив зі значеннями з інтервалу від $_low до $_high з кроком $_step.

Приклад:
$ arr = range (3, 5);
echo "<pre>";
print _ r ($ arr);
echo "</ pre>";

Результат:
Array (

```
[0] => 3
[1] => 4
[2] => 5
)
```

Асоційовані масиви

Доступ до елементів масиву за номером зручний тоді, коли потрібно вибирати їх у тому порядку, в якому вони були створені, або при сортуванні масиву. Однак іноді буває необхідно звернутися до елемента масиву за його ім’ям. В асоціаційованих масивах індексами є не числа, а рядки.

Асоційований масив − це масив, проіндексований рядками. Індекси такого масиву називають ключами.

Асоційований масив можна створити безпосередньо або за допомогою функції array ().

Різниця між асоційованими масивами і звичайними у мові PHP не є принциповою. Ці масиви не є об'єктами різних типів, як у мові Perl. Однак слід оперувати ними по-різному, тому що вони потребують різного підходу і стратегії.

Створення асоційованого масиву за допомогою функції Array() .

Для того щоб створити асоційований масив за допомогою функції Array(), потрібно задати як ім'я, так і значення для кожного елемента. У наступному прикладі створюється асоційований масив $_arr з чотирьох елементів.

```
$_arr = array (
name => "Кирило",
surname => "Приходько",
age => 19,
address => "вул. Сумська, 30"
)
```

Тепер можна звернутися до будь-якого елемента масиву:

```
print $_arr [age];
```

Імена елементів масиву − це рядки, але їх не обов'язково брати в лапки; робити це необхідно лише в тому випадку, якщо ім'я складається з кількох слів.

Безпосереднє створення асоційованого масиву

Створити новий масив або додати до існуючого пару ім'я / значення можна просто присвоївши значення елементу масиву, вказавши цей елемент на ім'я. Наприклад, у нижчеподаному фрагменті ми знову створюємо масив $Arr .

```
$Arr  [Name] = "Кирило";
$Arr  [Surname] = "Приходько";
$Arr  [age] = 19;
$Arr  [Address] = "вул. Сумська, 30";
```

Якщо спробувати створити два елементи масиву з однаковими ключами, то створюється один елемент з останнім значенням.

Створення асоційованого масиву за допомогою функцій.

Функція array_combine() дозволяє створювати асоційований масив з двох інших. Синтаксис функції:

```
array_combine ($keys, $values);
```

Функція приймає як параметри масив $keys з ключами і масив $values зі значеннями. Повертає асоційований масив у разі успіху і false – в разі, якщо кількість елементів у цих масивах не збігається.

Приклад:

```
$ k = array ( 'name', 'surname', 'age');
$ v = array ( 'Кирило', 'Приходько', '19');
$ arr = array _ combine ($ k, $ v);
echo "<pre>";
print _ r ($ arr);
echo "</ pre>";
```

Результат:

```
Array (
[Name] => Кирило
[Surname] => Приходько
[Age] => 19
)
```

Функція compact () також дозволяє створювати асоційовані масиви. синтаксис:

compact ($ var [...]);

Як параметри функція приймає рядки з назвами змінних. Результуючий масив складається з елементів, ключі і значення яких збігаються з назвами і значенням змінних.

Приклад.

```
$name = "Кирило";
$surname = "Приходько";
$age = 19;
$arr = compact ( "name", "surname", "age");
```

Як параметри функції compact() можуть передаватися масиви, в такому випадку вона розбирає їх і обробляє кожен елемент масиву як окремий параметр.

Функція extract() виконує зворотну задачу: перетворює асоційований масив у змінні PHP.

PHP підтримує багатовимірні масиви.

Перегляд масиву за допомогою циклу.

Існує багато способів переглянути всі елементи масиву в циклі. Розглянемо найбільш поширену і потужну інструкцію foreach.

Якщо ми маємо простий, проіндексований числами масив, інструкція foreach використовується таким чином:

```
foreach ($array  as $tmp) {
}
```

У даному випадку $array − це ім'я масиву, який потрібно переглянути, а $ tmp − змінна, де буде тимчасово зберігатися значення кожного елемента. Цей спосіб продемонстровано у прикладі:

```
$users  = array  ( "Oleg", "Serg", "Sanya", "Igor");
foreach ($users  as $val)
{
print "$val <br>";  }
```

Значення кожного елемента масиву тимчасово розміщується у змінній $tmp, а потім виводиться на друк.

Для того щоб переглянути в циклі асоційований масив, потрібно написати інструкцію foreach по-іншому. У цьому випадку конструкція матиме такий вигляд:

```
foreach ($array  as $k ey => $value) {
}
```

Тут $array − це ім'я масиву, $key − змінна, в якій зберігається ім'я кожного елемента масиву, а $value − змінна, де тимчасово зберігається значення кожного елемента.

Приклад перегляду асоційованого масиву:

```
<?  php
$arr  = Array (
name => "Кирило",
surname => "Приходько",
age => 19,
address => "вул. Сумська, 30"
);
foreach ($arr as $key => $ val)
{
print "$key = $val <br>";
}
?>
```

4.5. Обробка HTML-форми

Приклад простої HTML-форми. Створимо файл form. html:

```
<Html>
<head>
<title> Проста  HTML - форма  </ title>
</ head>
<body>
<form action = "form .php" method = "GET">
<input type = "text" name = "user">
<br>
```

```
<textarea name = "address" rows = "5" cols = "40">
</ textarea>
<br>
<input type = "submit" value = "hit it!">
</ Form>
</ Body>
</ Html>
```

У цій формі, в якій є текстове поле з ім'ям "user", текстова область з ім'ям "address" і кнопка передачі даних "Submit". Елемент ACTION тега FORM вказує на файл form. php, таким чином даний файл буде обробляти дані форми. Той файл має бути в тому самому каталозі на сервері, що і HTML-документи.

Наведемо програму, яка обробляє дані форми з попереднього прикладу.

```
<Html>
<Head>
<Title> Читання даних форми </ title>
</ head>
<body>
<? php
print "Welcome <b> $ user </ b> <P> \ n \ n";
print "Your address is: <P> \ n, \ n <b> $ address </ b>";
?>
</ Body>
</ Html>
```

Програму слід зберегти під ім'ям form. php

Для перегляду результатів роботи слід вказати в рядку адреси браузера http: // localhost / form. html

Далі необхідно ввести дані у форму і натиснути на кнопку, що відправляє дані на сервер. Після цього в браузері відобразиться результат обробки даних програмою form. php.

Обробка елементів з багатозначним вибором.

Для того щоб програма могла мати доступ до всіх значень, обраних користувачем, потрібно до імені даного елемента

приписати пару порожніх квадратних дужок. Це і зроблено в прикладі.

```
<Html>
<Head>
<Title> HTML - форма з тегом select </ title> </ head>
<body>
<form action = "ff .php" method = "POST">
<input type = "text" name = "user">
<br>
<textarea na me = "address" rows = "5" cols = "40">
</ textarea>
<br>
<select name = "products []" multiple>
<option> Sony
<option> Panasonic
<option> Samsung
<option> LG
</ select>
<br>
<input type = "submit" value = "hit it!">
</ Form>
</ Body>
  </ Html>
```

Тепер у програмі, призначеній для обробки цієї форми, значення, вибрані користувачем в елементі "products []", будуть доступні як елементи масиву $ products.

```
< Html >
< Head >
< Title > Обробка даних форми </ title >
</ Head>
<Body>
<? Php
print "Welcome <b> $user </ b> <p> \ n \ n > ;
print "Your address is: <p> \ n \ n <b> $address </ b> <p> \ n \
n";
print "Your product choicee are: <p> \ n \ n;
```

```
print "<ul> \ n \ n ";
foreach {$products as $value)
{
print "<li> $value <br> \ n"
print "</ ul >"
?>
</ Body >
</ Html >
```

Тег SELECT − не єдиний елемент, який дозволяє ввести кілька значень. Можна створити кілька прапорців з одним ім'ям і ввести кілька значень під цим одним ім'ям. Якщо ім'я закінчується порожніми квадратними дужками, то всі значення, передані користувачем, стають доступними програмі у вигляді елементів масиву.

4.6. Форми і програми для передачі файлів на сервер

Розглянемо засоби PHP для передачі файлів на сервер.

У першу чергу нам потрібно створити відповідну форму. HTML-форма, що призначена для копіювання файлів і має відповідне поле, має містити аргумент ENCTYPE :

ENCTYPE = " multipart / form - data "

Крім того, PHP потребує, щоб у формі перед полем для копіювання файлів розташовувалося приховане поле. Таке приховане поле має називатися max _ file _ size і в ньому має бути записаний максимальний розмір файлу, який дозволяється передавати. Цей розмір не повинен перевищувати розмір, встановлений у полі upload_max_filesize у файлі php.ini, який зазвичай дорівнює 2 Мбайтам. Тепер можна підготувати саме поле для передачі файлу. Це звичайний елемент INPUT , у якого в аргументі TYPE записано "file".

Приклад:

```
< Html >
< Head >
< Title > Форма для передачі файлу </ title >
</ Head>
```

<Body>
<form enctype = "multipart / form-data" action = "<? print $PHP_SELF?>" method = "POST">
<input type = "hidden" name = "MAX_FILE_SIZE" value = "51200">
<Input type = "file" name = "fupload">
<input type = "submit" value = "upload ! ">
</ Form >
</ Body >
</ Html >

Зверніть увагу на те, що ця форма викликає сторінку, в якій вона записана.

Після того як файл переданий на сервер, він отримує унікальне ім'я і зберігається в каталозі для тимчасових файлів. Повний шлях до файлу записується в глобальну змінну з ім'ям, що збігається з ім'ям поля для передачі цього файлу (в даному випадку це буде $fupload).

PHP зберігає ще деяку додаткову інформацію про переданий файл у глобальних змінних. Їхні імена зроблені від імені поля для передачі файлу, за яким іде символ підкреслення, а потім рядки "name", "size" і "type". У табл. 4 наведено описи цих змінних.

Таблиця 4

Ім'я змінної	Опис	Приклад
$fupload	Шлях до тимчасового файлу	/tmp/serg
$fupload_name	Ім'я переданого файлу	serg.gif
$fupload_size	Розмір переданого файлу	2048
$fupload_type	Тип переданого файлу в системі MIME	image / gif

Крім того, у PHP є вбудована змінна, в якій у вигляді масиву записів є інформація про переданий файл. Якщо формою було передано кілька файлів, то у змінній $HTTP _ POST _ FILES міститиметься масив, впорядкований за іменами полів форми. Кожен з елементів цього масиву є асоційованим масивом. Елементи таких масивів описано у табл. 5.

Таблиця 5

Ім'я змінної	Опис	Приклад
$HTTP_POST_FlLES [fupload] [name]	Ім'я переданого файлу	serg.gif
$HTTP_POST_FILES [fupload] [size]	Розмір переданого файлу	2048
$HTTP_POST_FILES [fupload] [type]	Тип переданого файлу в системі MIME	image / gif

Приклад. Програма, яка виводить інформацію про переданий файл. Якщо переданий файл є файлом рисунка у форматі GIF, то програма виводить його на екран браузера.

```
< Html >
< Head >
< Title > Програма обробки переданого файлу </ title
>
</ Head>
<? php
$ file_dir = "/ home / htdocs / uploads";
$ file _ url = "http: // www.i.ua / uploads";
if ( isset ( $fupload))
{
print "path: $fupload <br> \ n";
 print "name: $fupload_name <br> \ n"; :

 print "size: $fupload_size bytes <br> \ n";

print "type: $fupload_type <p> \ n \ n";
{
copy ( $fupload, "$file dir / $fupload_name") or die ( "Could not
copy");
print "<img src = \ " $file url / $fupload name \ "> <p> \ n \ n"; '
 }
}
?>
<Body>
```

<Form enctype = "multipart/form-data" action = "<? Php print $PHP_SELF?>"
method = "POST">
<Input type = "hidden" name = "MAX_FlLE_SIZE" value = "2400">
<Input type = "file" name = "fupload">
<Input type = "submit" value = "Send file!">
</ Form >
</ Body >
</ Html >

Спочатку ми перевіряємо, чи визначено змінну $fupload . Якщо її визначено, то можемо вважати, що файл переданий.

