Навчально-методична література

Міністерство освіти і науки, молоді та спорту України
Тернопільський національний технічний університет
імені Івана Пулюя

Кафедра
комп'ютерно-інтегрованих
технологій

Методичні вказівки
до виконання лабораторних робіт

з дисципліни

"Web-технології в системах автоматизації"

Тернопіль 2024

УДК 004.55
ББК 32.97

Методичні вказівки до виконання практичних робіт з дисципліни “Web-технології в системах автоматизації” (для студентів спеціальностей 174 “Автоматизація та комп’ютерно-інтегровані технології”).

Розробили доцент, к.т.н. Чихіра І.В.
доцент, к.т.н. Левицький В.В.

Рецензент доцент, к.т.н. Коноваленко І.В.

Методичні вказівки розглянуті на засіданні кафедри.
Протокол № 1 від 26 серпня 2024р.

Схвалено і рекомендовано до друку науково-методичною комісією факультету прикладних інформаційних технологій та електроінженерії Тернопільськтго національного технічного університету ім.Івана Пулюя
Протокол № 1 від 30 серпня 2024р.

Лабораторна робота №1

Тема: Встановлення та налаштування WEB сервера та PHP.
Мета: Ознайомитися з основними можливостями Web-сервера. Навчитись налаштовувати сервер для роботи в різних режимах.

ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Web-сервер – це набір програм, який забезпечує обмін даними через протокол передачі гіпертексту (HTTP – Hyper Text Transfer Protocol). На сьогодні найбільш поширеним серед Web-серверів є Apache. Сервер працює під управлінням таких ОС як Linux, Windows. Головний виконуваний файл працює як системна служба. Його копія завантажується в оперативну пам'ять при кожному звертанні до сервера. Система налаштувань, яка базується на файлах конфігурації, зазвичай сервер не має системи налаштувань із графічним інтерфейсом користувача.

Головний файл конфігурації Apache має назву httpd.conf. Залежно від версії системи цей файл може перебувати в різних каталогах, але формат його залишається незмінним. Рядки файлу httpd.conf, що починаються із символу #, містять коментарі. Опції, що визначають конфігурацію сервера, задаються в вигляді: *Директива Значення.* Деякі директиви дозволяють задавати декілька підопцій. У цьому випадку ім'я директиви міститься в кутових дужках, наприклад:

```
<Directory  /home/httpd/html>
   Options  FollowSymLinks
   AllowOverride  None
</Directory>
```

Зазначена директива стосується каталогів, а директива <Files> встановлює налаштування для файлів.

За замовчуванням сервер завантажується за допомогою програми або сценарію, хоча в ОС Linux існує можливість його завантаження за допомогою багатоскладового сервера (суперсервера) inetd. Під управлінням ОС Windows сервер завантаження зупинка та пере запуск сервера здійснюється за допомогою відповідних ярликів головного меню. У ОС Linux зазначені операції здійснюються виконанням сценарію /etc/init.d/httpd з відповідними параметрами (start stop restart). Зверніть увагу на те, що Web-сервер слід перезавантажити після кожного внесення змін у конфігураційні файли.

Налаштування загального призначення

Для більш ефективного опрацювання запитів клієнтів ОС завантажує декілька копій Apache. Кожен екземпляр опрацьовує окремий запит. З метою ефективного опрацювання запитів клієнтів використовують директиви MinSpareServers та MaxSpareServers Якщо число дочірніх процесів менше за MinSpareServers і вони не виконують опрацювання запитів, то у випадку надходження нових запитів ОС буде завантажувати нов. підпроцеси. Аналогічно, у випадку перевищення копій Apache числа MaxSpareServers нові копії завантажуватися не будуть.

MaxClients – задає максимальну кількість користувачів, що можуть одночасно звернутися до сервера. Примітка MaxClients – це не кількість браузерів, оскільки в межах однієї сторінки може бути реалізовано декілька з'єднань із сервером. Час очікування, після якого сервер відправить повідомлення про тайм аут встановлюється директивою TimeOut.

Listen – за замовчуванням Apache опрацьовує звернення до активних мережевих інтерфейсів, з використанням порту 80. Зазначена директива дає змогу обмежити звертання до сервера за інтерфейсами та портами. Наприклад, директива 172.25.62.19:8080 визначає, те що сервер працюватиме з інтерфейсом з адресою 172.25.62.19 та портом 8080.

BindAddress. Якщо комп'ютер, на якому виконується сервер Apache, містить кілька мережевих інтерфейсів, то, використовуючи дану директиву, можна організувати роботу сервера лише з одним з

інтерфейсів. По замовчуванню використовується директива BirsdAddress *, що відповідає роботі з усіма інтерфеймами.

Port Дана директива вказує Apache, який порт повинен використатися для взаємодії із клієнтами. За замовчуванням приймається номер порту 80.

> **Примітка:** директиву Listen можна використовувати декілька разів, налаштовуючи сервер на роботу з різними IP-адресами та портами.

ServerAdmin. За допомогою даної директиви можна вказати свою поштову адресу, яка використовується у повідомленнях про помилку.

ServerName. Якщо значення даної директиви відрізняється від імені комп'ютера, можете усунути невідповідність, задавши правильне значення.

Опис каталогів

ServerRoot. За допомогою цієї директиви задається корінь піддерева файлової системи, який використовується для зберігання двійкових файлів Apache.

DocumentRoot. У каталозі, зазначеному за допомогою цієї директиви, знаходяться Web-сторінки. За замовчуванням для даної опції задається "/var/www/html"

> **Примітка**. Значення директиви DocumentRoot не слід завершувати косою рискою, оскільки вона може привести виникнення помилки.

Directory/Index. Деякі URL не містять назву файлу; у них зазначене лише ім'я каталогу (у деяких випадках воно завершується косою рисою). Коли сервер Apache одержує подібний URL, він спочатку намагається знайти *файл індексу,* який задається даною директивою. За замовчуванням приймається назва файлу index.html. Якщо задано декілька файлів індексу, Apache по черзі здійснює їх пошук відповідно до вказаного порядку. Якщо файл індексу не заданий, то сервер повертає, залежно від налаштувань, вміст каталогу або повідомлення про помилку.

Для забезпечення автоматичного генерування індексу каталогу для нього слід зазначити директиву Indexes

Приклад

```
<Directory "/var/www/html">
   Options  Indexes
</Directory>
```

Для виводу вмісту каталогів із розширеним вмістом слід використати директиву IndexOptions FansyIndex.

Параметри налаштувань для окремих каталогів

Для конфігурування деяких параметрів використовується директива Options, значення якої можуть бути такими:

All – в каталозі дозволено все крім параметрів MultiViews

Exec CGI – дозволено виконання сценаріїв CGI

FollowSymLinks – сервер здійснює перехід за символьними посиланнями

Includes – дозволено включення вставок зі сторони сервера

MultiViews – якщо файлу, який намагається отримати клієнт не існує то сервер обере документ який максимально відповідає запиту

SymLinksIfOwnerMatch – сервер дозволяє перехід за символьними посиланнями у випадку, якщо каталог призначення належить тому ж користувачеві що і саме посилання

Для визначення параметрів кожному каталозі може бути використаний файл .htaccess, який може містити всі ті ж опції що і директива <Directory Для уникнення конфліктів між параметрами вказаними в конфігураційному файлі та у файлі .htaccess у кожній секції Directory файлу httpd.conf можна вказати директиву AllowOverride, яка визначає клас директив які можуть використовуватися у файлах htaccess.

Аргументами директиви AllowOverride можуть бути

All – у файлах .htaccess можуть використовуватись всі директиви

AuthConfig - дозволяється використання директив, які управляють аутинтифікацією

FileInfo - дозволяється використання директив, які відносяться до типів, кодувань файлів

Indexes - дозволяється використання директив, пов”язаних з індексуванням

Limit - дозволяється використання директив пов”язаних із захистом каталогів

Options - дозволяється використання директив Option

None – не дозволяється використання файлів .htaccess

У файлі .htaccess може бути проведена авторизація на рівні ip-адрес чи доммених імен

Приклад файлу .htaccess

```
allow form .tspu.edu.ua
deny 10.33.1.66
```

Як наслідок створення такого файлу до каталогу в якому він розміщений буде заборонено доступ вузлу 10.33.1.66 і дозволено всім вузлам з домену .tspu.edu.ua

У файлі .htaccess може бути проведена авторизація на рівні користувачів чи їх груп

Приклад файлу .htaccess

```
AuthType Basic
AuthUserFile /etc/httpd/htusers
AuthName "Введіть ім”я і пароль "
require valid-user
```

У першій стрічці вказано тип аутентифікації –
стандартний для ОС, друга стрічка задає файл користувачів та паролів, третя – підказка. Остання задає правила доступу: доступ до каталогу матимуть усі користувачі з файлу htusers.

Незважаючи на те що статичні дані часто використаються на Web-вузлах, типи даних, з якими може працювати Web-сервер, не вичерпуються ними. На сьогодні розроблені мови програмування, транслятори яких виконуються як модулі Web-сервера, дають змогу динамічно формувати Web-сторінки.

PHP- сценарії – скрипти, які виконуються інтерпретатором PHP на сервері. Після встановлення сервера під управлінням ОС Linux, як правило, інтерпритатор працює без додатковиз налаштувань. У ОС Windows для налаштування сервера Apache для роботи з PHP необхідно:

-обумовити тип файлів php: **AddType application/x-httpd-php php**

-визначити за допомогою якої програми опрацьовуватимуться файли описаного типу

```
ScriptAlias /_php/ "c:/Program Files/php4/"
Action  application/x-httpd-php "/_php/php.exe"
```

При налагодженні Apache у конфігураційний файл включається директива ScriptAlias, яка відображає си нонім для папки.

Використання віртуальних доменів

Розвиток мережі Інтернет призвів до того, що один комп'ютер виконує роль сервера для більш ніж одного Web-вузла. Ситуація, коли на одному сервері функціонують декілька вузлів називається віртуальним Web-хостингом. Для налаштування віртуальних вузлів використовуються декілька методів

-для кожного вузла виділяється окрема IP-адреса

-використання однієї IP-адреси для всіх вузлів

Віртуальний вузол не залежно від методу повинен бути описаний за допомогою секції <VirtualHost>, синтаксис якої наступний:

```
<VirtualHost  IP-адреса сервера [:порт]>
   DocumentRoot кореневий каталог документів
   ServerName імя вузла
   ServerAdmin адреса електронної пошти  адміністратора вузла
   ErrorLog файл журналу помилок звертання до  вузла
   CustomLog файл журналу звертання до вузла
 </VirtualHost>
```

У директиві <VirtualHost замість IP-адреси можна використовувати символ *, який визначає використання для даного вузла значень для сервера по замовчуванню. Крім цього в секції можуть бути визначені й інші директиви, які встановлюютьнаступні параметри для вузла:

-обробка сценаріїв

-визначення параметрів індексування каталогів

-управління доступом і авторизація

У випадку використання однієї IP-адреси для декількох вузлів імя вузла має бути зареєстрованим за допомогою сервера DNS.

Приклад опишемо віртуальний Web-вузол lab.school.te.ua, який опрацьовувати запити за номером порту 8080

```
<VirtualHost *:8080>
   ServerAdmin webmaster@school.te.ua
   DocumentRoot /var/www/vhosts/lab
   ServerName lab.school.te.ua
   ErrorLog  logs/lab.school.te.ua-error_log
 </VirtualHost>
```

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

1. Скачайте наступні файли:

apache_1.3.26-win32-x86-no_src.exe

php-4.2.2-installer.exe

php-4.2.2-Win32.zip

Встановлення сервера Apache

2. Запустіть отриманий файл дистрибутива Apache. У з'явившомуся діалоговому вікні натисніть кнопку **Next** , а потім — кнопку **Yes**, щоб погодитися з умовами ліцензії.

3. Натискайте кнопку **Next** у вікнах, що відкриваються, доти, поки не з'явиться запит про вибір каталогу для встановлення Apache. За замовчуванням записаний шлях "C:\Program Files\Apache Group\Apache". Це нас абсолютно не влаштовує -
потрібно установити Apache на свій диск Z. Для цього натискаємо "Browse" і вказуємо шлях "Z:\usr\apache"
(якщо такого каталогу неіснує, система запитає чи потрібно його створити, на що дайте ствердну відповідь). Натискаємо "**NEXT**".

4. У вікні, що з'явилося, встановіть **прапорець**Typical (Звичайна) і натисніть кнопку **Next**.

5. Програма інсталяції Apache запропонує створити папку в меню **Пуск** у папці **Програми**. Дозвольте їй це зробити, натиснувши кнопку **Next**. Почнеться процес копіювання програмного забезпечення.

6. Після закінчення копіювання натисніть кнопку **Finish**. Процес встановлення сервера завершено.

Налаштування сервера Apache

7. Відкриваємо файл "Z:\usr\apache\conf\httpd.conf" у якому-небудь текстовому редакторі, наприклад "Блокноті". Це дуже важливий файл, що містить усі налаштування Вашого сервера. У ньому необхідно задати кілька параметрів. Почнемо.