Якщо файл переданий, то шлях до цього файлу на сервері записаний у змінній $fupload і ми виводимо цей шлях на екран браузера. Крім того, ми виводимо ім'я файлу, записане у змінній $fupload_name, розмір файлу, який записаний у змінній $fupload_size, і тип файлу, записаний у змінній $fupload_type .

Після цього перевіряється значення змінної $fupload_type . Якщо воно збігається з "image/gif", ми вважаємо, що це GIF-файл.

Визначивши , що передано рисунок типу GIF, ми копіюємо його з тимчасового каталогу в наш каталог. Функція copy() приймає два аргументи − шлях до вихідного файлу і нове його положення. Ця функція повертає true, якщо файл скопійований успішно. Початкове розміщення і ім'я файлу записано у змінній $_fupload, а нове положення − у змінній $file_ ir. Доводиться об'єднувати нове положення файлу і його початкову назву для того, щоб сформувати другий аргумент функції copy(). У результаті файл у новому місці отримує своє первісне ім'я.

4.7. Дослідження файлів і каталогів

У мові PHP є багато функцій для отримання різноманітної інформації про файлову систему. Розглянемо деякі з них.

Для того щоб перевірити, чи існує файл, застосовується функція file_exists(). Ця функція приймає рядок, що містить повний або відносний шлях до файлу. Якщо файл знайдено, то функція повертає true , в іншому випадку − false .

if (file_exists ("serg.txt")) ;
print "serg.txt знайдений";

Функція filesize() визначає розмір файлу в байтах. Вона повертає значення false , якщо визначити розмір файлу не вдається.

print " Розмір файлу serg.txt -";
print filesize ("serg.txt ");

У мові PHP є кілька функцій для роботи з каталогами, за допомогою яких можна створювати каталоги, видаляти їх або читати.

Перед тим як читати вміст каталогу, його потрібно відкрити і отримати покажчик на нього за допомогою функції opendir(). Ця функція має тільки один аргумент − ім'я та шлях до каталогу. Вона повертає true , якщо каталог був успішно відкритий, і false − у противному випадку. Помилка при відкритті може бути викликана тим, що каталог не існує або програма не має права його читати.

$fp = opendir (" testdir ");

Так само, як читались рядки з файлу за допомогою функції fgets(), є можливість читати в каталозі імена файлів і підкаталогів за допомогою функції readdir(). Ця функція має один аргумент − ім'я каталогу − і повертає рядок, що містить ім'я знайденого об'єкта, тобто файлу або підкаталогу. Після досягнення кінця файлу вона повертає false. Функція readdir() повертає імена об'єктів, але не шляхи до них.

Приклад програми, яка читає вміст каталогу:

< Html >
< Head >
< Title > Читання каталогу за допомогою функції readdir()
</ title >
</ Head>
<Body>
< ? php
$dirname = "testdir";

```
$dh = opendir ( $dirname);
while (gettype ($file = readdir ($dh)) ! = boolean)
        {
if (is_dir ( "$dirname / $ file"))
        print "(D)";
print "$file <br>";
 }
        closedir ($dh);
        ?>
 </ Body>
</ Html >
```

Відкривається каталог, і в циклі читається його зміст. Викликається функція readdir() в перевірному циклі, привласнюючи повернене значення змінній $file. У тілі циклу ми об'єднуємо змінні $dirname і $file, отримуючи повний шлях, який потім перевіряємо. Якщо цей шлях веде до каталогу, то виводиться буква D перед його ім'ям. Потім друкується змінна $file.

4.8. Робота з базами даних

Перед тим як починати працювати з базою даних, потрібно підключитися до сервера. Для цього в мові PHP є функція mysql_connect(). Функція mysql_connect() повертає ідентифікатор підключення, якщо все пройшло успішно. Цей ідентифікатор можна зберегти у змінній і надалі користуватися ним для роботи із сервером бази даних.

Приклад. Підключення до сервера бази даних:

```
$link = mysql_connect ( "localhost", "root", "serg");
if (! $ link )
  die ( "Couldn ' t connect to MySQL");
```

Після того як з'єднання із сервером MySQL встановлено, потрібно вибрати базу даних, з якої збираєтеся працювати. Для цього існує функція mysql_select_db().

Приклад:

Вибрати базу даних з ім'ям sample.

```
$Database = " sample ";
```

mysql _ select _ db ($ sample) or die (" Couldn ' t open ' '$ sample);

Ми створюємо в базі даних sample таблицю з ім'ям domains . Таблиця містить чотири поля: поле первинного ключа з ім'ям id, значення якого буде автоматично збільшуватися при додаванні нових записів, поле domain, куди записаний рядок змінної довжини типу VARCHAR, поле sex , що складається з одного символа, і поле mail, що містить адресу користувача. Для створення цієї таблиці було використано таку інструкцію SQL :

create table domains (id INT NOT NULL AUTO INCREMENT, PRIMARY KEY (id),
domain VARCHAR (20),
sex CHAR (l),
mail VARCHAR (20));

Для додавання даних у цю таблицю нам потрібно сконструювати і виконати запит SQL . Для цього у мові PHP є функція mysql_query(). Цій функції потрібно передати рядок із запитом і необов'язковий ідентифікатор підключення. Якщо цей ідентифікатор опущений, то буде використовуватися останнє отримане підключення. Функція повертає додатне число, в разі успішного виконання запиту, і false, якщо в запиті міститься помилка або якщо немає прав на виконання такого запиту.

Приклад:

```
< Html >
< Head >
< Title > Додавання запису в таблицю </ title >
</ Head>
<Body>
<? Php
$user = " serg ";
$pass = " rgrgrrg ";
$db = "sample";
$link = mysql_connect ( "localhost", $user, $pass);
if (! $link)
die ( "Could not connect to MySQL");
```

```
mysql_select_db ($db, $link)
or die ( "Could not open $ db:" mysqlerror());
$query = "INSERT INTO domains (domain, sex, mail)
values ( 'xyz.com', 'F', 'serg@ i.ua') ";
mysql _ query ($query, $link)
or die ( "Could not add data to \" domains \ "table:"
mysql_error ());
mysql_close ($link);
?>
</ Body >
</ html >
```

4.9. Результат запиту до БД

Після виконання запиту SELECT і отримання його ідентифікатора можна в циклі переглянути всі записи, знайдені в результаті запиту. PHP створює внутрішній покажчик, в якому записано позицію в наборі записів результату. Цей покажчик автоматично переміщається на наступну позицію після звернення до поточного запису.

За допомогою функції mysql_fetch() можна для кожного запису отримати масив, що складається з її полів. Цій функції потрібно передати ідентифікатор запиту. Після досягнення кінця запиту функція mysql _ fetch () поверне значення false .

Програма виводить на екран всю таблицю domains .

```
< Html >
< Head >
< Title > Виведення усіх записів таблиці </ title >
</ Head>
<Body>
<? Php
$user = " serg ";
$pass = " rgrgrgr ";
$db = "sample";
$link = mysql_connect ( "localhost", $user, $pass);
if (! $link)
die ( "Could not connect to MySQL");
mysql_select_db ($db, $link )
```

```
or die ( "Could not open $ db:" mysql _ error ());
$ result = mysql _ query ( "SELECT * FROM domains" );
$ num_rows = mysql_num _rows ($result);
print "There are currently $num_rows rows in the table <P>";
print "<table border = l> \ n";
while ($a_row = mysql_fetch _ row ($result _y)
{
print "<tr> \ n";
foreach ($a_row as $field)
print "\ t <td> $f ield </ td> \ n";
print "</ tr> \ n";
 }
print "</ table> \ n";
mysql _ close ($link);
?>
</ Body >
</ Html >
```

Завдання і контрольні запитання

1. Надіслати дані форми (4-5 полів різного типу) на сервер і обробити їх.
2. Створити файл, каталог, відкрити файл для запису, записати у файл, прочитати з нього.
3. Створити таблицю (3-4 поля) відповідно до своєї предметної галузі. Додати записи.
4. Виконати запит до таблиці і проаналізувати результати.
5. Змінити дані в таблиці.

Бібліографічний список

1. Котеров, Д. PHP 5 в подлиннике. [Текст] / Д. Котеров, А. Костарев. – 2-е изд. – СПб. : «BHV-СПб», 2015. – 1104 с.

2. Гарнаев, С. Ю. Web-программирование на Java и JavaScript. [Текст] / С. Ю. Гарнаев, А. Ю. Гарнаев. – СПб. : БХВ, 2014. – 1072 с.

3. Development of method of definition maximum clique in a non-oriented graph [Text] / S. V. Listrovoy, V. M. Butenko, V. O. Bryksin, O. V. Golovko // EasternEuropean Journal of Enterprise Technologies. – 2017. – Vol. 5, № 4 (89). – P. 12–17.

4. Formulation of the Problem of Maximum Clique Determination in Non-Oriented Graphs / S. V. Listrovoy, O. V. Golovko, V. M. Butenko, M. V. Ushakov // International Journal of Engineering & Technology Vol 7 No 4.3 (2018): Special Issue 3. – P. 293–297. DOI: 10.14419/ijet.v7i4.3.19807.

5. Signal flow graph models and alternative gain formula for multiprobe microwave multimeter [Text] / O. B. Zaichenko, V. M. Butenko, M. A. Miroshnyk // Інформаційно-керуючі системи на залізничному транспорті. – 2016. – № 12. – С. 12–17.

6. Listrovoy, S. V. Algoritm of Sub Exponential Complexity for the SAT [Text] / S.V. Listrovoy, V.M.Butenko // International Jornal of Computer and Information Technology (ISSN: 2279-0764) Volume 02. – Issue 05, September 2013. – P. 837–842.

7. Математичне моделювання в розподілених інформаційно-керуючих системах залізничного транспорту [Текст] : монографія / С. В. Лістровий, С. В. Панченко, В. І. Мойсеєнко, В. М. Бутенко. – Харків : ФОП Бровін О. В., 2017. – 220 с.

8. Завдання і методичні вказівки до розрахунково-графічної та контрольної робіт з дисциплін «Програмування» та «Інформатика» для студентів факультету АТЗ [Текст] / В. М. Бутенко, О. В. Головко, М. О. Колісник, С. О.Бантюкова. – Харків : УкрДАЗТ, 2016. – 74 с.

9. Павленко, Е. П. Исследование методов разработки программного обеспечения компьютерной инженерии на основе типовых программных элементов [Текст] / Е. П. Павленко, В. М. Бутенко, В. А. Губин // Вісник Національного техн. ун-ту

"ХПІ". Сер. Системний аналіз, управління та інформаційні технології = Bulletin of the National Technical University "KhPI". Ser. System analysis, control and information technology : зб. наук. праць. – Харків : НТУ "ХПІ", 2019. – № 44 (1320). – С. 11-15.

10. Мойсеєнко, В. І. Нові процедури обслуговування інформаційно-керуючих систем та контроль фактичного виконання роботи [Текст] / В. І. Мойсеєнко, В. М. Бутенко, В. В. Гаєвський // 31-ша міжнародна наук.-практ. конф. "Інформаційно-керуючі системи на залізничному транспорті". – 2018. – № 4. – С. 59–60.