Знайдіть у файлі httpd.conf стрічку "#ServerName new.host.name". Замініть її на рядок "Servername localhost". Зверніть увагу, що знак "#" на початку рядка забирається.

Далі знайдіть стрічку "DocumentRoot "Z:/usr/apache/htdocs" і замініть його на "DocumentRoot "Z:/project/www". Зверніть увагу на відсутність слеша наприкінці. (це ви вказали каталог, в якому буть знаходитись ваші файли)

Тепер знайдіть стрічку "<Directory "Z:/usr/apache/htdocs">" і замініть її на <Directory "Z:/project/www">. Слеш знову відсутній.

Ініціалізуйте параметр DirectoryIndex так:
DirectoryIndex index.htm index.html index.php

Також знайдіть і замініть стрічку "ErrorLog logs/error.log" на стрічку "ErrorLog Z:/project/logs/error.log" і стрічку "CustomLog logs/access.log common" на "CustomLog Z:/project/logs/access.log common".

А тепер створіть каталог "Z:\project", і в ньому два каталоги "www" і "logs". У підсумку Ви повинні отримати таке дерево каталогів:
Z:\project
Z:\project\www
Z:\project\logs

Знайдіть і виправте наступний параметр: ScriptAlias /cgi-bin/ "z:/project/cgi/" Додайте після нього ще такий рядок:ScriptAlias /cgi/ "z:/project/cgi/" Це буде той каталог, у якому повинні розташовуватися ваші CGI-сценарії. Подібний параметр говорить Apache про те, що, якщо буде зазначений шлях виду http://localhost/cgi-bin, то насправді варто звернутися до каталогу z:/project/cgi.

Знайдіть і налаштуйте (не забудьте розкрити коментар!) наступний параметр:
AddHandler cgi-script .bat .exe .cgi
Він говорить Apache про те, що файли з розширеннями exe, bat і cgi треба розглядати як CGI-модулі.

І останнє — встановіть наступні параметри:
AddType text/html .shtmlAddHandler server-parsed .shtml .html .htm
Цим ви змушуєте Apache обробляти файли з зазначеними розширеннямипроцесором SSI.

Теперь не забудьте зберегти зміни і закрийте **Блокнот**.

Тестування сервера Apache

Перевірка html

8. Створіть в папці "Z:\project\www\" файл з ім’ям "index.html" наступного змісту:

```
<html>
  <head>
    <title>Головна сторінка сервера</title>
  </head>
  <body bgcolor=#ffffff>
    Вітаю Вас, сервер працює!<br>
```

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

9. Тепер запустіть браузер і наберіть: http://localhost/index.html. Повинен завантажитись ваш файл

Перевірка SSI

10. В каталозі "Z:\project\www\" з HTML-документами Apache створіть файл test.shtml наступного змісту (уважно слідкуйте за дотриманням пробілів в директорії include!):

```
SSI Test!<hr>
<!--#include virtual="/index.html" -->
<hr>
11. Тепер наберіть в браузері:
http://localhost/test.shtml
```

Повинен відкритися файл, який містить текст "SSI Test!", за яким слідує вмістиме файла index.html між двома горизонтальними лініями. Якщо цього не відбулося, значить, ви невірно сконфігурували SSI.

Перевірка CGI

11. В каталозі "Z:\project\www\cgi", призначеному для збереження CGI-сценаріїв, створіть файл test.bat з таким змістом:

```
@echo off
 echo Content-type: text/html
 echo.
 echo.
 Dir
```

12. Тепер в браузері наберіть: http://localhost/cgi-bin/test.bat. В вікні відобразиться результат команди DOS dir.

Встановлення PHP

13. Запустіть завантажений exe-файл (php-4.2.2-installer.exe). У діалоговому вікні, що відкрилося натисніть кнопку **Next.**

14. Погодьтеся з умовами ліцензії, натиснувши кнопку **I Agree**. У діалоговому вікні, що з'явилося виберіть тип установки **Standard**.

15. Тепер вкажіть каталог, в який буде встановлено PHP. За замовчуванням пропонується C:\PHP, але, логічніше було б вибрати Z:\usr\php. Для вказівки цього каталога натисніть кнопку **Browse...** і введіть його ім'я, потім натисніть, як звичайно, кпопку **OK** і потім — **Next**, щоб перейти до наступного діалогового вікна.

16. Задайте адресу вашого SMTP-сервера (Send Mail Transfer Protocol — Протокол пересилки поштової кореспонденції), а також адресу вашої електронної пошти. Саме цей сервер і зворотна адреса будуть використані для вихідних почтових запитів, коли викликається функція Mail() мови PHP. Загалом, це той самий сервер, через який відсилає пошту ваш звичайний поштовий клієнт —наприклад, Outlook Express. Утім, можете і залишити в текстових полях значення по замовчуванні — у цьому випадку функція Mail() просто не буде працювати на локальній машині.

17. Виберіть сервер, на який буде надбудований PHP. У нашому випадку це — Apache. Почнеться процес копіювання файлів. Після його закінчення, можливо, з'являться ще деякі діалогові вікна з різними повідомленнями. Не звертайте наних уваги.

На цьому етапі PHP можна вважати вже майже встановленим — нам залишилося тільки налаштувати Apache, щоб він міг розпізнати PHP-сценарії, а також подключити додаткові модулі, що знаходяться в завантаженому вами zip-архіві.

18. Розархівуйте zip-архів прямо в той же самий каталог, де вже встановлений PHP (в нашому прикладі це Z:\usr\php). Деякі файли перекриються, деякі — додадуться. Зокрема, з'явиться каталог extensions, який містить практично всі необхідні файли.

19. Тепер потрібно дати знати PHP, які модулі він може використовувати, а також здійснити ще деякі налаштування. Для цього відкрийте в **Блокноті** файл php.ini з каталогу з файлами Windows (звичайно C:\WINDOWS). Цей файл був поміщений туди програмою установки PHP. Файл являє собою набір рядків, кожен з яких відповідає значенню одного параметра. Частини рядків, що розміщені після символу ;, розглядаються як коментарі й ігноруються.

Знайдіть параметр magic_quotes_gpc і відключите його: magic_quotes_gpc=Off
Цим ми забороняємо PHP примусово вставляти зворотні слеші перед деякими символами, що надходять з форми.

Тепер знайдіть і нналаштуйте наступний параметр: extension_dir=C:\Program Files\PHP4\extensions
Тут ми повідомляємо PHP, що модулі він повинен шукати в каталозі C:\Program Files\PHP4\extensions, тобто саме там, де потрібно. Зверніть увагу на те, що за замовчуванням у цьому параметрі стоїть значення ./, тобто виклик буде відбуватися в тому ж самому каталозі, де встановлений PHP. Це, донечно ж, незручно.

Знайдіть "закоментовані" рядки, що починаються з ;extension=. Вам потрібно розкрити ті з них, що відповідають потрібним нам модулям.

Не забудьте зберегти зміни у файлі php.ini. Щоб зміни вступили в силу, перезапускати Apache не потрібно, адже ми встановили PHP не як модуль сірий-віра, а як окрему програму

Налаштування сервера Apache для роботи з PHP

20. Відкрийте в **Блокноті** файл конфігурації Apache httpd.conf, що знаходиться в каталозі “Z:\usr\apache\conf\”.

21. Знайдіть у тексті файлу таку закоментовану стрічку: #AddType application/x-httpd-php php

22. Розкрийте коментар: AddType application/x-httpd-php php

23. Тепер перейдіть у самий кінець файлу httpd.conf і впишіть у нього такі рядки:

```
ScriptAlias /php/ "f:/usr/php/"
AddType application/x-httpd-php .php .phtml  .php4
Action application/x-httpd-php  "/php/php.exe"
```

Ці рядки додають у настройки Apache можливість виконання файлів з разширенням .php, .phtml або .php4 як програм, написаних на PHP (PHP скриптів).

24. Збережіть зміни у файлі конфігурації, зупиніть Apache, якщо він був до цього запущений , і запустіть сервер знову. Якщо Apache не запускається (його вікно відкривається і відразу закривається), значить, ви десь допустили синтаксичну помилку.

Тестування PHP

25. Створіть в каталозі Z:\project\www два файли з наступними назвами і змістом:

form.html

```
<html>
  <body>
   <form  action=hello.php>
     Введіть своє ім'я: <input type=text  name="name" value="невідомий"><br>
     Введіть свій вік: <input type=text  name="age" value="невизначиний"><br>
     <input  type=submit value="Нажміть на кнопку, щоби запустити сценарій!">
```

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

hello.php

```
<html>
  <body>
   <?
      $name=$_GET['name'];
      $age=$_GET['age'];
      echo  "Привіт,   $name <br>";
      echo  "Я знаю, Вам $age років!";

  </body>
</html>
```

26. Запустіть на виконання файл **form.html** набравши в броузері адресу: http://localhost/form.html За повнивши необхідні дані отриманої форми і натиснення кнопки, повинен завантажитися файл **hello.php** зі змістом: Привіт, <Ваше і'мя>. Я знаю, Вам <Ваш вік> років!

Якщо отриманий саме такий результат, то PHP налаштований вірно.

КОНТРОЛЬНІ ЗАПИТАННЯ

Чи можна вважати Web-сервером програму, яка працюватиме із даними через 80 порт?

Пояснити різницю між методами опрацювання Web-сервером html- та php-файлів.

Якщо директиви, які зазначені для даного каталогу та у файлі .htaccess суперечать одна одній, то яке правило виконуватиметься?

У чому полягає різниця між параметрами MaxClients та MaxSpareServers?

Лабораторна робота №2

Тема: Основи роботи з HTML – документами.
Мета: Ознайомитися з основними тегами форматування тексту гіпертекстової розмітки мови HTML.

ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Види тегів в HTML

HTML-документ являє собою текстовий ASCII-файл, що містить сам текст, що повинен бути відображений у вікні Браузера, і команди розмітки – HTML-теги, що визначають зовнішній вигляд документа при його інтерпретації у вікні Браузера.

HTML-тег записується в кутових дужках (<>) і складається з імені, за яким може бути поміщений список атрибутів (для більшості тегов необов'язковий). Імена й атрибути являють собою англійські слова й абревіатури і майже завжди їхній зміст прозорий. Записувати теги можна в будь-якому регістрі - прописними або малими літерами.

Теги можна розділити на дві великі групи.Теги однієї групи, називаються *контейнерами*, впливають на частину документа, вкладену між ними. Вони мають два компоненти: відкриваючий (початковий) і закриваючий (кінцевий). Закриваючий Тег має таку ж назву, що і відкриваючий, але перед його назвою ставиться коса риса (символ /). Між відкриваючим і закриваючим тегами може розташовуватися текст або інші теги. *Автономні* (одиночні) теги не мають кінцевого компоненту. Вони викликають одноразову дію або при їхній інтерпретації у відображуваний документ вставляється той або інший об'єкт. Наприклад, тег <IMG SRC="pict.gif">, викликає вставку малюнка з файлу pict.gif.

Теги можуть мати уточнюючі параметри – атрибути. Атрибути записуються всередині автономного тега, а в контейнері тільки у відкриваючій частині. У списку атрибути відокремлюються один від одного пробілами. Послідовність атрибутів не істотна. Значення атрибутів вказуються після знака рівності в лапках. Лапки потрібно ставити обов'язково, якщо значення атрибута містить пробіли.

Приклади тегів з атрибутами:
<BODY BGCOLOR="LIGHTBLUE"> - задає ясно-синє тло для документа,
<FONT SIZE=+2 COLOR=RED> текст </FONT> - парний тег, дає вказівку Браузерові

вивести поміщений у «контейнер» текст символами, збільшеними щодо базового розміру (SIZE=+2) і червоний кольори (COLOR=RED).

Теги, що визначають структуру HTML-документа

HTML-документ поміщається в теги <HTML> і </HTML>. Між цими тегами розташовуються два розділи: розділ *заголовка* (між тегами <HEAD> і </HEAD>*)* і розділ *тіла документа* (між тегами <BODY> і </BODY>*).*

Розділ заголовка містить опис параметрів, використовуваних при відображенні документа, але не відображаються безпосередньо у вікні переглядача.

Розділ тіла документа містить текст, призначений для відображення переглядача і теги, що вказують на спосіб форматування тексту, що визначають графічне оформлення документа, що задають параметри гіперпосилань і так далі.

Наприклад:

```
<HTML>
  <HEAD>
   <TITLE>Приклад HTML-документа</TITLE>
  </HEAD>
  <BODY>
   Найпростіший HTML-документ
  </BODY>
</HTML>
```

Браузер відобразить цей документ, вивівши у своєму вікні рядок тексту: Найпростіший HTML-документ, розташований у секції тіла документа. Завдяки наявності тега <TITLE> у заголовку вікна Браузера буде виведене не ім'я файлу, а заголовок, що звичайно несе зазначений зміст. У даному випадку заголовок вікна Браузера буде: Приклад HTML-документа.

Основні теги HTML

Основні теги мови HTML наведені в таблиці 1.

Таблиця 1. Основні теги мови HTML

Призначення	**Вид тегу**	**Примітка**
Загальна структура документа HTML		
Тип документа	<HTML></HTML>	Початок і кінець.
Заголовок документа	<HEAD></HEAD>	Не відображається Браузером.
Ім’я документу	<TITLE></TITLE>	Відображається в рядку заголовка вікна Браузера.
Тіло	<BODY></BODY>	
Коментар	<! --***-->	Оглядачем ігнорується.
Структура змісту документа		
Внутрішні заголовки різного рівня	<Hn></Hn>	Де n – номер рівня заголовка (від 1 до 6-ти).
Заголовок з вирівнюванням	<Hn ALIGN=”left”> </Hn>	Типи вирівнювання: ▪ left - по лівому краю, ▪ center - по центрі, ▪ right - по правому краю, ▪ justify – по ширині.
Форматування абзаців		
Створення абзацу	<P></P>	

(параграфа)		
Розрив рядка всередині абзацу	 	Одиночний тег
Перенос	<WBR>	
Вирівнювання абзацу	<P ALIGN=”left”></P>	Типи вирівнювання: ***left***, ***right***, ***center*** і ***justify***
Горизонтальна лінія між абзацами	<HR>	Одиночний тег
Атрибути: товщина, довжина лінії, вирівнювання	<HR SIZE=? WIDTH=? ALIGN=”left”>	Товщина вказується в пікселях, довжина – у пікселях або %, вирівнювання за замовчуванням center
Форматування шрифту		
Логічне форматування		
Виділення тексту: курсивом - напівжирним - Як із клавіатури -	<EM></EM> <STRONG> </STRONG> <KBD> </KBD>	*Фрагмент документу*: звичайний текст <EM>Курсив </EM> <KBD>Моноширинний</KBD>
Фізичне форматування		<B> жирний</B> <S> підкреслений</S> X² - відобразиться як X2 <I> курсив </I>
Курсив	<I> </I>	
Підкреслений	<U> </U>	
Перекреслений	<S> </S>	
Моноширний	<TT> </TT>	
Верхній індекс		
Нижній індекс		
Розмір шрифту	<FONT SIZE=?></FONT>	Від 1 до 7
Базовий розмір шрифту (задається для всього документа в цілому)	<BASEFONT SIZE=?>	Від 1 до 7, за замовчуванням дорівнює 3
Список кращих гарнітур шрифтів, через кому	<FONT FACE=”шр-т1, шр-т2”>	<FONT FACE="Verdana,Arial">
Колір шрифту (задається або ключовим словом, або шістнадцятковим кодом)	<FONT COLOR=”$$$$$$”> </FONT>	Red -червоний Blue – синій #ff0000 – 16-ковий код червоного кольору у форматі RGB
Створення списків		
Упорядкований	<OL></OL>	*Фрагмент документу*: <OL> <LI>Елемент списку 1</LI> <LI>Елемент списку 2</LI> <LI>Елемент списку 3</LI> </OL>
Неупорядкований	<UL></UL>	
Элемент списка	<LI></LI>	
Списки визначень	<DL></DL>	
Визначаємі термінали	<DT></DT>	
Визначення	<DD></DD>	
Таблиця		
Визначити таблицю	<TABLE></TABLE>	<TABLE border=”1”> <TR> <TH>Товар</TH> <TH>Ціна</TH> </TR> <TR> <TD>Радіотелефон </TD> <TD align=”center”> 2000 </TD> </TR>
Рамка таблиці	<TABLE BORDER=?> </TABLE>	
Стрічка таблиці	<TR></TR>	
Вирівнювання	<TR ALIGN=LEFT | RIGHT | CENTER | JUSTIFY >	
Комірки таблиці	<TD></TD>	
Вирівнювання по	<TD ALIGN=LEFT | RIGHT |	

горизонталі	CENTER | JUSTIFY>	</TABLE>
Ширина (в пікселях або в % до ширини вікна)	<TD WIDTH=?>	
Заголовок стовбців або стрічок (напівжирний)	<TH></TH>	
Вставка зображень		
Вставка графічного файлу	<IMG SRC=”URL” WIDHT=”?” HEIGHT=”?”>	*Фрагмент документу*: <IMG SRC=”grafica.gif” WIDHT=”550” HEIGHT=”240” ALT=”Графічний файл”>
Вирівнювання щодо тексту	<IMG SRC=”URL” ALIGN=”TOP”>	
Виведення альтернативного тексту	<IMG SRC=”URL” ALT=”текст*”>	
Вставка посилань		
Посилання на іншу сторінку	<A HREF=”URL”> текст </A>	*Фрагмент документа*: <A HREF=”str2.htm”> Посилання1</A> <A HREF=”URL”>текст</A> <A NAME=”metka1”></A> <A HREF=”index.htm#metka1”> На головну сторінку</A>”
Посилання на закладку в іншому документі	<A HREF=”URL#*”> текст</A>	
На закладку в тому ж документі	<A HREF=”#*”> текст</A>	
Визначити закладку	<A NAME=”*”></A>	
Колір тла, тексту і посилань		
Фонова картинка	<BODY BACKGROUND=”URL”>	*Фрагмент тексту*: <BODY BACKGROUND=”grafica.gif” TEXT=”black” (чорний) LINK=”#FF0000” (червоний) VLINK=”#FFFF00” (жовтий) ALINK=”#FFFFFF”> (білий) ---------- </BODY>
Колір фону	<BODY BGCOLOR=”#$$$$$$”>	
Колір тексту	<BODY TEXT=”#$$$$$$”>	
Колір посилання	<BODY LINK=”#$$$$$$”>	
Колір пройденого посилання	<BODY VLINK=”#$$$$$$”>	
Колір активного посилання	<BODY ALINK=”#$$$$$$”>	

Коментарі - текст коментара поміщається в тезі <!-- …...текст коментара…... --> і не відображається у вікні Браузера.

Символьні примітиви – конструкції, що заміняють службові символи мови HTML (зазвичай починаються з амперсанта – символу &):

Числовий код	Іменна заміна	Символ	Опис
"	"	"	Лапки
&	&	&	Амперсант
<	<	<	Менше
>	>	>	Більше
			Нерозривний пробіл
¡	¡	?	Перевернений знак оклику
¢	¢	?	Цент
£	£	?	Фунт
¤	¤	¤	Валюта
¥	¥	?	Йена
¨	¨	?	Умляут
©	©	©	Копирайт
«	«	«	Ліві кутові лапки
®	®	®	Зареєстрована торговельна марка
±	±	±	Плюс або мінус

&#187;	&raquo;	»	Праві кутові лапки

Колір тексту і тло документа.

Колір основного тексту, колір гіперпосилань і тло документа описуються в початковому тезі тіла документа <BODY> за допомогою атрибутів.

Значення кольору можна задавати в одній із двох форм. Перша форма використовує задання кольору в RGB – палітрі (Red-Green-Blue). Код кольору вказується шістнадцятковими числами, що задають інтенсивність відповідної складової (по двох розрядах). Наприклад, яскраво-червоний має код – FF0000, яскраво зелений – 00FF00, чорний колір 000000, білий – FFFFFF. При заданні кольору перед шістнадцятковим числом ставиться символ #.

Можливе задання кольору за допомогою імен. Таблиця основних кольорів наведена в таблиці 2. Кожній назві кольору відповідає визначена RGB – тріада. Наприклад, кольору navy - #000080.

Таблиця 2. Основні кольори

Color's name	Назва	Red	Green	Blue
black	чорний	00	00	00
navy	темно-синій	00	00	80
blue	синій	00	00	FF
green	зелений	00	80	00
teal	синьо-зелений	00	80	80
lime	яскраво-зелений	00	FF	00
aqua	блакитний	00	FF	FF
maroon	вишневий	80	00	00
purple	фіолетовий	80	00	80
olive	маслиновий	80	80	00
gray	темно-сірий	80	80	80
silver	ясно-сірий	C0	C0	C0
red	червоний	FF	00	00
fuchsia	ліловий	FF	00	FF
yellow	жовтий	FF	FF	00
white	білий	FF	FF	FF

Теги форматування тексту

Для форматування тексту HTML-документа передбачені дві групи тегів, що називаються *логічними* і *фізичними* тегами форматування.

Теги логічної групи позначають своїми іменами структурні групи текстових фрагментів. Наприклад, тег <CODE> вказує на програмний код, тег <STRONG> - жирне виділення. Фрагменти з логічним форматуванням Браузер відображає певним чином відповідно до його можливостей.

Теги фізичного форматування вказують Браузерові як відобразити текстовий фрагмент відповідно до побажань автора. Наприклад, тег <B> використовується для відображення тексту напівжирним накресленням (що звичайно відповідає логічному тегу <STRONG>).

Сучасні Браузери підтримують і ті, і інші теги форматування. Однак з виходом специфікації HTML 4.0, перевага віддається логічному форматуванню, оскільки був проголошений принцип відділення структури документа від його представлення.

Текст виводиться у вікні Браузера нерозривно слово за словом, при цьому весь пробільний матеріал відображається як один пробіл. Тому для розриву рядка і вставки додаткового пробільного інтервалу необхідно використовувати спеціальні засоби.

- Символьний примітив   – один пробіл.
- Тег
 (від англійського break) – вставка нового рядка.

- Тег <P> (від англійського paragraph) – починає абзац з нового рядка, відокремлюючи від попереднього подвійним міжстрічковим інтервалом.
- Текст, вкладений між тегами <PRE> </PRE > (від англійського preformatted), відображається так, як він був відформатований попередньо, із усіма пробілами переносами рядків.

Заголовки різних рівнів.
Теги виду <Hn> **……</**Hn> оформляють поміщений в них текст у вигляді заголовка **n-рівня**. Значення **n** можуть змінюватися від 1 (самий великий) до 6 (самий дрібний). Так само, як і тег абзацу <P>, тег заголовка перериває текстовий потік і відокремлює його порожнім рядком.
Теги <H1>**,** <H2>**,** <H6> можуть мати атрибут вирівнювання ALIGN зі значеннями LEFT (за замовчуванням), CENTER, RIGHT і JUSTIFY.
Списки
HTML дозволяє створювати **нумеровані** і **маркіровані** списки.
Фрагмент тексту, що представляє список, поміщається в теги:

- <OL> **……** </OL> впорядкований список (ordered list),
- <UL> **……**</UL> невпорядкований список (unordered list),
- <DL> **……**</DL> список визначень (definition list).

Кожен елемент списку поміщається в теги <LI> **……**</LI> (від англійського list item).

Тег нумерованого списку може мати параметр TYPE= , що визначає вид нумерації, і START= , що задає початкове значення першого елемента списку (не залежно від типу вказується цифрою).

- TYPE=A – задає маркери у виді прописних латинських букв;
- TYPE=a – маркери – рядкові латинські букви;
- TYPE=I – маркери у виді великих римських цифр;
- TYPE=i – маркери у виді маленьких римських цифр;
- TYPE=1 – маркери – арабські цифри (за замовчуванням).

У тегах маркерованого списку параметр TYPE вказує тип маркера: зафарбовані кружечки – disc, не зафарбовані кружечки – circle, зафарбовані квадратики – square.

Графічні зображення.

Тег <IMG> вставляє зображення в текстовий потік. Закриваючого компонента цей тег не має.Обов'язковим атрибутом його є SRC=url (адреса графічного файлу, може бути відносною або абсолютною). Для прискорення завантаження Web-сторінки з малюнками рекомендується в тезі малюнка вказувати його розміри атрибутами HEIGHT і WIDTH. Це дозволяє Браузерові ще до повного завантаження малюнка виконати розмітку екрана і завантажити текст.
Основні необов'язкові атрибути наведені в табл.3.

Таблиця 3.

Атрибут	**Призначення атрибута**
ALT=текст	Альтернативний текст, виведений у режимі Браузера без завантаження зображень
BORDER= значення	Товщина рамки, що обрамляє зображення в пікселях
HEIGHT= значення	Висота зображення в пікселях або у відсотках від висоти вікна Браузера
WIDTH= значення	Ширина зображення в пікселях або у відсотках від ширини вікна Браузера
HSPACE= значення	Вільний простір ліворуч і праворуч від зображення в пікселях
VSPASE= значення	Вільний простір зверху і знизу від зображення в пікселях

ALIGN=значення	Вирівнювання зображення по горизонталі і по вертикалі. Значення задаються атрибутами TOP, MIDDLE, BOTTOM. При застосуванні цих значень вставлене зображення розриває текстовий потік. Якщо задані значення LEFT або RIGHT, зображення буде відповідним чином вирівняно по горизонталі, а текст буде обтікати його.

Гіперпосилання

Зв'язок між HTML-документами і фрагментами документів організується за допомогою тегів <A> …</A> (від англійського anchor- якір).
Тег <A> вживається в двох формах – для переходу на інший документ у його початок, або для переходу до поіменованого фрагмента того ж або іншого документа.

- У першому випадку обов'язковим атрибутом є HREF=url – адреса цільового документа.

Текст і зображення, розміщені між тегами <A> …</A>, стають активною зоною, чуттєвою до щиглика миші, що викликає завантаження цільового документа. Текст гіперпосилання виділяється підкресленням і кольором, зазначеними як значення атрибута LINK тега BODY, або кольором по замовчуванню.