11. Основи алгоритмізації базових обчислювальних процесів [Текст] : навч. посібник / В. С. Меркулов, В. М. Бутенко та ін. — Харків : УкрДАЗТ, 2008. – 163 с.

12. Бутенко, В. М. Інтернет на залізницях України [Текст] / В. М. Бутенко // Информационно-управляющие системы на железнодорожном транспорте. – 1997. – №1. – С.47–49.

13. Бутенко, В. М. Адресація та захист інформації в мережі RailWayNet [Текст] / В. М. Бутенко // Информационно-управляющие системы на железнодорожном транспорте. – 1997. – №3. – С. 24–26

14. Кількісний аналіз показників надійності систем автоматики з використанням моделювання дерев небезпечних відмов [Текст] / В. М. Бутенко, Д. О. Зубрицький, С. В. Сіроштан, Є. С. Строєв // Зб. наук. праць. – Харків: УкрДАЗТ, 2008. – Вип. 92. – С. 133–138.

15. Бутенко, В. М. Методичний посібник до лабораторних робіт з дисципліни “Математичні методи та моделі розрахунку на ЕОМ” [Текст] / В. М. Бутенко, О. Б. Болотов, В. В. Шумеєв. – Харків : УкрДАЗТ, 2006. – Ч. 1. – 28 с.

16. Бутенко, В. М. Інформаційні системи на залізничному транспорті [Текст] : конспект лекцій / В. М. Бутенко, С. Є. Бантюков, В. Г. Пчолін. – Харків : УкрДАЗТ, 2008. – Ч. 1. – 28 с.

17. Основи програмування мовами високого рівня [Текст] : навч. посібник / В. М. Бутенко, В. С. Меркулов, О. В. Чаленко, О. В. Казанко. – Харків : УкрДАЗТ, 2009. – 206 с.

18. Бутенко, В. М. Компьютерная система управления движением поездов [Текст] / В. М. Бутенко, В. И. Мойсеенко, Д. М. Кузьменко // Залізнич. трансп. України. – 2000.– № 5–6. – С. 80–82.

Додаток

Завдання до практичних робіт

Завдання до розділу 1. Мова розмітки гіпертексту HTML

Завдання 1

Створити HTML-документ, що містить вікно заголовка з написом «Математична задача варіанта № __» відповідно до варіанта та головною частиною з текстом опису розв'язання. Головна частина має містити: назву задачі шрифтом 1, словесне формулювання завдання шрифтом 2, список вхідних даних шрифтом 3, математичні формули для обчислення шуканої величини шрифтом 4, розміщені в рамці з ліній кольору 4, стиль рамки 5 .

Варіант 1
Обчислення площі круга.
(Шрифт 1 – Times New Roman, 18 п, жирний, вирівнювання по центру.
Шрифт 2 – Times New Roman, 16 п, жирний, вирівнювання по ширині.
Шрифт 3 – Arial, 13 п; курсив, вирівнювання по лівому краю.
Шрифт 4 – Times New Roman, 16 п; вирівнювання по центру.
Колір 4 – зелений.
Стиль рамки 5 – подвійна 2 п)

Варіант 2
Обчислення площі прямокутника.
(Шрифт 1 – Tahoma, 18 п, жирний, вирівнювання по центру.
Шрифт 2 – Tahoma, 16 п, жирний, вирівнювання по ширині.
Шрифт 3 – Arial, 13 п; курсив, вирівнювання по правому краю.
Шрифт 4 – Tahoma, 16 п; вирівнювання по центру.
Колір 4 – синій.
Стиль рамки 5 – одинарна 2 п)

Варіант 3
Обчислення площі прямокутного трикутника.

(Шрифт 1 – Times New Roman, 20 п, курсив, вирівнювання по центру.
Шрифт 2 – Times New Roman, 16 п, курсив, вирівнювання по ширині.
Шрифт 3 – Arial, 14 п; курсив, вирівнювання по центру
Шрифт 4 – Times New Roman, 16 п; вирівнювання по правому краю.
Колір 4 – світло-оранжевий.
Стиль рамки 5 – подвійна 2 п)

Варіант 4
Обчислення площі рівнобедреної трапеції.
(Шрифт 1 – Tahoma, 17 п, жирний, вирівнювання по центру.
Шрифт 2 – Tahoma, 16 п, жирний, вирівнювання по ширині.
Шрифт 3 – Arial, 13 п; курсив, вирівнювання по правому краю.
Шрифт 4 – Tahoma, 20 п; вирівнювання по центру.
Колір 4 – синій.
Стиль рамки 5 – одинарна 2 п)

Варіант 5
Обчислення площі ромба.
(Шрифт 1 – Times New Roman, 18 п, жирний, вирівнювання по центру.
Шрифт 2 – Times New Roman, 16 п, жирний, вирівнювання по ширині.
Шрифт 3 – Arial, 15 п; курсив, вирівнювання по лівому краю.
Шрифт 4 – Times New Roman, 16 п; вирівнювання по центру.
Колір 4 – чорний.
Стиль рамки 5 – подвійна 2 п)

Варіант 6
Обчислення площі рівнобедреного трикутника.
(Шрифт 1 – Tahoma, 22 п, жирний, вирівнювання по центру.
Шрифт 2 – Tahoma, 16 п, жирний, вирівнювання по ширині.
Шрифт 3 – Arial, 15 п; курсив, вирівнювання по правому краю.
Шрифт 4 – Tahoma, 16 п; вирівнювання по центру.
Колір 4 – блакитний.
Стиль рамки 5 – одинарна 5 п)

Варіант 7
Обчислення периметра кола.
(Шрифт 1 – Times New Roman, 18 п, жирний, вирівнювання по центру.
Шрифт 2 – Times New Roman, 16 п, вирівнювання по ширині.
Шрифт 3 – Arial, 13 п; курсив, вирівнювання по лівому краю.
Шрифт 4 – Times New Roman, 16 п; жирний, вирівнювання по центру.
Колір 4 – коричневий.
Стиль рамки 5 – подвійна 4 п)

Варіант 8
Обчислення периметра прямокутника.
(Шрифт 1 – Tahoma, 16 п, жирний, вирівнювання по центру.
Шрифт 2 – Tahoma, 16 п, жирний, вирівнювання по ширині.
Шрифт 3 – Arial, 15 п; курсив, вирівнювання по правому краю.
Шрифт 4 – Tahoma, 20 п; вирівнювання по центру.
Колір 4 – червоний.
Стиль рамки 5 – одинарна 5 п)

Варіант 9
Обчислення периметра прямокутного трикутника.
(Шрифт 1 – Times New Roman, 18 п, жирний, вирівнювання по центру.
Шрифт 2 – Times New Roman, 16 п, вирівнювання по ширині.
Шрифт 3 – Arial, 13 п; курсив, вирівнювання по центру.
Шрифт 4 – Times New Roman, 16 п; жирний, вирівнювання по центру.
Колір 4 – рожевий.
Стиль рамки 5 – подвійна 4 п)

Варіант 10
Обчислення периметра рівнобедреної трапеції.
(Шрифт 1 – Tahoma, 22 п, жирний, вирівнювання по центру.
Шрифт 2 – Tahoma, 16 п, жирний, вирівнювання по ширині.
Шрифт 3 – Arial, 15 п; курсив, вирівнювання по правому краю.
Шрифт 4 – Tahoma, 16 п; вирівнювання по центру.
Колір 4 – темно-сірий.
Стиль рамки 5 – одинарна 4 п)

Варіант 11
Обчислення периметра ромба.
(Шрифт 1 – Times New Roman, 18 п, жирний, вирівнювання по центру.
Шрифт 2 – Times New Roman, 16 п, вирівнювання по ширині.
Шрифт 3 – Arial, 13 п; жирний, вирівнювання по центру.
Шрифт 4 – Times New Roman, 16 п; курсив, вирівнювання по центру.
Колір 4 – оливковий.
Стиль рамки 5 – подвійна 4 п)

Варіант 12
Обчислення периметра рівнобедреного трикутника.
(Шрифт 1 – Tahoma, 22 п, жирний, курсив, вирівнювання по центру.
Шрифт 2 – Tahoma, 14 п, вирівнювання по ширині.
Шрифт 3 – Arial, 15 п; вирівнювання по правому краю.
Шрифт 4 – Tahoma, 14 п; жирний, вирівнювання по центру.
Колір 4 – блакитний.
Стиль рамки 5 – одинарна 5 п)

Варіант 13
Обчислення об’єму кулі.
(Шрифт 1 – Times New Roman, 18 п, жирний, вирівнювання по центру.
Шрифт 2 – Times New Roman, 16 п, вирівнювання по ширині.
Шрифт 3 – Arial, 13 п; жирний, вирівнювання по центру.
Шрифт 4 – Times New Roman, 16 п; курсив,вирівнювання по центру.
Колір 4 – жовтий.
Стиль рамки 5 – подвійна 4 п)

Варіант 14
Обчислення об’єму паралелепіпеда.
(Шрифт 1 – Tahoma, 22 п, жирний, курсив, вирівнювання по центру.
Шрифт 2 – Tahoma, 14 п, вирівнювання по ширині.
Шрифт 3 – Arial, 15 п; вирівнювання по правому краю.

Шрифт 4 – Tahoma, 14 п; жирний, вирівнювання по центру.
Колір 4 – зелений
Стиль рамки 5 – подвійна 2 п)

Варіант 15
Обчислення об’єму циліндра.
(Шрифт 1 – Times New Roman, 20 п, жирний, вирівнювання по центру.
Шрифт 2 – Times New Roman, 15 п, вирівнювання по ширині.
Шрифт 3 – Arial, 15 п; жирний, вирівнювання по центру.
Шрифт 4 – Times New Roman, 16 п; курсив, вирівнювання по центру.
Колір 4 – оливковий.
Стиль рамки 5 – подвійна 4 п)

Завдання 2

Створити сайт за певною тематикою, що складається з кількох HTML-документів, деякі з яких створені згідно з варіантом першого завдання.

Варіант 1
Створити сайт-довідку про формули обчислення характеристик кола (периметр і площа). Сайт складається з HTML-документів, створених згідно з варіантам 1 і 7 першого завдання, та містить головну сторінку із заголовком «Характеристики кола», підзаголовками, визначенням кола як геометричної фігури та посиланням на вищевказані сторінки. Також на цій сторінці має бути розташований рисунок заданої фігури з параметрами , що її характеризують. (Коло – радіус). Усім сторінкам встановити один фон – блідо-рожевий.

Варіант 2
Створити сайт-довідку про формули обчислення характеристик прямокутника (периметр і площа). Сайт складається з HTML-документів, створених згідно з варіантами 2 і 8 першого завдання, та містить головну сторінку із заголовком «Характеристики прямокутника», підзаголовками, визначенням

прямокутника як геометричної фігури та посиланням на вищевказані сторінки. Також на цій сторінці має бути розташований рисунок заданої фігури з параметрами , що її характеризують. (Прямокутника – висота і ширина). Усім сторінкам встановити один фон – блідо-синій.

Варіант 3
Створити сайт-довідку про формули обчислення характеристик прямокутного трикутника (периметр і площа). Сайт складається з HTML-документів, створених згідно з варіантами 3 і 9 першого завдання, та містить головну сторінку із заголовком «Характеристики прямокутного трикутника», підзаголовками, визначенням прямокутного трикутника як геометричної фігури та посиланням на вищевказані сторінки. Також на цій сторінці має бути розташований рисунок заданої фігури з параметрами, що її характеризують. (Прямокутного трикутника – довжини катетів). Усім сторінкам встановити один фон – блідо-зелений.