- В другому випадку – при створенні посилання (мітки або закладки) на фрагмент – обов'язковий атрибут NAME=#имя, де ім'я – ім'я ідентифікатора фрагмента (якоря).

Приклад. Нехай у документі report.htm була визначена закладка:
<A NAME="CHEPTER2"> </A>.
Тоді гіперпосилання на цю закладку з іншого документа, що знаходиться в цьому ж каталозі, буде мати такий вигляд:
<A HREF="report.htm#CHEPTER2"> перехід до Глави 2 </A>.

Таблиці.

Таблиці в HTML-документах використовуються не стільки для того, щоб розташовувати дані в обрамлених комірках, скільки з метою позиціонованого розміщення фрагментів тексту і зображень один відносно іншого, створення багатоколонного тексту, обтікання малюнків і т.п.
Основні теги таблиць:

- <TABLE> …< /TABLE > - початок і закінчення таблиці;
- <TR> …</TR> - початок і закінчення рядка;
- <TD> …</TD> - початок і закінчення комірки.

Комірки таблиці можуть містити будь-як дані, допустимі в HTML-документі, в тому числі і вкладені таблиці. Не слід залишати комірки таблиці незаповненими. Якщо за задумом окомірка повинна виглядати порожньою, варто розташувати в ній хоча б нерозривний пробіл - .

Для оформлення таблиць можуть бути використані атрибути, що задаються в тегах <TABLE>, <TR> і <TD>.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

Частина 1. Основні теги оформлення тексту документа і малюнків.

1. На робочому диску створити свою папку, а в ній текстовий файл із назвою **first.htm**

- Запустити редактор Блокнот, ввести в нього текст:
  Вітаю Вас на моїй першій web-сторінці.
- Зберегти файл у створеній папці. При збереженні, у вікні діалогу "зберегти як…"у рядку "Тип файлу:" вибрати варіант "Усі файли (*.*)" , а в рядку "Ім'я файлу" задати ім'я з розширенням **html** або **htm**, наприклад **first.htm**.
- Закрити документ, знайти його піктограму у вікні Провідника.

- Відкрити файл безпосередньо з вікна Провідника. Проаналізувати, за допомогою якого додатка відображається файл і як виглядає введена фраза.
- Зробити висновки про те, що HTML-документ це всього навсього лише текст.

2. Ввести теги, що визначають структуру HTML-документа.

- За допомогою меню Браузера "Вид" - "У вигляді HTML" викликати документ для його редагування. Ввести наведені нижче теги, у розділі заголовка (TITLE) вказати своє прізвище.

```
<HTML>
 <HEAD>
  <TITLE> Прізвище - перший</TITLE>
 </HEAD>
 <BODY>
   Вітаю Вас на моїй першій Web-сторінці.
 </BODY>
</HTML>
```

- Зберегти документ під тим же ім'ям, "Обновити" його відображення в Браузері. Проаналізувати зміни, що відбулися, у відображенні документа.

3. Відредагувати документ.

- Викликати меню Браузера "Вид" - "У вигляді HTML" і додати текст підпису, наприклад,
  Студент групи XX факультету YYY Ім'я Прізвище
  Зберегти документ (не закривати) і обновити його перегляд у Браузері.
- Відредагувати документ так, щоб підпис починався з нового рядка - використовувати тег <BR>. Переглянути в Браузері новий варіант.
- Додати перед підписом горизонтальну лінію.

Увага! Після кожної зміни документ потрібно зберігати, а перегляд у Браузері починати з відновлення завантаження документа за допомогою кнопки "Обновити" на панелі інструментів або натисканням клавіші F5.

4. Виконати оформлення тексту стилем Заголовок.

- Оформити перший рядок документа стилем Заголовок-1 за допомогою парного тега <H1> ...</H1>.
- Оформити рядок з підписом стилем Заголовок-3 і вирівняти по правому краю.
- Змінити стиль оформлення першого рядка на Заголовок-2 з вирівнюванням по центрі, а підпис на Заголовок-4.
- Переглянути отриманий документ у Браузері при різних розмірах шрифту (меню «Вид»).

5. Виконати оформлення абзаців.

- Після заголовка ввести текст монологу Гамлета.

```
To be, or not to be, that is the question:
Whether 'tis nobler in the mind to  suffer
The slings and arrows of outrageous  fortune,
Or to take arms against a sea of  troubles,
And by opposing, end them...
```

Быть иль не быть – вот в чем вопрос.
Что благороднее: сносить удары
Неистовой судьбы - иль против моря
Невзгод вооружиться, в бой вступить
И все покончить разом...

- Оформити кожний з варіантів монологу (5 рядків) як окремий абзац за допомогою тега <P>. Вирівняти англійський текст по лівому краї, а російський – по правому.
- Скопіювати російський варіант монологу і сформувати з нього один абзац: *Быть иль не быть - вот в чем вопрос. Что благороднее: сносить удары неистовой судьбы - иль против моря невзгод вооружиться, в бой вступить. И все покончить разом...*
  Оформити вирівнювання абзацу – «по ширині».

6. Оформлення абзаців за допомогою парного тега <PRE>.

- Скопіювати в кінець документа англійський варіант монологу, помістити його в середину тега PRE і оформити по наведеному зразку "драбинкою":

To be, or not to be, that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles,
And by opposing, end them...

- Зберегти документ **first.htm**, скопіювати його на свій робочий диск на сервері.

7. Виконати шрифтове оформлення документа.

- Створити документ **second.htm** у Вашій папці на жорсткому диску. За основу документа взяти файл **first.htm**, залишити в ньому заголовок, англійський варіант монологу і підпис. Замінити в тезі <TITLE> слово «перший» на «другий».
- Оформити кожен рядок тексту монологу різним кольором.
- 1-ю і 3-ю рядка оформити напівжирним шрифтом, 2-ю і 4-ю – курсивом, 5-ю – підкресленим.

8. Оформлення списків.

- Доповнити текст документа - ввести після заголовка ще три рядки:
  Я знаю як оформляти:
  - Заголовки,
  - Абзаци
- Оформити два останні рядки як нумерований список
- Доповнити список своїх знань. Наприклад, між пунктами "Заголовки" і "Абзаци" додати пункт "Текст". Проаналізувати, як змінилася нумерація елементів списку.
- Створити вкладений список. Додати уточнення видів оформлення шрифтів і абзаців і оформити отриманий список по наступному зразку.

- Оформити:
  1. Шрифти
     - Розмір
     - Колір
     - Гарнітурові
  2. Заголовки
     - Від 1-го до 6-го рівня

3. Абзаци
    - Вирівнювання
    - Розрив рядків усередині абзацу
    - Використовувати преформатування.

9. Вставити в текст малюнок.

- Створити додаткову папку Малюнки. Підготувати в редакторі Paint свій автопортрет, розміром 3Х4см. (60х80 пікселів). Зберегти файл у папці Малюнки чотири рази в різних форматах (різного типу):
    - 24-х розрядний малюнок (*.bmp) - під ім'ям pic_24.bmp,
    - 16-ти колірний малюнок (*.bmp) - під ім'ям pic_16.bmp,
    - Формат GIF (*.gif) - під ім'ям pic_g.gif,
    - Формат JPEG (*.jpg, *.jpeg) - під ім'ям pic_j.jpg.
- Співставити якість і розмір малюнків. Оптимальний варіант розмістити в робочій папці під ім'ям **pic_1.gif** (якщо Ви вибрали формат GIF).
- Вставити малюнок у початок документа - перед заголовком помістити тег:
  <IMG SRC="pic1.gif">
  Переглянути результат у Браузері.
- Вирівняти малюнок по лівому краю (ввести в тег малюнка атрибут вирівнювання). Переглянути результат обтікання в Браузері.
- Доповнити тег малюнка атрибутами задающими розмір малюнка й альтернативний текст "Це мій автопортрет".
- Зробити копію документа **second.htm** і малюнка на диску сервера.

10. Навчитися оформляти фон HTML - документа.

- Створити документ **third.htm**, взявши за основу документ **second.htm.** Зберегти документ **second.htm** під ім'ям **third.htm**. Відредагувати тег <TITLE>, замінивши слово «другий» на «третій». Відредагувати тег <BODY>, ввівши в нього атрибут колірного оформлення тла BGCOLOR=. Задати значення атрибута у форматі RGB (див. табл. основних кольорів), підібрати колір тла у світлих тонах.
- Відредагувати документ **first.htm** так, щоб тло в ньому задавалося фоновим малюнком. Як малюнок можна використовувати файл pic1.gif або будь-який інший кольоровий малюнок невеликого розміру.

11. Продемонструвати створені документи викладачеві. Зберегти копії всіх документів на диску сервера для подальшої роботи.

Частина 2. Робота з таблицями, створення гіперпосилань

1. Створити новий HTML-документ, зберегти його в тій же папці, що і попередні три файли, задати йому ім'я **index.htm**.

2. Розмістити в документі наступні елементи.

- На початку документа помістити заголовок:
  Звіт по лабораторній роботі – Створення HTML-документів.
- Створити таблицю, що складається з двох комірок (без обрамлення). У правій комірці помістити малюнок – Ваш автопортрет, у лівій – свої дані (прізвище, ім'я, навчальну групу). Виконати шрифтове оформлення тексту.
- Далі помістити текст:
  Мною створені три документи!
- Нижче розмістити таблицю, що складається з 3-х комірок по наведеному зразку.

1-й документ	2-й документ	3-й документ

- Оформити таблицю. Таблиця повинна мати рамку, «розфарбувати» комірки у світлі тони.

3. Зв'язати створені документи за допомогою гіперпосилань.

- За текстом у кожній комірці закріпити гіперпосилання на відповідний документ.
- Наприкінці документа ввести фразу: Пишіть мені. Закріпити за нею гіперпосилання на свою адресу e-mail.
- Наприкінці кожного документа помістити малюнок  і закріпити за ним гіперпосилання для повернення до файлу **index.htm.**
- Зберегти всі документи. Перевірити переходи по гіперпосиланнях.

4. Оформити основний заголовок за допомогою рядка, Що Біжить.

5. Продемонструвати створені документи викладачеві.

К О Н Т Р О Л Ь Н І З А П И Т А Н Н Я

1. Що означає поняття HTML-документ?
2. Які переваги цього виду документа?
3. Що означає поняття «тег»?
4. Які різновиди тегов існують?
5. Що входить до складу об'єктів керування тегами?
6. Яка структура HTML-документа?
7. Який вид тега використовується для вставки гіперпосилання?
8. Який вид тега використовується для створення таблиці?
9. Який вид тега використовується для вставки графічного об'єкта?

Лабораторна робота №3

Тема: WEB редактори.
Мета: Ознайомитися з программою Macromedia Dreamweaver. Навчитися створювати основні структурні елементи web-сторінок з допомогою Macromedia Dreamweaver.

ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Інтерфейс середовища Dreamweaver

Потужна професійна середу Dreamweaver володіє всіма необхідними засобами для генерації сторінок HTML будь-якої складності та масштабу. Вона забезпечує режим візуального проектування (WYSIWYG або What You See Is What You Get - "що ти бачиш, то ти і отримаєш"), відрізняється дуже чистою роботою з вихідним текстом веб-документів, володіє вбудованими засобами підтримки великих мережевих проектів. Ні в одному з існуючих у наш час редакторів концепція WYSIWYG не реалізована повністю. [12] Програма Dreamweaver підійшла до декларованому ідеалу ближче конкурентів. Пряма робота з кодами не виключена повністю, але зведена до розумного мінімуму. Програма не тільки володіє потужним арсеналом засобів візуального проектування, але і здатна відображати веб-сторінки майже як спеціалізовані програми перегляду: Microsoft Internet Explorer або Netscape Navigator. [1]
Macromedia Dreamweaver MX - одна із самих потужних програм, що підтримують всі сучасні стандарти Інтернету і неймовірно полегшує виконання навіть найскладніших завдань. Крім того, вона містить у своєму складі розвинену систему підказки та інтерактивних уроків, що дозволяють починаючому користувачеві швидко приступити до роботи. [17]
Інтерфейс програми в порівнянні з попередніми версіями зазнав змін. Новий стиль помітно полегшує роботу з програмою. Якщо в попередніх версіях програми доводилося постійно перемикатися між вікнами, то тепер можна одночасно працювати і з кодом, і з кінцевим видом сайту, що істотно полегшує роботу. [13]
При завантаженні Dreamweaver з'являється стартове вікно (рис.1), що дозволяє вибрати тип нового створюваного документа (HTML, ColdFusion, PHP та ін), або створити документ за готовим зразкам (CSS Style Sheets, Framesets та ін), а також відкрити нещодавно що використовувалися документи. Крім цього є посилання на інтернет-ресурси: сайт Dreamweaver MX Exchange, огляд програми Dreamweaver MX і довідкова інформація по Dreamweaver MX.