Варіант 4
Створити сайт-довідку про формули обчислення характеристик рівнобедреної трапеції (периметр і площа). Сайт складається з HTML-документів, створених згідно з варіантами 4 і 10 першого завдання, та містить головну сторінку із заголовком «Характеристики рівнобедреної трапеції», підзаголовками, визначенням рівнобедреної трапеції як геометричної фігури та посиланням на вищевказані сторінки. Також на цій сторінці має бути розташований рисунок заданої фігури з параметрами , що її характеризують. (Трапеції – довжини основ і висота). Усім сторінкам встановити один фон – блідо-оливковий.

Варіант 5
Створити сайт-довідку про формули обчислення характеристик ромба (периметр і площа). Сайт складається з HTML-документів, створених згідно з варіантами 5 і 11 першого завдання, та містить головну сторінку із заголовком «Характеристики ромба», підзаголовками, визначенням ромба як геометричної фігури та посиланням на вищевказані сторінки. Також на цій сторінці має бути розташований рисунок заданої фігури з параметрами , що її

характеризують. (Ромб – довжини діагоналей). Усім сторінкам встановити один фон – блідо-лимонний.

Варіант 6
Створити сайт-довідку про формули обчислення характеристик рівнобедреного трикутника (периметр і площа). Сайт складається з HTML-документів, створених згідно з варіантами 6 і 12 першого завдання, та містить головну сторінку із заголовком «Характеристики рівнобедреного трикутника», підзаголовками, визначенням рівнобедреного трикутника як геометричної фігури та посиланням на вищевказані сторінки. Також на цій сторінці має бути розташований рисунок заданої фігури з параметрами , що її характеризують. (Рівнобедреного трикутника – довжина основи і висота). Усім сторінкам встановити один фон – блідо-фіолетовий.

Варіант 7
Створити сайт-довідку про формули обчислення площі різних фігур. Сайт складається з HTML-документів, створених згідно з варіантами 1, 2, 3 (можна використати не всі, але не менше ніж дві на вибір) з першого завдання, та містить головну сторінку із заголовком «Площа», підзаголовками, визначенням площі фігур з вибраних сторінок та посиланням на вищевказані сторінки. Також на цій сторінці мають бути розташовані рисунки фігур з параметрами , що дають можливість обчислити площу. Сторінкам встановити один фон – блідо-лимонний, а головній – на 10 пунктів яскравіший.

Варіант 8
Створити сайт-довідку про формули обчислення об’єму різних фігур. Сайт складається з HTML-документів, створених згідно з варіантами 13, 14, 15 (можна використати не всі, але не менше ніж дві на вибір) з першого завдання, та містить головну сторінку із заголовком «Об’єм», підзаголовками, визначенням площі фігур з вибраних сторінок та посиланням на вищевказані сторінки. Також на цій сторінці мають бути розташовані рисунки фігур з параметрами, що дають можливість обчислити об’єм. Сторінкам встановити один фон – блідо-салатовий, а головній – на 8 пунктів яскравіший.

Варіант 9

Створити сайт-довідку про формули обчислення периметра різних фігур. Сайт складається з HTML-документів, створених згідно з варіантами 1, 2,3 (можна використати не всі, але не менше ніж дві на вибір) з першого завдання, та містить головну сторінку із заголовком «Периметр», підзаголовками, визначенням периметра фігур з вибраних сторінок та посиланням на вищевказані сторінки. Також на цій сторінці мають бути розташовані рисунки фігур з параметрами , що дають можливість обчислити периметр. Сторінкам встановити один фон – блідо-лимонний, а головній – на 10 пунктів яскравіший.

Варіант 10

Створити сайт-довідку про формули обчислення площі різних фігур. Сайт складається з HTML-документів, створених згідно з варіантами 4, 5, 6 (можна використати не всі, але не менше ніж дві на вибір) з першого завдання, та містить головну сторінку із заголовком «Площа», підзаголовками, визначенням площі фігур з вибраних сторінок та посиланням на вищевказані сторінки. Також на цій сторінці мають бути розташовані рисунки фігур з параметрами, що дають можливість обчислити площу. Сторінкам встановити один фон – блідо-синій, а головній – на 5 пунктів яскравіший.

Варіант 11

Створити сайт-довідку про формули обчислення периметра різних фігур. Сайт складається з HTML-документів, створених згідно з варіантами 4, 5, 6 (можна використати не всі, але не менше ніж дві на вибір) з першого завдання, та містить головну сторінку із заголовком «Периметр», підзаголовками, визначенням периметра фігур з вибраних сторінок та посиланням на вищевказані сторінки. Також на цій сторінці мають бути розташовані рисунки фігур з параметрами , що дають можливість обчислити периметр. Сторінкам встановити один фон – блідо-червоний, а головній – на 15 пунктів яскравіший.

Варіант 12
Створити сайт-довідку про фігури, повязані з колом. Сайт складається з HTML-документів, створених згідно з варіантами 1, 13, 15 з першого завдання, та містить головну сторінку із заголовком «Коло та пов'язані з ним фігури», підзаголовками, визначенням кола та поясненням як фігури з вибраних сторінок, з ним пов'язаних, та посиланням на вищевказані сторінки. Також на цій сторінці мають бути розташовані рисунки фігур. Сторінкам встановити один фон – блідо-бузковий, а головній – на 15 пунктів яскравіший.

Варіант 13
Створити сайт-довідку про фігури – чотирикутники та їх площі. Сайт складається з HTML-документів, створених згідно з варіантами 2, 4, 5 з першого завдання, та містить головну сторінку із заголовком «Чотирикутники», підзаголовками, визначенням чотирикутника взагалі та його видів з вибраних сторінок та посиланням на вищевказані сторінки. Також на цій сторінці мають бути розташовані рисунки фігур. Сторінкам встановити один фон – блідо-зелений, а головній – на 15 пунктів яскравіший.

Варіант 14
Створити сайт-довідку про фігури – чотирикутники та їх периметри. Сайт складається з HTML-документів, створених згідно з варіантами 8, 10, 11 з першого завдання, та містить головну сторінку із заголовком «Чотирикутники», підзаголовками, визначенням чотирикутника взагалі та його видів з вибраних сторінок та посиланням на вищевказані сторінки. Також на цій сторінці мають бути розташовані рисунки фігур. Сторінкам встановити один фон – блідо-сірий, а головній – на 5 пунктів яскравіший.

Варіант 15
Створити сайт-довідку про фігури – трикутники та їх площі. Сайт складається з HTML-документів, створених згідно з варіантами 3, 6 з першого завдання, та містить головну сторінку із заголовком «Трикутники», підзаголовками, визначенням трикутника взагалі та його видів з вибраних сторінок та посиланням на вищевказані сторінки. Також на цій сторінці мають бути розташовані рисунки фігур. Сторінкам встановити один фон – блідо-зелений, а головній – на 15 пунктів яскравіший.

Завдання до розділу 2. Динамічний HTML

Завдання 1

У завданні 2 до розділу 1 було створено сайти за певною тематикою, що складались з кількох HTML. Створити для них сторінку змісту сайта на основі фреймів з використанням каскадних стилів.

Варіант 1

Створено сайт-довідку про формули обчислення характеристик кола (периметр і площа). Сайт складається з HTML-документів, створених згідно з варіантами 1 і 7 першого завдання, та містить головну сторінку із заголовком «Характеристики кола», підзаголовками, визначенням кола як геометричної фігури та посиланням на вищевказані сторінки. Сторінка-зміст має відповідати такій схемі:

Головна сторінка	
Сторінка варіанта 1	Сторінка варіанта 7

Варіант 2

Створено сайт-довідку про формули обчислення характеристик прямокутника (периметр і площа). Сайт складається з HTML-документів, створених згідно з варіантами 2 і 8 першого завдання, та містить головну сторінку із заголовком «Характеристики прямокутника», підзаголовками, визначенням прямокутника як геометричної фігури та посиланням на вищевказані сторінки. Сторінка -зміст має відповідати такій схемі:

Головна сторінка	
Сторінка варіанта 2	Сторінка варіанта 8

Варіант 3

Створено сайт-довідку про формули обчислення характеристик прямокутного трикутника (периметр і площа). Сайт складається з HTML-документів, створених згідно з варіантами 3 і 9 першого завдання, та містить головну сторінку із заголовком «Характеристики прямокутного трикутника», підзаголовками, визначенням прямокутного трикутника як геометричної фігури та посиланням на вищевказані сторінки. Сторінка-зміст має відповідати такій схемі:

<table>
<tr><td rowspan="2">Головна сторінка</td><td>Сторінка варіанта 3</td></tr>
<tr><td>Сторінка варіанта 9</td></tr>
</table>

Варіант 4

Створено сайт-довідку про формули обчислення характеристик рівнобедреної трапеції (периметр і площа). Сайт складається з HTML-документів, створених згідно з варіантами 4 і 10 першого завдання, та містить головну сторінку із заголовком «Характеристики рівнобедреної трапеції», підзаголовками, визначенням рівнобедреної трапеції як геометричної фігури та посиланням на вищевказані сторінки. Сторінка -зміст має відповідати такій схемі:

<table>
<tr><td rowspan="2">Головна сторінка</td><td>Сторінка варіанта 4</td></tr>
<tr><td>Сторінка варіанта 10</td></tr>
</table>

Варіант 5

Створено сайт-довідку про формули обчислення характеристик ромба (периметр і площа). Сайт складається з HTML-документів,

створених згідно з варіантами 5 і 11 першого завдання, та містить головну сторінку із заголовком «Характеристики ромба», підзаголовками, визначенням ромба як геометричної фігури та посиланням на вищевказані сторінки. Сторінка-зміст має відповідати такій схемі:

Головна сторінка	Сторінка варіанта 5	Сторінка варіанта 11

Варіант 6

Створити сайт-довідку про формули обчислення характеристик рівнобедреного трикутника (периметр і площа). Сайт складається з HTML-документів, створених згідно з варіантами 6 і 12 першого завдання, та містить головну сторінку із заголовком «Характеристики рівнобедреного трикутника», підзаголовками, визначенням рівнобедреного трикутника як геометричної фігури та посиланням на вищевказані сторінки. Сторінка -зміст має відповідати такій схемі:

Головна сторінка	Сторінка варіанта 6	Сторінка варіанта 12

Варіант 7

Створено сайт-довідку про формули обчислення площі різних фігур. Сайт складається з HTML-документів, створених згідно з варіантами 1, 2, 3 з першого завдання, та містить головну сторінку із заголовком «Площа», підзаголовками, визначенням площі фігур з вибраних сторінок та посиланням на вищевказані сторінки. Сторінка -зміст має відповідати такій схемі:

Головна сторінка	Сторінка варіанта 1
	Сторінка варіанта 2
	Сторінка варіанта 3

Варіант 8

Створено сайт-довідку про формули обчислення об'єму різних фігур. Сайт складається з HTML-документів, створених згідно з варіантами 13, 14, 15 з першого завдання, та містить головну сторінку із заголовком «Об'єм», підзаголовками, визначенням площі фігур з вибраних сторінок та посиланням на вищевказані сторінки. Сторінка-зміст має відповідати такій схемі:

<table>
<tr><td rowspan="3">Головна сторінка</td><td>Сторінка варіанта 13</td></tr>
<tr><td>Сторінка варіанта 14</td></tr>
<tr><td>Сторінка варіанта 15</td></tr>
</table>