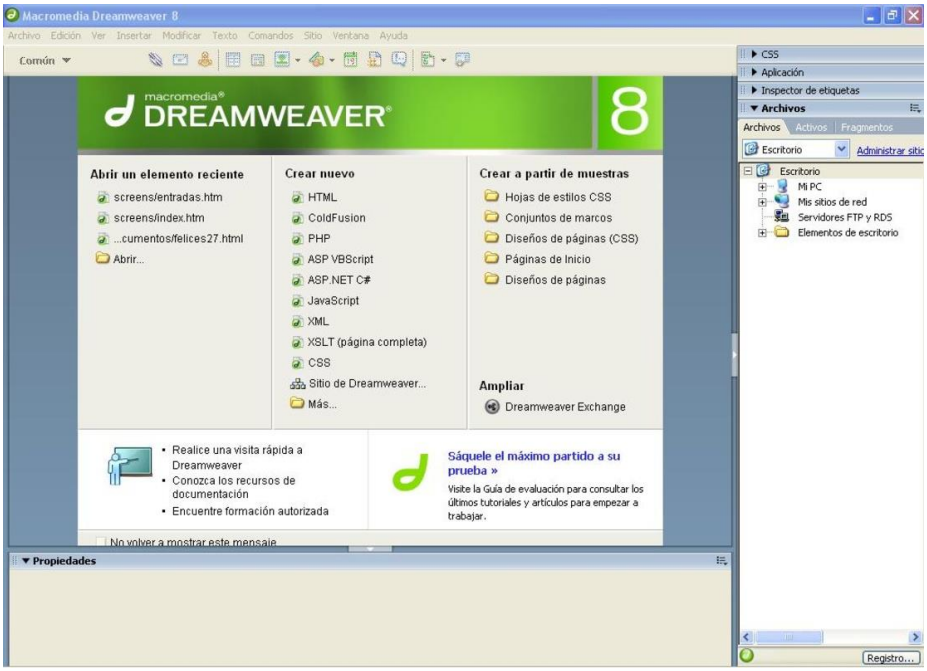


Рис.1. Стартове вікно Dreamweaver 8

Робоча область Dreamweaver містить наступні вікна і панелі:

вікно документа (рис.2), що дозволяє переглядати і редагувати код документа (Code), переглядати зовнішній вигляд майбутнього документа (Design) або одночасно код і зовнішній вигляд (Split). Також є можливість перегляду документа в інтернет-браузері, перевірки помилок і т.д.

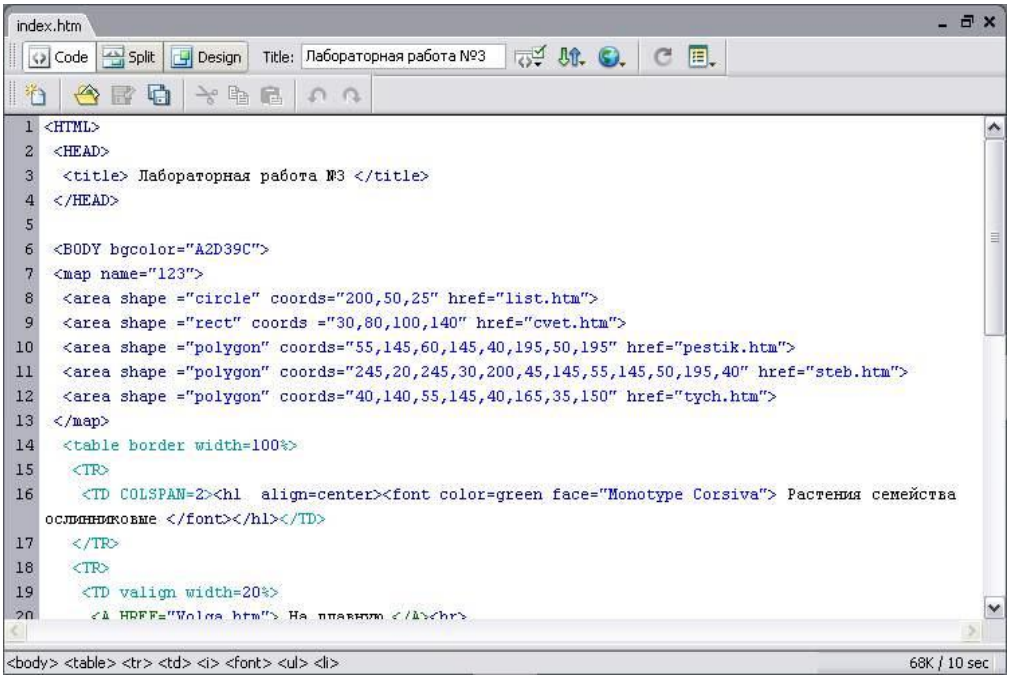


Рис.2. Вікно документа

Стандартне відкриваюче меню (File, Edit, View, Insert, Modify, Text, Commands,Site, Window, Help) (рис.3);

Рис.3. Стандартное відриваюче меню

Панель Properties (Властивості) (рис.4), що дозволяє міняти і додавати властивості виділеного фрагмента коду;

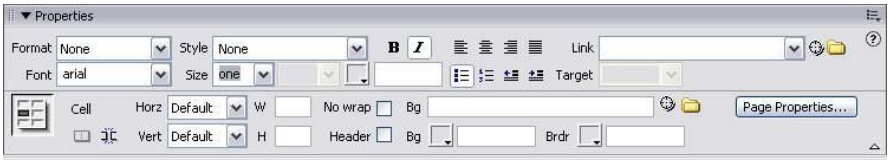


Рис.4. Панель Properties

Панель Insert (Вставка) (мал. 5), що включає наступні вкладки: Common (Основні) - вставка гіперпосилань, таблиць, малюнків, коментарів тощо, а також Tag Chooser (Вибір тега); Layout (Розмітка) - використання осередків і таблиць для розмітки документа перед додаванням вмісту; Forms (Форми) - додавання різних видів форм; Text (Текст) - форматування тексту і Font Tag Editor (Редактор тегів тексту); HTML - додавання горизонтальної лінії, елементів таблиці, фреймів, властивостей head, скриптів; Application (Програми) - робота з інформаційними структурами; Flash elements (елементи Flash) - додавання Flash-роликів, Favorites (Избранное) - можливість додавання на окрему вкладку найбільш часто використовуваних об'єктів.

Рис.5. Панель Insert

Група панелей: Design (містить стилі CSS), Code (містить довідку по тегам, об'єктам і функціям різних технологій), Application (містить інформацію про бази даних, компонентах і т.д.), Files (Диспетчер файлів) (Рис.6) .

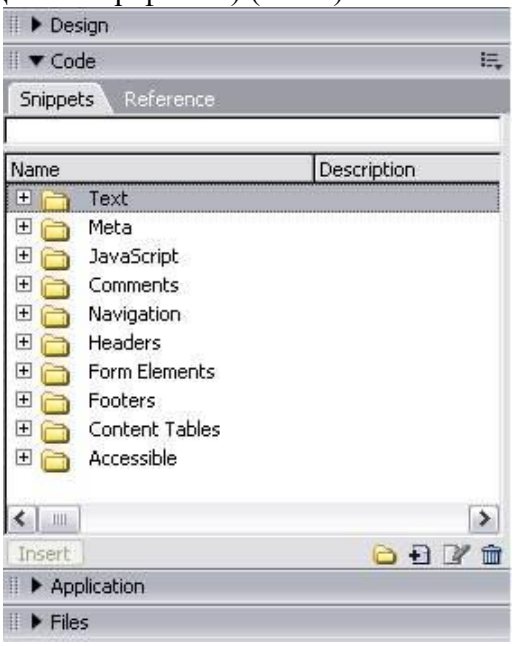


Рис.6. Группа панелей Design, Code, Application, Files

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

1. Створити на своєму робочому диску дерево каталогів з такою структурою:
    - папку **html** — для збереження створюваних Web сторінок,
    - у папці **html** — папку **Pictures** для зберігання використовуваних рисунків.
2. Відкрити вікно програми Dreamweaver.
3. Додати таблицю з двома рядками та двома колонками.
4. Змінити параметри таблиці, встановивши такі параметри: Border=0 CellPadding=0 CellSpacing=0 Width=100% Height=100%**.**
5. Виділити перший рядок і викликати команду Об'єднати комірки.
6. У першому рядку набрати заголовок сторінки (див рис.1.)
7. Для першого рядка встановити параметр висоти 80 пікселів, а для першої комірки другого рядка встановити параметр ширина 180 пікселів. Для першого рядка встановити фонове зображення попередньо створивши його засобами графічного редактора.

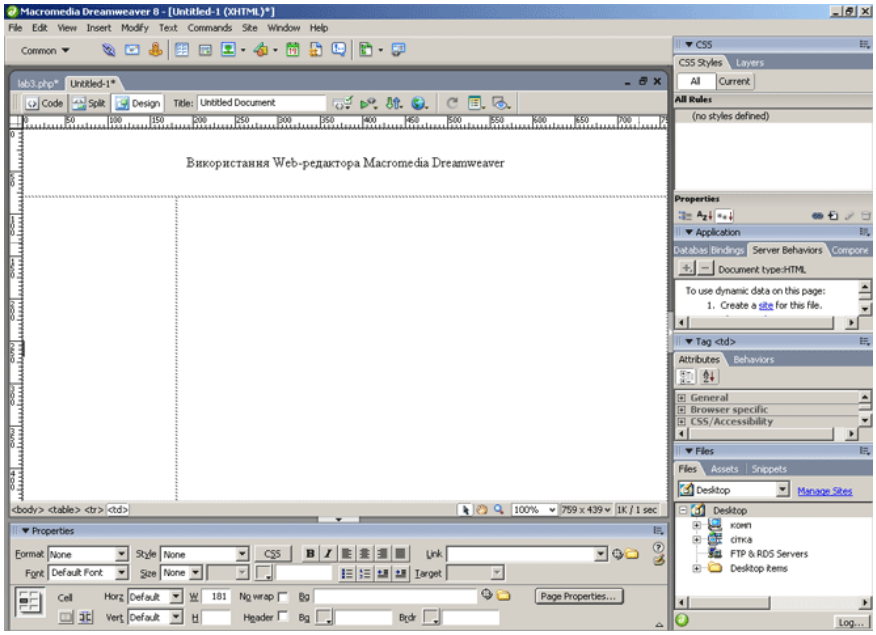


Рис. 1. Шаблон створюваної сторінки

8. У налаштованій комірці з допомогою списка набрати назви пунктів меню і встановити фоновий колір на свій смак. (див. рис. 2.). Для кожного елемента списку створити гіперпосилання на web-сторінку ідентичної структури (сторінки,що відповідають різним розділам відрізнятимуться лише текстовим наповненням).

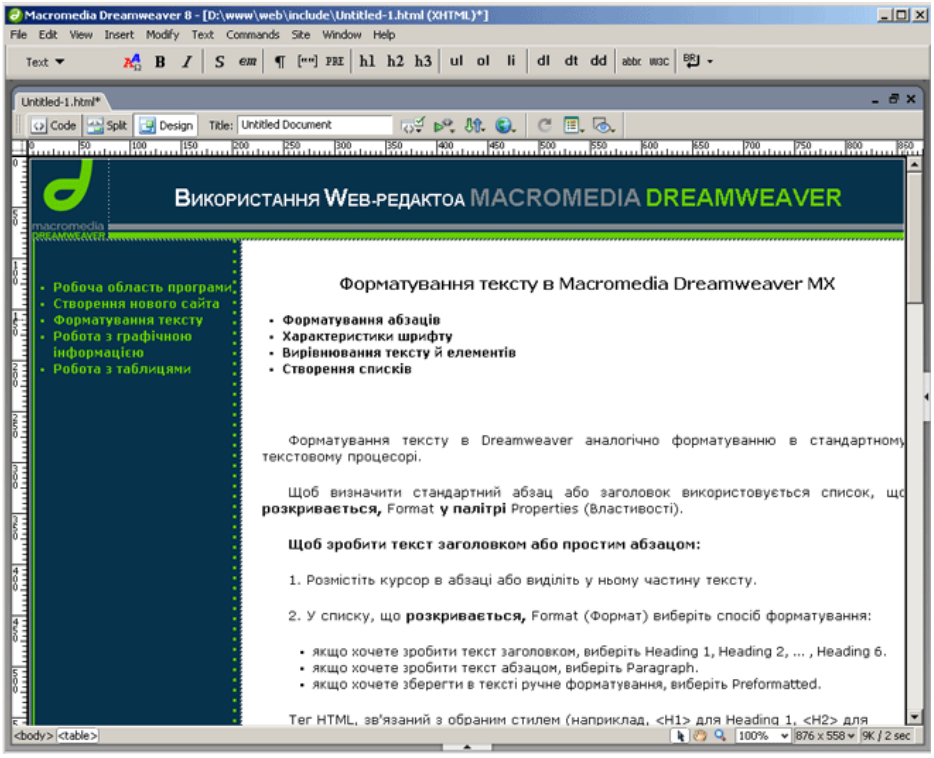


Рис.2. Приклад створюваної HTML сторінки.

9. У секції основного тексту створюємо заголовки підрозділів, що відповідають назвам підрозділів секції, що відповідає назві даного розділу (див. рис. 2).
10. До кожної назви підрозділу додаємо якір (назву внутрішнього гіперпосилання). Якір додається у поточну позицію курсора, назву якоря задаємо будь яким словом символом англійського алфавіту, наприклад A1.
11. Виділяємо перший пункт меню і створюємо гіперпосилання на створений якір (#A1).
12. У секції основного тексту створюємо текстове та художнє наповнення відповідно до вибраних заголовків розділів.
13. У текстовому наповненні обов'язково виділити ключові слова відповідним шрифтом, додати списки, рисунки. Правила оформлення тексту, створення списків, додавання рисунків описано у теоретичних відомостях.
14. Зберегти створену сторінку у створеній папці за допомогою команди **Save as...**
15. Переглянути HTML-код створеної Web сторінки, визначити основні структурні елементи і вказати теги для їх створення. Заповнити таблицю:

Тип	Кількість елементів на сторінці	Послуга створення елементу Web редактором Dreamweaver
заголовок;		
абзац, текст		
таблиця;		
список;		
рисунок;		
посилання		

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Назвати засоби створення HTML сторінок.
2. Порівняти можливості розробки HTML сторінок MS Word та редактора (WEB редактора Dreamweaver).

3. Режими редагування та відображення вихідного HTML тексту у редакторі Dreamweaver.
4. Створення списків засобами редактора Dreamweaver.
5. Створення таблиць засобами редактора Dreamweaver.
6. Форматування тексту (шрифтове оформлення) засобами редактора Dreamweaver.
7. Додавання зображень у HTML сторінки використовуючи редактор Dreamweaver.

Лабораторна робота №4

Тема: Створення HTML форм для введення даних та пересилання їх на сервер.
Мета: Ознайомитися з методами передачі даних на Web-сервер за допомогою HTML форм.

ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Форма – складова HTML-документу, яка містить елементи графічного інтерфейсу користувача (поля введення, списки вибору, кнопки і т.д.) і призначена для відправки введених користувачем даних для опрацювання на Web-сервері.

Опис форм в мові HTML здійснюється за допомогою дескриптора **<form параметри>…</form>**, параметрами якого повинні бути:

1. **action –**сторінка-сценарій, яка виконуватиме обробку переданих даних.
2. **method –** метод передачі даних. Параметр може набувати значень **get** або **post.** При використанні методу **get** дані передаються у форматі: ?ім'я _змінної1=значення1& ім'я _змінної2=значення2&…&ім'я _змінноїn=значенняn. Метод **post** передає дані форми в тілі запиту.
3. **enctype –** формат даних, що передаються. За замовчуванням параметр набудає значення **application/x-www-formurlencoded,** що відповідає передачі даних у текстовому форматі. У випадку завантаження файлів на сервер **enctype** повинен бути

вказаний як **multipart/form-data.**

Елементи графічного інтерфейсу, які використовуються в формах:

Назва	**Опис**	**Параметри**	**Результат**
Тестове поле	<input type="text"…>	**type** – тип елементу (text) **name** –ім'я змінної, для збереження даних вводу; **size** –розмір поля; **maxlength** –максимальна кількість символів вводу; **value** –текст в полі за замовчуванням.	Ваше ім'я
Поле введення паролю	<input type="password"…>	**size** –розмір поля; **maxlength** –максимальна кількість символів вводу;	
Тестове поле (приховане)	<input type="hidden"…>	**type** – тип елементу (hidden) **name** –ім'я змінної, для збереження даних вводу;	
Область тексту	<textarea …> </textarea>	**name** –ім'я змінної, для збереження даних вводу; **rows** – кількість стрічок; **cols** —кількість рядків.	Введіть
Незалежний перемикач (checkbox)	<input type="checkbox"…>	**type** – тип елементу (checkbox); **name** –ім'я змінної, для збереження даних вводу; **value** –значення змінної за замовчуванням; **checked** – вибраний;	
Незалежний перемикач (radio)	<input type="radio">	**type** – тип елементу (checkbox); **name** –ім'я змінної, для збереження даних вводу; **value** –значення змінної за замовчуванням; **checked** – вибраний;	10 100 1000
Список, що розкривається	<select name="ім._зм"> <option value="ім._змN"> </option> </select>	**value** –значення, що буде присвоєне змінній; **multiple** – можливість вибору декількох значень одночасно;	Україна Україна Росія Інша
Кнопка відправки форми	<input type="submit">	**name** –ім'я змінної; **value** –напис на кнопці;	Відправити
Кнопка очищення форми	<input type="reset">	**name** –ім'я змінної; **value** –напис на кнопці;	Очистити

Приклад: створити форму для реєстрації користувача чату, яка містить дані (ім'я, прізвище, адресу електронної пошти, назву користувача (нік), пароль та поле для

повторного вводу паролю).

```
<form action="adduser.php">
  <table align="left">
  <tr>
  <td>
  <div align="right">
  Ім’я <input type="text" name="name"  maxlength="20" size="25"><br>
  Прізвище <input type="text" name="surname"  maxlength="20" size="25"><br>
  Адреса e-mail<input type="text" name="mail"  maxlength="20" size="25"><br>
  Нік <input type="text" name="nik"  maxlength="20" size="25" ><br>
  Пароль<input type="password" name="password"  maxlength="20" size="25"><br>
  Пароль ще  раз<input type="password" name="confirm_pass" maxlength="20" size="25">
  <br><input type="submit"  value="Зареєструватися">
  </div>
  </td>
  </tr>
  </table>
</form>
```

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

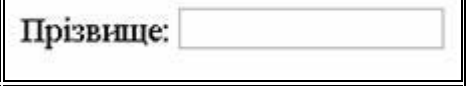
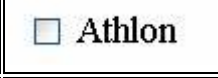
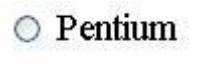
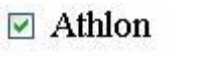
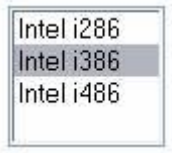
1. Розробити сторінку для формування замовлення віртуального книжкового магазину. Для пересилання введених даних використати метод POST. Передбачити введення таких даних:
   - Назва
   - Автор
   - Ціна
   - Рік видання
   - Кольрова чи чорно-біла (з допомогою прапорців)
   - Видавництво
2. Розробити форму для введення анкетних даних студента:
   - Прізвище; Ім'я;
   - По батькові;
   - Факультет;
   - Спеціальність;
   - Рік вступу на навчання;
   - Рік завершення навчання;

   Для пересилання введених даних використати метод GET

3. Доповнити розроблену анкету пунктом стать (надати можливість вибору з списку).

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Пояснити різницю між методами пересилання даних серверу GET та POST
2. Встановити відповідність між наведеним фрагментом коду і зображеною формою:

1	<INPUT TYPE="checkbox" NAME="type" VALUE="Athlon" CHECKED>	**a**	
2	Прізвище: <INPUT NAME= "street">	**b**	
3	<INPUT TYPE="checkbox" NAME="type" VALUE="Athlon"> Athlon	**c**	
4	<SELECT MULTIPLE NAME=type> <OPTION VALUE="Intel i286"> Intel i286 <OPTION SELECTED VALUE="Intel i386"> Intel i386 <OPTION VALUE="Intel i486"> Intel i486 </SELECT>	**d**	
5	<INPUT TYPE="radio" NAME="type" VALUE="Pentium"> Pentium	**e**	
6	<INPUT TYPE="radio" NAME="type" VALUE="Pentium" CHECKED> Pentium	**f**	
7	<SELECT NAME=type> <OPTION SELECTED VALUE="Intel i286"> Intel i286 <OPTION VALUE="Intel i386"> Intel i386 <OPTION VALUE="Intel i486"> Intel i486 </SELECT>	**g**	
8	<TEXTAREA NAME="adress" COLS=20 ROWS=6> </TEXTAREA>	**h**	

Лабораторна робота №5

Тема: Каскадні таблиці стилів CSS.
Мета: Освоїти основні методи використання каскадних таблиць стилів CSS. Навчитися вказувати властивості для елементів гіпертекстової розмітки HTML-документа, створювати свої класи стилів.

ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

DHTML-термін, який використовується для позначення сторінок, вміст яких динамічно змінюється.

DHTML являє собою сукупність HTML, каскадних таблиць стилів CSS (Cascade Style Sheets) та мови написання сценаріїв (JavaScript або VBScript). Ці компоненти пов'язані між собою об'єктною моделлю документа (DOM), яка є інтерфейсом прикладного програмування. Каскадні таблиці стилів являють собою технологію визначення і приєднання стилів до HTML-документів. Каскадні таблиці стилів можна порівняти зі

стилями текстового редактора. Таблиця стилів - це шаблон, який управляє форматуванням тегів HTML у Web-документі.

Таблиця стилів може міститись у самому документі або зберігатись у окремому текстовому файлі із розширенням **css**. Правило каскадних таблиць складається з двох частин: селектора і визначення. Селектором може бути будь-який тег HTML, для якого визначення задає, яким чином необхідно тег форматувати. Саме визначення теж складається з двох частин: властивості і її значення, розділених знаком двокрапка (:).

```
h1 {color:blue; font-size: 16pt}
```

У прикладі для заголовку першого рівня визначені такі властивості: колір тексту - синій, розмір тексту - 16 пунктів.

Усі розміри можна задавати у таких одиницях:

- **px** - pixels (пікселі),
- **in** - inches (дюйми),
- **ex** - x-height (висота літери "x" шрифта, що використовується),
- **cm** - centimeters (сантиметри),
- **mm** - milimeters (міліметри),
- **pt** - points (пункти; lpt=l/72 in),
- **pc** - picas (піки; lpc= 12pt).

Існує чотири способи зв'язування документу і таблиці стилів:

- **Зв'язування** - дозволяє використовувати одну таблицю стилів для форматування багатьох сторінок;
- **Впровадження** - дозволяє задавати всі правила таблиці стилів у самому документі:
- **Імпортування** - дозволяє вбудовувати у документ таблицю стилів, розміщену на сервері;
- **Вбудовування** у теги документа - дозволяє змінити форматування конкретних елементів сторінки.

Спосіб 1. Зв'язування дозволяє зберігати таблицю стилів у окремому файлі і приєднувати її до документів за допомогою тега <link> у розділі <head>:

```
<link  rel="stylesheet" type="text/css" href="mystyles.css">
```

Параметр href тега <link> вказує на місце знаходження файлу, що містить таблицю стилів.

Спосіб 2. Впровадження: використовується тег <style>, параметр якого type повинен дорівнювати "text/ess". Таблиця стилів обмежується коментарем. (<!-- -->). Це робиться для того, щоб браузери, що не вміють працювати з і стилями, ігнорували їх.

```
<head>
  <style  type="text/css">
     b  {text-transform: uppercase;}
     p  {background-color: grey; text-align: center}
  </style>
</head>
```

У прикладі визначено таблицю стилів для тегів , : текст, що міститься у контейнері   буде записано у верхньому регістрі; текст, що міститься у контейнері   буде відображуватись сірим кольором та центруватися.

Спосіб 3. Імпортування: за допомогою тега <style> можна імпортувати зовнішню таблицю стилів, визначивши значення властивості @import, яке повинно містити URL таблиці стилів: @import: url (mystyle.css)

Спосіб 4. Вбудовування у тег: використовується для визначення стилю об'єкта, що форматується певним тегом, за допомогою параметру style.

<h1 style="color: red">Загловок відображується червоним кольором </h1>

Для зменшення розміру CSS використовується групування – різні селектори записуються через кому.

Приклад. hl,h2,h3: {font-family: Arial};

Вкладені елементи наслідують правила форматування батьківського елемента.

Селектор class дозволяє задати різні правила форматування для елемента визначенного типу. Наприклад:

```
<style  type="text/css">
        h1.red  {color: red; }
        h1.bluebgrd  {color: red; background-color: blue}
</style>
```

У тексті документа посилання на відповідний клас задається у параметрі class:

```
<h1 class="red"> Червоний шрифт</h1>
<h1 class="bluebgrd>Червоний шрифт на синьому фоні</h1>
Якщо клас повинен використовуватись  для всіх елементів документа,
то конкретний тег не вказується:
  <style  type="text/css">
       .red  {color red;}
       .bluebgrd  {color: red; background-color: blue}
  </style>
```

Тепер класи red і bluebgrd можна застосовувати до будь-яких елементів документу, що мають властивості, визначені у стилі.