Варіант 9

Створено сайт-довідку про формули обчислення периметра різних фігур. Сайт складається з HTML-документів, створених згідно з варіантами 1, 2, 3 (можна використати не всі, але не менше ніж дві на вибір) з першого завдання, та містить головну сторінку із заголовком «Периметр», підзаголовками, визначенням периметра фігур з вибраних сторінок та посиланням на вищевказані сторінки. Сторінка-зміст має відповідати такій схемі:

<table>
<tr><td colspan="3">Головна сторінка</td></tr>
<tr><td>Сторінка варіанта 1</td><td>Сторінка варіанта 2</td><td>Сторінка варіанта 3</td></tr>
</table>

Варіант 10

Створено сайт-довідку про формули обчислення площі різних фігур. Сайт складається з HTML-документів, створених згідно з варіантами 4, 5, 6 (можна використати не всі, але не менше ніж дві на вибір) з першого завдання, та містить головну сторінку із заголовком «Площа», підзаголовками, визначенням площі фігур з вибраних сторінок та посиланням на вищевказані сторінки. Сторінка-зміст має відповідати такій схемі:

<table>
<tr><td colspan="3">Головна сторінка</td></tr>
<tr><td>Сторінка варіанта 4</td><td>Сторінка варіанта 5</td><td>Сторінка варіанта 6</td></tr>
</table>

Варіант 11

Створено сайт-довідку про формули обчислення периметра різних фігур. Сайт складається з HTML-документів, створених згідно з варіантами 4, 5, 6 (можна використати не всі, але не менше ніж дві на вибір) з першого завдання, та містить головну сторінку із заголовком «Периметр», підзаголовками, визначенням периметра фігур з вибраних сторінок та посиланням на вищевказані сторінки. Сторінка-зміст має відповідати такій схемі:

<table>
<tr><td rowspan="2">Сторінка варіанта 4</td><td>Головна сторінка</td><td rowspan="2">Сторінка варіанта 6</td></tr>
<tr><td>Сторінка варіанта 5</td></tr>
</table>

Варіант 12

Створено сайт-довідку про фігури, пов’язані з колом. Сайт складається з HTML-документів, створених згідно з варіантами 1, 13, 15 з першого завдання, та містить головну сторінку із заголовком «Коло та пов’язані з ним фігури», підзаголовками, визначенням кола та поясненням як фігури з вибраних сторінок, з ним пов’язаних, та посиланням на вищевказані сторінки. Сторінка-зміст має відповідати такій схемі:

<table>
<tr><td rowspan="2">Головна сторінка</td><td rowspan="2">Сторінка варіанта 1</td><td>Сторінка варіанта 13</td></tr>
<tr><td>Сторінка варіанта 15</td></tr>
</table>

Варіант 13

Створено сайт-довідку про фігури – чотирикутники та їх площі. Сайт складається з HTML-документів, створених згідно з варіантами 2, 4, 5 з першого завдання, та містить головну сторінку із заголовком «Чотирикутники», підзаголовками, визначенням чотирикутника взагалі та його видів з вибраних сторінок та посиланням на вищевказані сторінки. Сторінка-зміст має відповідати такій схемі:

<table>
<tr><td colspan="2">Головна сторінка</td></tr>
<tr><td colspan="2">Сторінка варіанта 2</td></tr>
<tr><td>Сторінка варіанта 4</td><td>Сторінка варіанта 5</td></tr>
</table>

Варіант 14
Створено сайт-довідку про фігури – чотирикутники та їх периметри. Сайт складається з HTML-документів, створених згідно з варіантами 8, 10, 11 з першого завдання, та містить головну сторінку із заголовком «Чотирикутники», підзаголовками, визначенням чотирикутника взагалі та його видів з вибраних сторінок та посиланням на вищевказані сторінки. Сторінка-зміст має відповідати такій схемі:

Головна сторінка	
Сторінка варіанта 8	
Сторінка варіанта 10	Сторінка варіанта 11

Варіант 15
Створено сайт-довідку про фігури – трикутники та їх площі. Сайт складається з HTML-документів, створених згідно з варіантами 3, 6 з першого завдання, та містить головну сторінку із заголовком «Трикутники», підзаголовками, визначенням трикутника взагалі та його видів з вибраних сторінок та посиланням на вищевказані сторінки. Сторінка-зміст має відповідати такій схемі:

Головна сторінка	
Сторінка варіанта 3	Сторінка варіанта 6

Завдання до розділу 3. Програмування мовою JAVASCRIPT

Завдання 1

Створити сторінку, що здійснює розрахунок заданої характеристики геометричної фігури за формулою, використовуючи скрипти та об’єкт <INPUT type=text>.

Варіант 1
Обчислення площі круга.
За допомогою об’єкта <INPUT type=text> вводимо значення радіуса.

Варіант 2
Обчислення площі прямокутника.
За допомогою об’єктів <INPUT type=text> вводимо значення висоти і ширини.

Варіант 3
Обчислення площі прямокутного трикутника.
За допомогою об’єктів <INPUT type=text> вводимо значення катетів .

Варіант 4
Обчислення площі рівнобедреної трапеції.
За допомогою об’єктів <INPUT type=text> вводимо значення висоти і довжини основ.

Варіант 5
Обчислення площі ромба.
За допомогою об’єктів <INPUT type=text> вводимо значення діагоналей.

Варіант 6
Обчислення площі рівнобедреного трикутника.
За допомогою об’єктів <INPUT type=text> вводимо значення висоти і довжини основи.

Варіант 7
Обчислення периметра кола.
За допомогою об’єкта <INPUT type=text> вводимо значення радіуса.

Варіант 8
Обчислення периметра прямокутника.
За допомогою об’єктів <INPUT type=text> вводимо значення висоти і ширини.

Варіант 9
Обчислення периметра прямокутного трикутника.
За допомогою об’єктів <INPUT type=text> вводимо значення катетів.

Варіант 10
Обчислення периметра рівнобедреної трапеції.
За допомогою об’єктів <INPUT type=text> вводимо значення висоти і довжини основ .

Варіант 11
Обчислення периметра ромба.
За допомогою об’єктів <INPUT type=text> вводимо значення діагоналей.

Варіант 12
Обчислення периметра рівнобедреного трикутника.
За допомогою об’єктів <INPUT type=text> вводимо значення висоти і довжини основи .

Варіант 13
Обчислення об’єму кулі.
За допомогою об’єкта <INPUT type=text> вводимо значення радіуса.

Варіант 14
Обчислення об’єму паралелепіпеда.
За допомогою об’єктів <INPUT type=text> вводимо значення висоти, довжини і ширини.

Варіант 15
Обчислення об’єму циліндра.
За допомогою об’єктів <INPUT type=text> вводимо значення висоти і радіуса основи.

Завдання 2

Створити сторінку, де виконуються обчислення згідно з варіантом завдання (табл. Д.1), та вивести результат на Web-сторінку. Значення незалежних змінних, що не вказані в умовах задач, вважати довільними.

Таблиця Д.1

Номер варіанта	Функції для розрахунків	Вивести на екран
1	2	3
1	$q = \frac{2x + \ln^2 x + \sqrt{x}}{a + mc + t};\ t = \frac{mb}{a + x};\quad m = a + 2b^2$, *якщо* $b = 4.3$	*q, t, m*
2	$t = \frac{2s + m^2}{a - x} + \sqrt{s + c^3} - x^2;\ s = \frac{a + 2c}{m} + q;\quad m = 2\sqrt{a^2 + c}$, *якщо* $a = 5{,}6$	*t, s, m*
3	$y = \frac{cx^2 + mb^3 + \ln x + 7}{b + m + \sqrt{c}};\ m = 2x + \sqrt{x};\ b = ac^2 - m$	*y,m,b*
4	$r = \frac{mc + \sqrt{2x + f^2}}{\sin^2 x + 3c};\ f = \frac{a}{x^2 + b} + \frac{m}{2c} - x;\ m = b + \cos(a - x)$, *якщо* $b = 0{,}27$	*r, f, m*
5	$z = \frac{a + bx}{m} + \frac{c}{a + x} - q;\ q = \frac{b + cx}{2} + \frac{1}{a - cx};\ a = bx - mt^2$, *якщо* $t = 0{,}5$	*z, q, a*
6	$c = \frac{a + \sin 2x}{m - r} + \frac{\cos^2 x}{ax};\ r = \frac{2m}{x} + arctg\,2x;\ m = \frac{b + ax}{2} + \frac{3f}{b - x}$	*c, r, m*
7	$f = \frac{3y}{m + x} + \sqrt{\frac{b}{m + 2x}} + \sin(m^2 + x);\ m = 2a - cd;\ c = a + x^2$	*f, m, c*
8	$r = \sqrt{t + 2q};\quad q = \frac{m + x}{\ln x};\quad m = \frac{c + 2{,}5d}{a + x};\quad c = \sqrt{x^2 + a}$	*r, m, q*
9	$s = \frac{7}{m + 2t} + \frac{a}{b + cx} + m^2 - 4z;\quad z = 3m + at^2;\ m = \sqrt{a + 2cx^2}$, *якщо* $b = 3{,}25$	*s, z, m*
10	$z = \frac{2x}{b + y} + \frac{c}{b - y} + \frac{2bx}{b - xy};\quad y = 2\ln(\lvert x - a \rvert);\quad b = c - 2x,\quad$ *якщо* $\quad c = 12$	*z, y, b*
11	$q = mc^2 + 2f^5 + \cos^2 x;\quad m = \frac{a}{c} + \frac{b}{x} + \sqrt{f};\quad f = \ln(a + 2{,}5x),\quad$ *якщо* $\quad a = 3{,}7;$	*q, m, f*
12	$z = \frac{m^2 cx}{\sqrt{a}} + bc^3 - \ln a;\quad b = a + \sin 2x^2;\ m = a + b^2 - \cos x,$ *якщо* $a = 2$	*z, b, m*

Продовження табл. Д.1

1	2	3
13	$r=\frac{c+2x^2}{m}+at^2+\sqrt{x};\quad c=b+2x-\sin x;\quad m=\frac{2+x}{c}+arctgx-a$	*r, c, m*
14	$q=\frac{3m}{ax}+\sqrt{b+cx}+\lg\left(\lvert b-mx\rvert\right);\quad m=a+r^3+\cos x;\ r=c+2x;c=2a-x$	*q, m, r*
15	$q=\frac{at^2+rc-ax}{b+\sqrt{c+x}-5};r=\left(mx+2t+c\right)^3;m=2cx;\quad t=ax^2+bx-c,якщо\quad c=7{,}5$	*q, r, m, t*

Завдання 3

Створити сторінку, де виконуються обчислення згідно з варіантом завдання (табл. Д.2), та вивести результат на Web-сторінку. Значення незалежних змінних, що не вказані в умовах задач, вважати довільними.