Параметр id задає унікальне ім'я елемента, яке використовується для посилань на нього у сценаріях і таблицях стилів. Для цього перед ім'ям записується символ #.

```
<html>
  <head>
    <title> Приклад використання параметру id </title>
    <style  type="text/css">
      #red  {color: red;}
      #bluebgrd  {color: red; background-color: blue}
    </sryle>
  </head>
  <body>
```

```
    <p  id=red>12345</p> <h1 id=bluebgrd> 123456 </h1>
  </body>
</html>
```

За допомогою контейнера можна перевизначити стиль фрагмента тексту всередині іншого стандартного контейнера

```
<html>
  <head>
    <title> Приклад використання контейнера span </title>
    <style  type="text/css">
      p  { color:red; font-size: 24pt; font-family: times;}
       span  {color:yellow; font-size: 18pt; font-family: courier}
    </style>
  </head>
  <body>
    <p>  Текст червоного кольору
    <span>Текст жовтого  кольору</span>  Знову текст червоного кольору
    </p>
  </body>
</html>
```

До одного документа можна приєднати декілька таблиць стилів, які утворюють "каскад" (тому таблиці стилів називають каскадними). Таблиці стилів мають такий пріоритет (від найменшого до найбільшого): зв'язана таблиця стилів, імпортована таблиця стилів, правило з елементом HTML у якості селектора, правило з параметром CLASS у якості селектора, правило з параметром ID у якості селектора, вбудоване у тег HTML правило.

При роботі з гіперпосиланнями використовують псевдокласи тега <a>:

```
<style  type="text/css">
       a:link  {colonred} /*невідвіданий зв'язок*/
       a:visited  {colonblue} /*відвіданий зв'язок*/
       a:active  {colongreen} /*активний зв'язок*/
      a:hover  {colorbeige} /*зв'язок, на якому розміщено вказівник миші*/
</style>
```

За допомогою псевдокласу hover можна змінити колір гіперпосилання, на якому розміщено вказівник миші.

Деякі властивості, що використовуються у таблицях стилів:

ВЛАСТИВІСТЬ	**ПРИЗНАЧЕННЯ**	**ПРИКЛАДИ**
font-family	задає пріоритетний список шрифтів	body {font-family: "Times New Roman", serif}
font-style	стиль шрифта (normal-нормальний, italic-курсив, oblique-нахилений)	h1 {font-style: italic}
font-weight	використовується для визначення товщини шрифта (normal-нормальний, bold-напівтовстий). Використовуються і числові значення з кроком 100, починаючи від 100 і закінчуючи 900.	h1 { font-weight: bold}
font-size	розмір шрифта	p {font-size :10pt}

font	одночасне встановлення властивостей шрифта	p {font: italic 12pt serif]
color	колір тексту	p {color: blue}
background-color	колір фону	body { background-color: blue}
background-image	фонове зображення	body {background-image: url(/image/image 1 .gif)}
text-decoration	тип підкреслювання (overline- верхнє підкреслення, underline- нижнє підкреслення, line-through— закреслений текст.)	hi {text-decoration: underline}
text-align	вирівнювання тексту {left, right, center)	h2 {text-align: center}
vertical-align	вирівнюваня тексту по вертикалі, (top -згори, middle - по середині, bottom - знизу)	h2 { vertical-align: middle}
text-transform	зображення усіх символів тексту у верхньому регістрі (uppercase) або у HHHCHbOMy(lowercase) none-відміна використання параметра	h2 { text-transform: uppercase }
margin	значення полів (верхнє,праве,нижне, ліве).	body { margin: 10 10 20 20}
padding	відступи від елемента	img { padding:5 5 5 7}
border-color	колір границі	img {border-colonblue}
visibility	визначає, чи є елемент видимий у даний момент (visible - видимий, hidden - прихований)	img {visibility: hidden}
list-style-type	тип маркера списку (для нумерованого списку можливі значення: lower-roman - маленькі римські цифри, upper-roman - великі римські цифри, lower-alpha маленькі латинські літери, upper-alpha -великі латинські літери. Для списку з маркерами можливі значення: disc, circle, square.)	ul {list-style-type: disk}
list-style- image	URL-зображення, яке буде використовуватися, як зображення маркер	ol {list-style-image: url (list gif)}
list-style- position	Позиція маркера (inside-у абзаці, outside-вліво від абзацу)	ol { list-style-image: url (list.gif); list-style-position: inside }

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

Для створеного сайту "Використання WEB-редактора Macromedia Dreamweaver" (в лабораторній роботі №3):

1. Створити зв'язану **CSS,** яка б містила значення таких властивостей:
   - колір фону, колір символів;
   - пріоритетність використання шрифтів;
   - розмір та колір заголовків 1 -го рівня;
   - відображення курсивом заголовків 2-го рівня;
   - вид маркерів у списках;
   - відображення певним кольором гіперпосилань та зміну кольору гіперпосилання при наведенні на нього вказівника миші.

- для абзацу задати такі влативості: міжстрічковий інтервал, абзацний відступ, кегль, розмір шрифту.

2. Створити впроваджену **CSS** на кожній зі сторінок, яка б містила значення таких властивостей:
   - значення полів;
   - відступи від зображень;
   - шрифт в заголовках таблиць.

   Передбачити у цій таблиці стилів для значень полів використання селектору class.

3. Використати на всіх сторінках сайту вбудовані у теги **<h2>**, **<b>**, **<p>** стилі.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що називають таблицею стилів?
2. Чому таблиці стилів називають каскадними?
3. Як записується правило CSS?
4. Назвіть спопоби зв'язування CSS із документом.
5. За яким пріоритетом використовуються CSS у документі?
6. Яке призначення селектору class?
7. Наведіть приклади використання параметру id.
8. У яких випадках потрібно використовувати контейнер ?
9. Які ви знаєте псевдокласи тега <a> і для чого вони використовуються?
10. Які властивості у CSS використовуються для визначення вигляду шрифта?
11. Які властивості у CSS використовуються для роботи з кольором?
12. Які властивості у CSS використовуються для встановлення значень полів?
13. Назвіть властивості, що використовуються у CSS для форматування списків.
14. Назвіть властивості, що використовуються у CSS для форматування границь.

Лабораторна робота №6

Тема: Розробка простих PHP сценаріїв.
Мета: Ознайомитися з базовими алгоритмічними структурами мови PHP.

ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Сценарії PHP зберігаються у файлі з розширенням php. Самі ж фрагменти коду на мові PHP відокремлюються конструкціями <? ?>. Іншими складовими php-файлу можуть бути фрагменти HTML-коду. Стрічки що починаються символами "//" вважаються коментаріями. Багато стрічкові коментарії формуються за допомогою конструкції /* */. У кінці кожного оператора повинен бути символ ";".
Імена змінних завжди розпочинаються із символу "$", самі ж змінні не потребуються попереднього опису. У PHP підтримуються шість основних типів даних.
Цілі числа. Крім чисел у 10-ковій системі числення у PHP реалізовано підтримку 8-кових та 16-кових чисел. Числа у 8-ковій системі числення записуються із символом "0" попереду (04528), у 16-ковій – "0x" або "0X" (0x9A5F). Над змінними цілочисельного типу

виконуються арифметичні операції додавання ($a+$b), віднімання ($a-$b), множення ($a*$b), ділення ($a/$b), отримання остачі від ділення ($a%$b), інкременту ($a++), декременту($a--), порівняння: рівність ($a= =$b), нерівність ($a!=$b), більше ($a>$b), менше ($a<$b).

Дійсні числа записуються у форматі з плаваючою крапкою (16.53) та з фіксованою крапкою 16.736e5.

Рядки у мові PHP – це послідовність символів, записаних в лапках “” або ‘’, причому при опрацюванні рядків у межах подвійних лапок (“”) замість змінних підставляються їх значення. Над змінними рядкового типу виконуються операції визначення символу у вказаній позиції ($s[4] –поверне 5 символ стрічки) та об’єднання (конкатенації ($s1.$s2)).

Масиви – є списком однотипних даних. У PHP реалізовано 2 типи масивів. У масивах першого типу положення елемента визначається його індексом. Масиви 2 типу мають асоціативні зв’язки.

Об'єктом є змінна, яка створюється за взірцем (класом). Тобто об’єкт є екземпляром класу, а тому повинен обов’язково бути описаним.

Логічні величини набувають лише два значення істина (true) або хибність (false). Над змінними логічного типу виконуються операції заперечення (!$b), логічного множення (($a && $b) та додавання ($a || $b).

Для виводу стрічки у поле браузера використовується оператором **echo**, після якого вказати змінну або стрічку.

Операція присвоєння здійснюється за допомогою символу дорівнює (=). Іншими базовими конструкціями мови є оператори умови (if, switch) та організації циклів while, for.

Синтаксис використання оператора if:

if (умова) { послідовність операторів; }else { послідовність операторів; }.

Інколи замість декількох послідовних конструкцій if-else доцільно використати switch-case

```
switch (умова) {
  case значення1: команди1; break;
  case значення2: команди2; break;
  . . .
  case значенняN: командиN; break;
  default: команди_за_замовчуванню; break
}
```

Синтаксис циклу з передумовою:

```
while (умова виконання циклу) {
 послідовність команд;
 }
```

Синтаксис циклу з післяумовою:

```
do {
 послідовність команд;
 while (умова виконання циклу) {
 }
```

Синтаксис циклу з наперед заданим числом ітерацій:

```
for (початкове_значення_змінної_лічильника;
кінцеве_значення_змінної_лічильника;
        операція_зміни_лічильника){
 послідовність команд;
 }
```

Завдання для виконання

1. Розробити сценарій для виведення привітання користувачу, який вводить своє ім’я за допомогою форми. Для передачі даних використати метод GET.
2. Розробити сценарій обчислення суми, різниці, добутку, частки й остачі для введених двох чисел. Перевірити його виконання. Для передачі даних використати метод POST.

3. Розробити сценарій для форматування введеного тексту (передбачити визначення таких властивостей тексту: кольору, гарнітури, розміру, кеглю).
4. Розробити сценарій для обчислення факторіала введеного числа (виконати трьома способами, використовуючи три типи циклів).
5. Розробити сценарій для обчислення кількості нажимань на кожну з двох кнопок протягом одного сеансу.
6. Розробити сценарій для проведення тестування (4-5 питань) та виведення результатів анкетування користувачу.

   Приклад анкети:

   1. Вкажіть пристрої, які необхідні для існування обчислювальної системи:
      - пристрої введення-виведення;
      - клавіатура;
      - процесор;
      - оперативний запам'ятовуючий пристрій;
      - дисплей;
      - принтер.
   2. Вкажіть пристрої, які входять до системного блоку персонального комп'ютера:
      - центральний мікропроцесор;
      - клавіатура;
      - дисплей;
      - оперативний запам'ятовуючий пристрій;
      - постійний запам'ятовуючий пристрій;
      - контролери (адаптери) пристроїв введення-виведення;
      - системна шина;
      - накопичувачі для гнучких, жорстких, лазерних дисків;
      - миша.
   3. Вкажіть стандартні пристрої введення-виведення;
      - дисплей;
      - накопичувачі на магнітних дисках;
      - накопичувачі на лазерних дисках;
      - клавіатура;
      - миша;
      - сканер;
      - принтер.
   4. Вкажіть нестандартні пристрої введення-виведення;
      - дисплей;
      - накопичувачі на магнітних дисках;
      - накопичувачі на лазерних дисках;
      - клавіатура;
      - миша;
      - сканер;
      - принтер.
   5. Вкажіть правильні відповідності:
      - 1 Гбайт = 210 Мбайт;
      - 1 Мбайт = 210 байт;
      - 1Кбайт = 2Е+10 байт;
      - 1 Мбайт = 1024 Кбайт.
   6. Вставити пропущене слово: Поіменовану сукупність елементів інформації, що зберігається на носіях інформації, називають:____
7. Використовуючи конструкцію:

```
<?
$file=empty($_GET['page'])?null:$_GET['page'];
if(empty($file))
```

```
        include("include/head.php");
elseif(file_exists("include/".dirname($file)."/".basename($file)))
        include("include/".dirname($file)."/".basename($file));
else
        include("include/error.php");
            ?>
```

переробити створений у лабораторих роботах №3,5 сайт "Використання WEB-редактора MACROMEDIA DREAMWEAVER" так, щоб браузер завжди відображав один і той самий файл, в який інклудиться наповнення з інших файлів в залежності від обраного пункту меню.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Чи можна виконати php-скрипт у файлі з розширенням html?
2. Чи можна використовувати дескриптори мови HTML у php-файлі?
3. На Вашу думку транслятор мови PHP є інтерпретатором чи компілятором?
4. З'ясуйте де виконується php-код, а де код html?
5. Назвіть базові алгоритмічні структури мови php?
6. Перелічіть базові типи даних мови php? Чи є обов'язковим попередній опис змінних?

Лабораторна робота №7

Тема: Розробка PHP сценаріїв з використанням та опрацюванням масивів.
Мета: Ознайомитися з типом даних в PHP масивами і стандартними функціями роботи з масивами.

ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Тип array (масив)

Масив у PHP являє собою впорядковану карту – тип, що перетворює значення в ключі. Цей тип оптимізовано у декількох напрямках, тому ви можете використовувати його як

масив, список (вектор), хеш-таблицю (що є реалізацією карти), стек, черга і т.д. Оскільки ви можете мати як значення інший масив PHP, можна також легко емулювати дерева.

Задати масив можна за допомогою конструкції array() або безпосередньо задаючи значення його елементам.