Таблиця Д.2

Номер варіанта	Значення для розрахунку	Функції для розрахунків	Прим.
1	2	3	4
1	При $A=130, B=2, D=4$ та С =1; С =2; С =3 обчислити ***y*** за формулою	$y=\begin{cases}(A-B)+C*2; якщо\ C<2\\ D*B;\ якщо\ C>2\\ D-B;\ якщо\ C=2\end{cases}$,	
2	При $A=32, B=18, D=2$ та С =2; С =3; С =4 обчислити ***y*** за формулою	$y=\begin{cases}(A-B)+C*2; якщо\ C<3\\ D*2;\ якщо\ C>3\\ B+D-9;\ якщо\ C=3\end{cases}$,	
3	При $A=14, B=3, D=4$ та С =1; С =2; С =3 обчислити ***y*** за формулою	$y=\begin{cases}(A-6)+C*2; якщо\ C<2\\ D*2;\ якщо\ C>2\\ B+D-9;\ якщо\ C=2\end{cases}$,	
4	При $A=34, B=8, D=6$ та С =3; С =4; С =5 обчислити ***y*** за формулою	$y=\begin{cases}(A-C)+C*2; якщо\ C<4\\ 2+D*2;\ якщо\ C>4\\ B*4-9;\ якщо\ C=4\end{cases}$,	
5	При $A=30, B=8, D=1$ та С =4; С =5; С =6 обчислити ***y*** за формулою	$y=\begin{cases}(A-6)+C*2; якщо\ C<5\\ D*2;\ якщо\ C>5\\ B+D+9;\ якщо\ C=5\end{cases}$,	

Продовження табл. Д.2

1	2	3	4
6	При $A=34, B=8, D=6$ та С =5; С =6; С =7 обчислити ***y*** за формулою	$y=\begin{cases}(A-9)+C*2; якщо\ C<6\\ D*3; якщо\ C>6\\ B+D+6; якщо\ C=6\end{cases},$	
7	При $A=31, B=11, D=6$ та С =6; С =7; С =8 обчислити ***y*** за формулою	$y=\begin{cases}(A+9)-C*2; якщо\ C<7\\ D/2-A; якщо\ C>7\\ B+D-7; якщо\ C=7\end{cases},$	
8	При $A=34, B=8, D=6$ та С =7; С =8; С =9 обчислити ***y*** за формулою	$y=\begin{cases}(A-9)+C*2; якщо\ C<8\\ D/4; якщо\ C>8\\ B-D+2; якщо\ C=8\end{cases},$	
9	При $A=34, B=8, D=6$ та С =8; С =9; С =10 обчислити ***y*** за формулою	$y=\begin{cases}(A+9)-C*4; якщо\ C<9\\ D-3; якщо\ C>9\\ B-D+5; якщо\ C=9\end{cases},$	
10	При $A=34, B=8, D=6$ та С =9; С =10; С =11 обчислити ***y*** за формулою	$y=\begin{cases}(A+2)-C/2; якщо\ C>10\\ D-7; якщо\ C<10\\ B-D+8; якщо\ C=10\end{cases},$	
11	При $A=84, B=9, D=8$ та С =8; С =9; С =10 обчислити ***y*** за формулою	$y=\begin{cases}(A+9)-C*4; якщо\ C<9\\ D-3; якщо\ C>9\\ B-D+5; якщо\ C=9\end{cases},$	
12	При $A=140, B=18, D=26$ та С =11; С =12; С =13 обчислити ***y*** за формулою	$y=\begin{cases}(A+3)-C*2; якщо\ C>12\\ D-9; якщо\ C<12\\ B-D+8; якщо\ C=12\end{cases},$	
13	При $A=180, B=3, D=60$ та С =2; С =3; С =4 обчислити ***y*** за формулою	$y=\begin{cases}(A-B)+C*2; якщо\ C<3\\ D*2; якщо\ C>3\\ B+D-9; якщо\ C=3\end{cases},$	
14	При $A=204, B=2, D=9$ та С =3; С =4; С =5 обчислити ***y*** за формулою	$y=\begin{cases}(A+C)-C*4; якщо\ C>4\\ 5+D*2; якщо\ C<4\\ B*4-9; якщо\ C=4\end{cases},$	
15	При $A=134, B=2, D=6$ та С =4; С =5; С =6 обчислити ***y*** за формулою	$y=\begin{cases}(A-4)+C*2; якщо\ C<5\\ D-12; якщо\ C>5\\ B+D*2; якщо\ C=5\end{cases},$	

Завдання 4

Створити сторінку, де виконуються обчислення згідно з варіантом завдання (табл. Д.3), та вивести результат на Web-сторінку. Значення незалежних змінних, що не вказані в умовах задач, вважати довільними.

Таблиця Д.3

Номер варіанта	Значення для розрахунку	Функції для розрахунків	Прим.
1	2	3	4
1	При $A=13, B=2, D=4$ обчислити *y* за формулою	$y = A*2 + \sum_{i=0}^{D}(i+2)$	
2	При $A=2, B=18, D=2$ обчислити *y* за формулою	$y = D + \sum_{i=0}^{B}(i+A)$	
3	При $A=4, B=30, D=4$ обчислити *y* за формулою	$y = B + \sum_{i=0}^{D}(i-A)$	
4	При $A=34, B=8, D=6$ обчислити *y* за формулою	$y = D + \sum_{i=0}^{B}(i+A)$	
5	При $A=30, B=8, D=3$ обчислити *y* за формулою	$y = A + \sum_{i=0}^{D}(i+B)$	
6	При $A=3, B=8, D=26$ обчислити *y* за формулою	$y = D + \sum_{i=0}^{B}(i+A)$	
7	При $A=31, B=11, D=6$ обчислити *y* за формулою	$y = A + \sum_{i=0}^{D}(i+2)$	
8	При $A=3, B=8, D=6$ обчислити *y* за формулою	$y = D + \sum_{i=0}^{B}(i+A)$	
9	При $A=34, B=8, D=6$ обчислити *y* за формулою	$y = A + \sum_{i=0}^{D}(i+B)$	
10	При $A=9, B=8, D=6$ обчислити *y* за формулою	$y = D + \sum_{i=0}^{B}(i+A)$	
11	При $A=84, B=9, D=8$ обчислити *y* за формулою	$y = 4*\sum_{i=0}^{D}(i+2)$	
12	При $A=14, B=3, D=6$ обчислити *y* за формулою	$y = D + \sum_{i=0}^{B}(i+A)$	

Продовження табл. Д.3

1	2	3	4
13	При $A=18, B=3, D=6$ обчислити *y* за формулою	$y = 2 * \sum_{i=0}^{D} (i+2)$	
14	При $A=20, B=2, D=9$ обчислити *y* за формулою	$y = D + \sum_{i=0}^{B} (i+A)$	
15	При $A=34, B=2, D=6$ обчислити *y* за формулою	$y = A + \sum_{i=0}^{D} (i+B)$	

Завдання 5

Створити сторінку, де виконуються обчислення згідно з варіантом завдання (табл. Д.4–Д.6), та вивести результат на Web-сторінку. Значення незалежних змінних, що не вказані в умовах задач, вважати довільними.

Таблиця Д.4

Номер варіанта	Значення для розрахунку	Функції для розрахунків	Прим.
1	2	3	4
1	При $A=13, B=2, D=4$ обчислити ***y*** за формулою	$y = A + \sum_{i=0}^{D} \sum_{j=0}^{B} (i+j),$	
2	При $A=32, B=18, D=2$ обчислити ***y*** за формулою	$y = (-A) + \sum_{i=0}^{B} \sum_{j=0}^{D} (i+j),$	
3	При $A=14, B=9, D=4$ обчислити ***y*** за формулою	$y = -(2 * A) + \sum_{i=0}^{D} \sum_{j=D}^{B} (i+j),$	
4	При $A=34, B=8, D=6$ обчислити ***y*** за формулою	$y = \left\lfloor \sum_{i=0}^{B} \sum_{j=0}^{D} (i+j) \right\rfloor - A/2,$	
5	При $A=30, B=8, D=13$ обчислити ***y*** за формулою	$y = \left\lfloor \sum_{i=0}^{B} \sum_{j=0}^{D} (i+j) \right\rfloor - A*2),$	
6	При $A=4, B=8, D=6$ обчислити ***y*** за формулою	$y = 4 * A + \sum_{i=0}^{D} \sum_{j=0}^{B} (i+j),$	
7	При $A=31, B=11, D=6$ обчислити ***y*** за формулою	$y = (-A) + \sum_{i=0}^{B} \sum_{j=0}^{D} (i+j),$	

Продовження табл. Д.4

1	2	3	4
8	При $A=34, B=8, D=6$ обчислити ***y*** за формулою	$y=-(2*A)+\sum_{i=0}^{D}\sum_{j=D}^{B}(i+j),$	
9	При $A=28, B=8, D=6$ обчислити ***y*** за формулою	$y=\left\lfloor\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j)\right\rfloor-A/4,$	
10	При $A=14, B=8, D=6$ обчислити ***y*** за формулою	$y=\left\lfloor\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j)\right\rfloor-A*4),$	
11	При $A=84, B=9, D=8$ обчислити ***y*** за формулою	$y=4*A+\sum_{i=0}^{D}\sum_{j=0}^{B}(i+j),$	
12	При $A=14, B=8, D=6$ обчислити ***y*** за формулою	$y=-1*A+\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j),$	
13	При $A=18, B=3, D=6$ обчислити ***y*** за формулою	$y=-1*2*A+\sum_{i=0}^{D}\sum_{j=D}^{B}(i+j),$	
14	При $A=20, B=2, D=9$ обчислити ***y*** за формулою	$y=\left\lfloor\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j)\right\rfloor-A/4,$	
15	При $A=8, B=2, D=6$ обчислити ***y*** за формулою	$y=\left\lfloor\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j)\right\rfloor-A*4),$	

Таблиця Д.5

Номер варіанта	Значення для розрахунку	Функції для розрахунків	Прим.
1	2	3	4
1	При $A=32, B=18, D=2$ обчислити ***y*** за формулою	$y=(-A)+\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j),$	
2	При $A=14, B=9, D=4$ обчислити ***y*** за формулою	$y=-(2*A)+\sum_{i=0}^{D}\sum_{j=D}^{B}(i+j),$	
3	При $A=34, B=8, D=6$ обчислити ***y*** за формулою	$y=\left\lfloor\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j)\right\rfloor-A/2,$	
4	При $A=30, B=8, D=13$ обчислити ***y*** за формулою	$y=\left\lfloor\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j)\right\rfloor-A*2),$	
5	При $A=4, B=8, D=6$ обчислити ***y*** за формулою	$y=2*A+\sum_{i=0}^{D}\sum_{j=0}^{B}(i+j),$	
6	При $A=31, B=11, D=6$ обчислити ***y*** за формулою	$y=(-A)+\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j),$	

Продовження табл. Д.5

1	2	3	4
7	При $A=34, B=8, D=6$ обчислити ***y*** за формулою	$y=-(2*A)+\sum_{i=0}^{D}\sum_{j=D}^{B}(i+j),$	
8	При $A=28, B=8, D=6$ обчислити ***y*** за формулою	$y=\left\lfloor\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j)\right\rfloor-A/4,$	
9	При $A=14, B=8, D=6$ обчислити ***y*** за формулою	$y=\left\lfloor\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j)\right\rfloor-A*4),$	
10	При $A=84, B=9, D=8$ обчислити ***y*** за формулою	$y=4*A+\sum_{i=0}^{D}\sum_{j=0}^{B}(i+j),$	
11	При $A=14, B=8, D=6$ обчислити ***y*** за формулою	$y=-1*A+\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j),$	
12	При $A=18, B=3, D=6$ обчислити ***y*** за формулою	$y=-1*2*A+\sum_{i=0}^{D}\sum_{j=D}^{B}(i+j),$	
13	При $A=20, B=2, D=9$ обчислити ***y*** за формулою	$y=\left\lfloor\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j)\right\rfloor-A/4,$	
14	При $A=8, B=2, D=6$ обчислити ***y*** за формулою	$y=\left\lfloor\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j)\right\rfloor-A*4),$	
15	При $A=8, B=2, D=6$ обчислити ***y*** за формулою	$y=\left\lfloor\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j)\right\rfloor-A*2),$	

Таблиця Д.6

Номер варіанта	Значення для розрахунку	Функції для розрахунків	Прим.
1	2	3	4
1	При $A=13, B=2, D=4$ обчислити ***C*** за формулою	$C=A+\sum_{i=0}^{D}\sum_{j=0}^{B}(i+j),$	
2	При $A=32, B=18, D=2$ обчислити ***C*** за формулою	$C=(-A)+\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j),$	
3	При $A=14, B=9, D=4$ обчислити ***C*** за формулою	$C=-(2*A)+\sum_{i=0}^{D}\sum_{j=D}^{B}(i+j),$	
4	При $A=34, B=8, D=6$ обчислити ***C*** за формулою	$C=\left\lfloor\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j)\right\rfloor-A/2,$	
5	При $A=30, B=8, D=13$ обчислити ***C*** за формулою	$C=\left\lfloor\sum_{i=0}^{B}\sum_{j=0}^{D}(i+j)\right\rfloor-A*2),$	

Продовження табл. Д.6

1	2	3	4
6	При $A=4, B=8, D=6$ обчислити C за формулою	$C = 4*A + \sum_{i=0}^{D}\sum_{j=0}^{B}(i+j),$	
7	При $A=31, B=11, D=6$ обчислити C за формулою	$C = (-A) + \sum_{i=0}^{B}\sum_{j=0}^{D}(i+j),$	
8	При $A=34, B=8, D=6$ обчислити C за формулою	$C = -(2*A) + \sum_{i=0}^{D}\sum_{j=D}^{B}(i+j),$	
9	При $A=28, B=8, D=6$ обчислити C за формулою	$C = \left\lfloor \sum_{i=0}^{B}\sum_{j=0}^{D}(i+j) \right\rfloor - A/4,$	
10	При $A=14, B=8, D=6$ обчислити C за формулою	$C = \left\lfloor \sum_{i=0}^{B}\sum_{j=0}^{D}(i+j) \right\rfloor - A*4),$	
11	При $A=84, B=9, D=8$ обчислити C за формулою	$C = 4*A + \sum_{i=0}^{D}\sum_{j=0}^{B}(i+j),$	
12	При $A=14, B=8, D=6$ обчислити C за формулою	$C = -1*A + \sum_{i=0}^{B}\sum_{j=0}^{D}(i+j),$	
13	При $A=18, B=3, D=6$ обчислити C за формулою	$C = -1*2 + \sum_{i=0}^{D}\sum_{j=D}^{B}(i+j),$	
14	При $A=20, B=2, D=9$ обчислити C за формулою	$C = \left\lfloor \sum_{i=0}^{B}\sum_{j=0}^{D}(i+j) \right\rfloor - A/4,$	
15	При $A=8, B=2, D=6$ обчислити C за формулою	$C = \left\lfloor \sum_{i=0}^{B}\sum_{j=0}^{D}(i+j) \right\rfloor - A*4),$	

Завдання 6

Створити сторінку, де виконуються обчислення згідно з варіантом завдання (табл. Д.7), та вивести результат на Web-сторінку. Значення незалежних змінних, що не вказані в умовах задач, вважати довільним.

Таблиця Д.7

Номер варіанта	Підзавдання 6.1	Підзавдання 6.2
1	2	3
1	$R = \dfrac{2{,}5m! + a\sum_{k=1}^{5} k^3}{m\prod_{i=1}^{m}(i+2) + a^{m!-1}}$	$Q = \dfrac{mb^2 + 2x}{\prod_{i=1}^{7}(i+2)} - \dfrac{\sqrt{k!+m!}}{m+5x};$ $Z = \dfrac{k+m+2Q}{m+c+k!} + 2^x;$ *якщо* $x \in [a;c]$; $h_x = 0{,}15a$

1	2	3
2	$Q=\dfrac{2(m+k)!+a\sum_{i=1}^{12}(i-a)^3}{\sqrt{k!+7a}+3{,}2k!}$	$F=\dfrac{3\sqrt{(m+a)!}+2{,}2x^2+\sin x}{\lg^2 x+\ln^2 x};$ $Z=\sqrt{F}+\sum_{i=3}^{9}\left(i+a^2\right)+\sqrt{x};$ $R=a+Z-13{,}5x,$ $якщо\ x\in[2;9;];\ hx=0{,}1a$
3	$S=\dfrac{2{,}3\lg\left(\prod_{i=1}^{11}(i+b)^2+3x\right)}{\sum_{j=1}^{3}(j+mc)^3+2{,}5x^2};$ $якщо\ x\in[3;12];\ hx=0{,}5$	$R=\sqrt{\prod_{i=2}^{9}\dfrac{i+a}{2}+3f^3};$ $Z=\ln(a+R)+\lg\left(\sum_{i=3}^{5}i^3\right);$ $Q=R+fZ-5a;$ $якщо\ f\in[2;10];\ h_f=0{,}2a$
4	$Q=\dfrac{x\sqrt{m+k!}+a\ln(m!+k)}{2x+(m+k)!};$ $якщо\ x\in[3;6{,}5];$ $hx=0{,}1$	$L=\dfrac{1}{2x}+\dfrac{2}{m!}+\dfrac{(m+2)!}{a+x};$ $S=x\prod_{l=10}^{20}(i-m)^2+3{,}45c;$ $якщо\ x\in[a;2c];\ hx=0{,}25$
5	$Z=\dfrac{\sqrt{m!+k}+2k!}{\prod_{i=1}^{7}(i-a)^2+\ln\sum_{j=2}^{10}(b-j)^3};$	$Z=\prod_{i=1}^{7}\left(i+2a^2\right)+3x\sum_{k=2}^{12}k^5;$ $R=3\sqrt{Z}+m!x-17{,}2;$ $якщо\ x\in[a;m];\ hx=0{,}125$
6	$Q=\dfrac{\ln(k!+4{,}2)-\lg(m^3-3)}{m!+\prod_{j=1}^{5}(j+k)^2+3{,}7\sum_{i=2}^{9}(i-a)^2}$	$Z=\dfrac{2m-1}{k!+x}+\dfrac{\prod_{i=1}^{7}(i+2a)^2}{(m+k)!}+2^{x+m};$ $Q=aZ+2mx;$ $якщо\ x\in[m;k];\ hx=1$
7	$Z=\dfrac{2m!+3k!+\sqrt{m!-k!}}{\prod_{i=1}^{m}(i+m)^2+5\sum_{i=3}^{7}(m+k+i)^3}$	$Z=\dfrac{a^x}{m!}+\dfrac{R-3}{(k+2)!}+\dfrac{(b-a)^3}{3S};$ $R=\prod_{i=3}^{5}(i+k);$ $S=\sum_{i=1}^{7}(i+a)^3;$ $якщо\ x\in[a;5];\ hx=0{,}15$

1	2	3
8	$R = \frac{3m!+k+x^2+b^x}{\lg(a+2k+x)};$ якщо $x \in [0,2;3,5]$ $hx = 0,05$	$Q = \frac{b^2-a}{m!+x} + \sqrt{\frac{c+2m!}{\sum_{k=3}^{7}(a+k)^2}};$ $F = 2\sqrt{Q-c};$ $R = mx + 2F^3;$ якщо $x \in [1;2m]$; $hx = 0,1$
9	$Z = \frac{e^x}{n!} + \frac{(m+n)!}{x} + \frac{a^x+b}{(n+3)!};$ якщо $x \in [2;5]$; $hx = 0,25$	$Z = \frac{a\sum_{i=1}^{7} ik^2 + (k+a)^x}{\sqrt{k!+x} + (k+a)!};$ $R = 3Z + ax^5;$ якщо $x \in [2;5]$; $hx = 0,25$
10	$Z = \frac{(m!+2k)!-x\sqrt{2k+1}}{\prod_{i=1}^{3}(i+2)^2 + \ln(m!+4) + e^x};$ якщо $x \in [m;k]$; $hx = 0,2m$	$Q = \frac{a+mt}{\prod_{i=1}^{7}(i+2m)} + 2\sum_{j=4}^{7}\frac{m!}{j+2};$ $Z = \frac{Q}{a+t} + \sqrt[3]{a+mt};$ якщо $t \in [5;8]$; $ht = 0,25$
11	$Y = \frac{\sqrt{m!+2k} + \prod_{i=2}^{5}(i+2k)^2}{3\sum_{i=1}^{7}(a^i + 2x^2 + k!)};$ якщо $x \in [3;k]$; $hx = 0,5k$	$F = \sqrt[3]{\frac{b+2m!-4x^2}{3\sum_{i=2}^{7}(i+m)^2}};$ $R = \frac{a}{F+x} + 3,4x^2 + \prod_{i=2}^{9}\frac{a}{i+1};$ якщо $x \in [5;10]$; $hx = 0,5$
12	$R = \frac{k!+1}{t+2k} + \prod_{i=1}^{5} i^3 + \frac{t^k}{2\sum_{i=1}^{3}(i+a)^2};$ якщо $t \in [12;15]$; $ht = 0,5$	$Q = \frac{2t}{r} + \frac{a}{m!} + \frac{b+x}{(k+2)!};$ $Z = ax\sum_{i=1}^{5}(i+m)^2 + 3,7ct;$ якщо $t \in [2;6]$; $h_t = 0,25$
13	$Y = \frac{k^{x+1}}{\left(\prod_{i=1}^{3} i^2\right) + x} + \frac{m!+x^5+(m+2)!}{k!+2x+3\sum_{i=2}^{7} i^5};$ якщо $x \in [4;7]$; $hx = 0,25$	$M = \sqrt{a + x\sqrt{c + \ln^2 x}};$ $T = \frac{2n!}{k\prod_{i=2}^{7} i^2} + \lg\left(\frac{x}{n!}\right)^3;$ якщо $x \in [12;31]$; $hx = 0,5n$

Продовження табл. Д.7

1	2	3
14	$R = \frac{x^2 + \lg(m!+3k!)}{\sqrt[3]{x + m!+k!} - a^x};$ $якщо\ x \in [3,2;6,1];$ $h_x = 0,1$	$D = \frac{a}{2m+x} + \prod_{i=1}^{9} \frac{c}{i+2} + \sqrt{m!};$ $Q = 3,2mx^2 + \sqrt{\sum_{i=4}^{9} (i+a)^3};$ $якщо\ x \in [2;6];\ hx = 0,2$
15	$B = (2P)!;$ $P = x^3 + \sqrt{ax^2 + mt + 1};$ $x = \sum_{i=1}^{15} (i + \sin i)$	$Y = \frac{P+2a}{2m-x} + \frac{\sqrt{m!+1}}{a-x} - \frac{(m+5)!}{S+x};$ $P = \prod_{i=1}^{3} (i+2);$ $S = \sum_{j=5}^{2m} (j+a)^2;$ $якщо\ x \in [3;5]; hx = 0,1$

Завдання 7

Створити сторінку, де виконуються обчислення згідно з варіантом завдання (табл. Д.8), та вивести результат на Web-сторінку. Значення незалежних змінних, що не вказані в умовах задач, вважати довільним.

Таблиця Д.8

Номер варіанта	Підзавдання 7.1	Підзавдання 7.2
1	2	3
1	$Y = \left(\frac{(a+b)^5 + \ln 2x}{m + a + b - 2.3x} \right)^m;$ $Z = (y-a)^2 + \ln \frac{x}{a};$ $a \in [2;5]; h_a = 0,2;$ $x \in [b;12]; h_x = 0,15;$ $m \in [3;9]; h_m = 3$	$Z = \frac{a^x}{m!} + \frac{R-3}{(k+2)!} + \frac{(b-a)^3}{3S};$ $R = \prod_{i=3}^{5} (i!+k);$ $S = \sum_{i=1}^{7} (i!+a)^3;$ $x \in [a;5]\ ; hx = 0,15$

1	2	3
2	$T = \frac{2.5\ln x - ab^2 + e^{x^2}}{\sin 2x + q^2};$ $S = \sqrt[5]{T} + \ln 2x + x^m;$ $Q1 = 2T + \frac{b}{x} + m^{x^2};$ $x \in [2;8]; hx = 0{,}5;$ $m \in [a;b]; hm = 0{,}1a;$ $q \in [3;5]; hq = 0{,}2$	$Q = \frac{b^2 - a}{m!+x} + \sqrt{\frac{c + 2m!}{\sum_{k=3}^{7}(a + k!)^2}};$ $F = 2\sqrt{Q - c};$ $R = mx + 2F^3;$ $x \in [1;2m]; hx = 0{,}1$
3	$Z = \frac{(a+b)^x + c\ln x}{\lg 2x + e^x};$ $Q = 2m + \frac{b}{x} - \ln Z;$ $R = 3Z + 2Q - m^t;$ $x \in [1;5]; hx = 1;$ $m \in [a;b]; hm = 0{,}1a;$ $t \in [a;c]; ht = 0{,}2c;$	$Z = \frac{a\sum_{i=1}^{7}(i!+k^2) + (k+a)^x}{\sqrt{k!+x} + (k+a)!};$ $R = 3Z + ax^5;$ $x \in [2;5]; hx = 0{,}25$
4	$Y = \left(\frac{a^c}{\cos 2x} + \frac{b}{\ln x}\right)^x - lqx;$ $R = \sqrt[3]{ax + 2y - b};$ $Q = (R - 2y)^2 + 0{,}3x;$ $a \in [1;5]; ha = 0{,}1$ $b \in [3;8]; hb = 0{,}25$ $x \in [2;6]; hx = 0{,}1c$	$Q = \frac{a + mt}{\prod_{i=1}^{7}(i + 2m)} + 2\sum_{j=4}^{7}\frac{m!}{j!+2};$ $Z = \frac{Q}{a+t} + \sqrt[3]{a + mt};$ $t \in [5;8]; ht = 0{,}25$
5	$Y = \frac{2.5 + \sqrt[4]{ax^5 + 3m}}{ax + e^x - t}$ $Z = \sqrt[3]{5y + 2c - \ln ax}$ $a \in [2;8]; ha = 0{,}2$ $m \in [c;10]; hm = 0{,}1c$ $x \in [1;t]; hx = 0{,}5$	$F = \sqrt[3]{\frac{b + 2m! - 4x^2}{3\sum_{i=2}^{7}(i!+m)^2}};$ $R = \frac{a}{F + x} + 3{,}4x^2 + \prod_{i=2}^{9}\frac{a}{i!+1};$ $x \in [5;10]; hx = 0{,}5$

1	2	3
6	$Y = \left(\frac{a+c}{\ln x} + \frac{b-c}{\ln t} \right)^{x-t};$ $R = \sqrt[3]{2y + 0.75x^t};$ $x \in [a;c]; hx = 0,2a$ $t \in [2;12]; ht = 2$ $b \in [5;c]; hb = 0,1c$	$Q = \frac{2t}{r} + \frac{a}{m!} + \frac{b+x}{(k+2)!};$ $Z = ax \sum_{i=1}^{5} (i!+m)^2 + 3,7ct;$ $t \in [2;6]; ht = 0,25$
7	$Y = \frac{\sqrt[3]{abx} + 2mq - 0.7a}{(1.3ax+b)^2};$ $Q = \ln y - \sqrt{b+2Y};$ $a \in [c;m]; ha = 0,2$ $b \in [c;12]; hb = 0,1c$ $x \in [2;8]; hx = 0,5$	$M = \sqrt{a + x\sqrt{c + \ln^2 x}};$ $T = \frac{2n!}{k \prod_{i=2}^{7} (i!+a^2)} + \lg \sum_{i=1}^{n} \left(\frac{x}{a+i!} \right)^3;$ $x \in [12;31]; hx = 0,5n$
8	$Y = \left(\frac{x}{m} + \frac{\ln a}{x} - \frac{a+b}{t} \right)^2$ $S = \sqrt[5]{\sin 2x + at - b}$ $m \in [1;5]; hm = 1$ $x \in [0.1;0.5]; hx = 0,1$ $a \in [2;6]; ha = 2$	$S = \ln \left(b + 2 \sum_{i=1}^{3} a \prod_{j=2}^{5} (i+j) \right);$ $b \in [1;5]; hb = 0,1$
9	$R = \sqrt[4]{\frac{a + 2qx^3 + m}{\ln a + e^{-x}}}$ $Q = 2,7R + \ln 2x$ $S = (Q - \ln R)^2 + 3,2x$ $m \in [2;8]; hm = 1$ $x \in [8;q]; hx = 0,2$ $a \in [1;3]; ha = 0,5$	$D = \frac{3ac + m!+c \sum_{i=2}^{5} a^i}{m \sum_{i=1}^{5} 2c \sum_{j=3}^{6} (ai+bj)}$
10	$S = \sqrt[4]{\frac{ax^5 + 3m}{2+a}} + e^{x^2} - b^{\sqrt{x}};$ $Q = arctgS + \lg(b+x)$ $x \in [12;16]; hx = 0,5$ $b \in [0,3;0,9]; hb = 0,3$	$T = 2m \sum_{i=2}^{9} c \prod_{j=7}^{1} \left(\frac{i^2+2}{j!+1} \right)^3;$ $Q = \sqrt{T+c};$ $m \in [3;12]; hm = 0,5$

1	2	3
11	$Y = \dfrac{(a+b)^m + m^x}{\ln(a+mx)}$ $Z = \dfrac{\sin y + 3x}{\ln x}$ $R = \left(\dfrac{Y+Z}{x}\right)^2 + \left(\dfrac{Z-Y}{3}\right)^3$ $a \in [1;3]; h_a = 1$ $b \in [7;15]; h_b = 0{,}5$ $x \in [2;4]; h_x = 1$ $m \in [3;5]; h_m = 0{,}1$	$F = \dfrac{\sqrt[3]{(m+a)!} + 2{,}2x^2 + \sin x}{\lg^2 x + \ln^2 x};$ $Z = \sqrt{F} + \sum_{i=3}^{9}\left(i! + a^2\right) + \sqrt{x};$ $R = a + Z - 13{,}5x,$ $x \in [2;9]; h_x = 0{,}1a$
12	$S = \dfrac{tx^2 + c\sqrt{a + \ln x}}{t + arctgx + e^x}$ $Q = 5S + \lg x^2 + ct$ $Z = \sqrt[3]{\dfrac{a+S}{m}} + \dfrac{b}{t}$ $x \in [3;8]; h_x = 1$ $t \in [a;c]; h_t = 2$ $m \in [2;6]; h_m = 0{,}1a$	$R = \sqrt{\prod_{i=2}^{9} \dfrac{i+a}{2} + 3f^3};$ $Z = \ln(a+R) + \lg\left(\sum_{i=3}^{5}(i+1)!\right);$ $Q = R + fZ - 5a;$ $f \in [2;10]; h_f = 0{,}2a$
13	$Y = \dfrac{3.2ab + e^x}{\sqrt{b+3cx}} - x^5$ $Z = 2\ln y + \dfrac{b}{\sqrt{x-c}} + m^3$ $Q = 3ay + \dfrac{b}{t} + \ln x$ $x \in [a;b]; h_x = 0{,}1a$ $c \in [2;5]; h_c = 1$ $m \in [5;12]; h_m = 0{,}5$	$L = \dfrac{1}{2x} + \dfrac{2}{m!} + \dfrac{(m+2)!}{a+x};$ $S = x\prod_{l=1}^{7}(i!-m)^2 + 3{,}45c;$ $x \in [a;2c]; h_x = 0{,}25$

Продовження табл. Д.8

1	2	3
14	$Y = \frac{ab + x^2}{m^2 + t^2} - \sqrt{mb} + c$ $Z = \ln b + 2\sin Y - a$ $x \in [1;3]; hx = 0{,}5$ $t \in [0.5;1]; ht = 0{,}25$ $m \in [a;b]; hm = 0{,}1c$	$Z = \prod_{i=1}^{7}(i!+2a^2) + 3x\sum_{k=2}^{12} k^5;$ $R = 3\sqrt{Z} + m!x - 17{,}2;$ $x \in [a;m]; hx = 0{,}125$
15	$Y = \frac{mb^2 + 2px + \cos^2 x}{at}$ $Q = \sqrt[4]{a + 2Y - 3t} + bx^2$ $a \in [1;b]; ha = 0{,}1b$ $m \in [b;10]; hm = 1$ $x \in [3;5]; hx = 0{,}2$	$Z = \frac{2m-1}{k!+x} + \frac{\prod_{i=1}^{7}(i!+2a)^2}{(m+k)!} + 2^{x+m};$ $Q = aZ + 2mx;$ $x \in [m;k]; hx = 1$

Завдання до розділу 4. Мова програмування PHP

Завдання 1. Створити сторінку, що здійснює обчислення згідно з варіантом завдання. Обчислити функцію $y=a*x^3–b*x^2+c*x–d = f(x)$ на відрізку, який задається відповідно до варіанта, де a, b, c, d – константи, значення яких задаються або вибираються користувачем. Результат виводиться з використанням масивів. Значення величин, які обчислюються згідно з варіантом завдання (max, min або ін.), вивести окремо від масиву (табл. Д.9).

Таблиця Д.9

Номер варіанта	Завдання
1	2
1	Для функції y=f(x) визначити, чи перебуває max значення функції на відрізку x∈[5, 12] на 15 кроці, якщо *x* змінюється з кроком h_x
2	Для функції y=f(x) знайти min на відрізку x∈[s, w] з кроком 0.1b

Продовження табл. Д.9

1	2
3	Для функції y=f(x) знайти номер кроку, на якому перебуває max, на відрізку x∈[k, l], якщо *x* змінюється з кроком 0.1w
4	Для функції y=f(x) визначити, чи є max значення функції додатним числом на відрізку x∈[q, g], якщо *x* змінюється з кроком 0.15
5	Для функції y=f(x) знайти значення аргументу, при якому досягається max, на відрізку x∈[m, n], якщо *x* змінюється з кроком 0.25
6	Для функції y=f(x) знайти суму значень номерів кроків, при яких досягається max та min значення функції на відрізку x∈[m, n], якщо *x* змінюється з кроком 0.01m
7	Для функції y=f(x) знайти значення аргументу, при якому досягається max, на відрізку x∈[m, n], якщо *x* змінюється з кроком 0.25
8	Для функції y=f(x) визначити, чи є min значення функції від'ємним числом на відрізку x∈[t, j], якщо *x* змінюється з кроком 0.25
9	Для функції y=f(x) знайти номер кроку, на якому перебуває min, на відрізку x∈[r, f], *x* змінюється з кроком 0.1a
10	Для функції y=f(x) визначити, у скільки разів max значення функції більше min значення на відрізку x∈[q, l], якщо *x* змінюється з кроком 0.05c
11	Для функції y=f(x) визначити різницю між значеннями аргументів, при яких досягається max значення функції та min значення функції на відрізку x∈[z, t], якщо *x* змінюється з кроком 0.025d
12	Для функції y=f(x) знайти суму значень аргументів, при яких досягається max та min значення функції на відрізку x∈[r, s], якщо *x* змінюється з кроком 0.01r
13	Для функції y=f(x) знайти значення аргументу, при якому досягається min, на відрізку x∈[k, t], якщо *x* змінюється з кроком 0.05
14	Для функції y=f(x) знайти max на відрізку x∈[k, l] з кроком 0.1t
15	Для функції z=f(x) визначити, чи перебуває min значення функції на відрізку x∈[t, n] на 10 кроці, якщо *x* змінюється з кроком 0.15m