Визначення за допомогою array()

```
array ([key] => value,[key1] => value1, ... )
```

Мовна конструкція array() приймає як параметри пари **ключ** => значення, розділені комами. Символ => встановлює відповідність між значенням і його ключем. Ключ може бути як цілим числом, так і рядком, а значення може бути будь-якого ніснуючого в PHP типу. Числовий ключ масиву часто називають індексом. Індексування масиву в PHP починається з нуля. Значення елемента масиву можна одержати, вказавши після імені масиву в квадратних дужках ключ шуканого елемента. Якщо ключ масиву являє собою стандартний запис цілого числа, то він розглядається як число, в противному випадку – як рядок. Тому запис $a["1"] рівносильний запису $a[1], так само як і $a["-1"] рівносильне $a[-1].

```
<?php
$books = array ("php" =>"PHP users guide",12 => true);
echo $books["php"];
//виведе "PHP users guide"
echo $books[12];     //виведе 1
?>
```

Якщо для елемента ключ не заданий, то в якості ключа береться максимальний числовий ключ, збільшений на одиницю. Якщо вказати ключ, якому вже було присвоєно якесь значення, то воно буде перезаписано. Починаючи з PHP 4.3.0, якщо максимальний ключ – від'ємне число, то наступним ключем масиву буде нуль (0).

```
<?php
// масиви $arr і $arr1 еквіваленти
$arr = array(5 => 43, 32, 56, "b" => 12);
$arr1 = array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

Якщо використовувати як ключ TRUE або FALSE, то його значення переводиться відповідно в одиницю і нуль типу integer. Якщо використовувати NULL, то замість ключа одержимо порожню стрічку. Можна використовувати і сам порожню стрічку як ключ, при цьому її треба брати в лапки. Так що це не те ж саме, що використання порожніх квадратних дужок. Не можна використовувати як ключ масиви й об'єкти.

Визначення за допомогою синтаксису квадратних дужок

Створити масив можна, просто записуючи в нього значення. Як ми вже говорили, значення елемента масиву можна одержати за допомогою квадратних дужок, всередині яких потрібно вказати його ключ наприклад, $book["php"]. Якщо вказати новий ключ і нове значення наприклад, $book["new_key"]="new_value", то в масив додасться новий елемент. Якщо ми не вкажемо ключ, а тільки привласнимо значення $book[]="new_value", то новий елемент масиву буде мати числовий ключ, на одиницю більший максимального існуючого. Якщо масив, у який ми додаємо значення, ще не існує, то він буде створений.

```
<?
$books["key"]= value; // додали в масив
```

```
                        // $books значення
                        // value із ключем key
$books[] = value1; /* додали в масив
                        значення value1 з
                        ключем 13, оскільки
                        максимальний ключ у
                        ми був 12 */
?>
```

Для того щоб змінити конкретний елемент масиву, потрібно просто присвоїти йому з його ключем нове значення. Змінити ключ елемента не можна, можна тільки знищити елемент (пару ключ/значення) і додати нову. Щоб видалити елемент масиву, потрібно використовувати функцію unset().

```
<?php
$books = array ("php" =>"PHP users guide",12 => true);
$books[] ="Book about Perl"; // додали елемент
                    // з ключем (індексом)
                    // 13 це еквівалентно
                    // $books[13] =
                   // "Book about Perl";
$books["lisp"] =123456; /* Це додає до масиву новий
            елемент із ключем "lisp" і
            значенням 123456 */
unset($books[12]); // Це видаляє елемент
                   // с ключем 12 з масиву
unset ($books); // видаляє масив цілком
?>
```

Помітимо, що, коли використовуються порожні квадратні дужки, максимальний числовий ключ шукається серед ключів, що існують у масиві з моменту останнього переіндексування. Переіндексувати масив можна за допомогою функції array_values().

```
<?php
$arr =array ("a","b","c"); /* Створюємо масив зі значеннями "a", "b"
і "c".
                                    Оскільки ключі не зазначені, вони
будуть
                                 0,1,2 відповідно */
print_r($arr); // виводимо масив (і ключі, і значення)
unset($arr[0]);
unset($arr[1]);
unset($arr[2]);
         // знищуємо з нього всі значення
print_r($arr);  // виводимо масив (і ключі, і значення)
$arr[] = "aa";  // додаємо новий елемент
                // у масив.
                // Його індексом (ключем)
                // буде 3, а не 0
print_r($arr);
$arr =array_values($arr); // переіндексуємо масив
$arr[] = "bb"; // ключем цього елемента буде 1
print_r($arr);
?>
```

Результатом роботи цього скрипта буде:

```
Array ( [0] => a [1] => b [2] => c )
Array ( )
Array ( [3] => aa )
Array ( [0] => aa [1] => bb )
```

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

1. Розробити форму введення масиву з наперед заданою кількістю елементів.

2. Користуючись розробленою формою в завданні 1, ввести масив елементів з цілих чисел і виконати над ним такі дії:
   - вивести ключ (індекс) першого нульового елементу масиву
   - вивести кількість від'ємних елементів масиву;
   - впорядкувати за спаданням елементів (від найбільшого до найменшого);
   - впорядкувати за зростанням елементів (від найменшого до найбільшого);
   - вивести максимальний елемент масиву;
3. Розробити скрипт, який формує таку web-сторінку.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Які типи даних існують в php.
2. Яким чином оголосити тип arrey.
3. Як змінити значення елементу масиву.
4. Як можна видалити елемент масиву.

Лабораторна робота №8

Тема: PHP функції, визначені користувачем.
Мета: Навчится задавати власні PHP функції та використовувати їх в програмних кодах.

ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

1.Функції, задані користувачем.
Для чого потрібні функції? Щоб відповісти на це питання, потрібно зрозуміти, що взагалі являють собою функції. У програмуванні, як і в математиці, функція є відображенням безлічі її аргументів на безліч її значень. Тобто функція для кожного набору значень аргументу повертає якісь значення, що є результатом її роботи. Навіщо потрібні функції, спробуємо пояснити на прикладі. Класичний приклад функції в програмуванні – це функція, що обчислює значення факторіала числа. Тобто ми задаємо їй число, а вона повертає нам його факторіал. При цьому не потрібно для кожного числа, факторіал якого

ми хочемо одержати, повторювати той самий код – досить просто викликати функцію з аргументом, рівним цьому числу.

Функція обчислення факторіала натурального числа

```
<?php
function fact($n){
  if ($n==0) return 1;
  else return $fact = $n * fact($n-1);
}
?>
```

Таким чином, коли ми здійснюємо дії, в яких простежується залежність від якихось даних, і при цьому, можливо, нам знадобиться виконувати такі ж дії, але з іншими вихідними даними, зручно використовувати механізм функцій – оформити блок дій у виді тіла функції, а змінні дані – у якості її параметрів.

Подивимося, як у загальному виглядає задання функції. Функція може бути визначена за допомогою наступного синтаксису:

```
function Ім'я_функції (параметр1, параметр2,... параметр){
   Блок_дій
   return "значення, що повертаються функцією";
}
```

де *Ім'я_функції* й імена параметрів функції (параметр1, параметр2 і т.д.) повинні відповідати правилам найменування в PHP. Імена функцій нечуттєві до регістра. *Параметри функції* – це змінні, тому перед назвою кожної з них повинен стояти знак $. *Блок_дій* у тілі функції - будь-який правильний PHP-код (не обов'язково залежний від параметрів). І нарешті, після ключового слова *return* повинен йти коректний php-вираз (що-небудь, що має значення). Крім того, у функції може і не бути параметрів, як і значення, що повертається. Приклад правильного оголошення функції – функція обчислення факторіала, наведена вище.

Як відбувається виклик функції? Вказується ім'я функції й у круглих дужках список значень її параметрів, якщо такі існують:

```
<?php
Ім'я_функції                                    ("значення_для_параметра1",
"значення_для_параметра2",...);
// приклад виклику функції – виклик функції
// обчислення факторіала приведений вище,
// там для обчислення факторіала числа 3
// ми писали: fact(3);
// де fact – ім'я викликуваної функції,
// а 3 – значення її параметра з ім'ям $n
?>
```

Коли можна викликати функцію? Здавалося б, дивне питання. Функцію можна викликати після її визначення, тобто в будь-якому рядку програми нижче блоку function f_name(){...}. У PHP3 це було дійсно так. Але вже в PHP4 такої вимоги немає. Уся справа в тому, як інтерпретатор обробляє одержуваний код. Єдине виключення складають функції, обумовлені умовно (всередині умовних операторів або інших функцій). Коли функція визначається таким чином, її визначення повинне передувати її викликові.

```
<?
$make = true;
/* тут не можна викликати Make_event();
```

```
тому що вона ще не існує, але можна
викликати Save_info() */

Save_info("Вася","Іванов", "Я вибрав курс по PHP");
if ($make){
// визначення функції Make_event()
  function Make_event(){
    echo "<p>Хочу вивчати Python<br>";
  }
}
// тепер можна викликати Make_event()
Make_event();
// визначення функції Save_info
function Save_info($first, $last, $message){
  echo "<br>$message<br>";
  echo "Ім'я: ". $first . " ". $last . "<br>";
}
Save_info("Федя","Федоров", "А я вибрав Lisp");
// Save_info можна викликати і тут
?>
```

Якщо функція один раз визначена в програмі, то перевизначити або видалити її пізніше не можна. Незважаючи на те, що імена функцій нечуттєві до регістра, краще викликати функцію по тому ж імені, яким вона була задана у визначенні.

Розглянемо докладніше аргументи функцій, їхнє призначення і використання.

2. Аргументи функцій.

У кожної функції може бути, як ми вже говорили, список аргументів. За допомогою цих аргументів у функцію передається різна інформація (наприклад, значення числа, факторіал якого треба підрахувати). Кожен аргумент являє собою змінну або константу.

За допомогою аргументів дані у функцію можна передавати трьома різними способами. Це передача аргументів за значенням (використовується за замовчуванням), по посиланню і задання значення аргументів за замовчуванням. Розглянемо ці способи докладніше.

Коли аргумент передається у функцію за значенням, зміна значення аргументу всередині функції не впливає на його значення поза функцією. Щоб дозволити функції змінювати її аргументи, їх потрібно передавати по посиланню. Для цього у визначенні функції перед ім'ям аргументу варто написати знак амперсанд «&».

```
<?php
// напишемо функцію, яка б додавала до рядка слово checked
function add_label(&$data_str){
  $data_str .= "checked";
}
$str = "<input type=radio name=article ";
  // нехай є такий рядок
echo $str ."><br>";
    // виведе елемент форми – не відзначену радіо кнопку
add_label($str);
    // викличемо функцію
echo $str ."><br>";
    // це виведе уже відзначену радіо кнопку
?>
```

У функції можна визначати значення аргументів, використовувані за замовчуванням. Саме значення за замовчуванням повинне бути сталим виразом, а не змінною і не представником класу або викликом іншої функції.

У нас є функція, що створює інформаційне повідомлення, підпис до якого змінюється в залежності від значення переданого їй параметра. Якщо значення параметра не задано, то використовується підпис "Оргкомітет".

```
<?php
function Message($sign="Оргкомітет"){
// тут параметр sign визначається за замовчуванням
// значення "Оргкомітет"
  echo "Наступні збори відбудуться завтра.";
  echo "$sign .<br>";
}
Message();
    // викликаємо функцію без параметра.
    // У цьому випадку підпис – це Оргкомітет
Message("З повагою, Вася");
    // У цьому випадку підпис
    // буде "З повагою, Вася"
?>
```

Якщо у функції кілька параметрів, то ті аргументи, для яких задаються значення за замовчуванням, повинні бути записані після всіх інших аргументів у визначенні функції. У противному випадку з'явиться помилка, якщо ці аргументи будуть опущені при виклику функції.

Наприклад, ми хочемо внести опис статті в каталог. Користувач повинен ввести такі характеристики статті, як її назва, автор і короткий опис. Якщо користувач не вводить ім'я автора статті, вважаємо, що це Іванов Іван.

```
<?php
function Add_article($title, $description, $author="Іванов Іван"){
  echo "Заносимо в каталог статтю: $title,";
  echo "автор $author";
  echo "<br>Короткий опис: ";
  echo "$description <hr>";
}
Add_article("Інформатика і ми","Це стаття про інформатику ...","Петров
Петро");
Add_article("Хто такі хакеры", "Це стаття про хакерів ...");
?>
```

У результаті роботи скрипта одержимо наступне

```
Заносимо в каталог статтю: Інформатика і ми, автор Петров Петро.
Короткий опис:    Це стаття про інформатику...
Заносимо в каталог статтю: Хто такі хакеры, автор Іванов Іван.
Короткий опис: Це стаття про хакерів...
```

Якщо ж ми напишемо от так:

```
<?php
function Add_article($author="Іванов Іван", $title, $description){
// ...дії як у попередньому прикладі
}
Add_article("Хто такі хакеры", "Це стаття про хакеров...");
?>
```

То в результаті одержимо:

```
Warning: Missing argument 3 for
add_article() in c:\users\nina\tasks\func\def_bad.php
    on line 2
```

3. Списки аргументів змінної довжини.

У PHP4 можна створювати функції із змінною кількістю аргументів. Тобто ми створюємо функцію, не знаючи заздалегідь, зі скількома аргументами її викликають. Для написання такої функції ніякого спеціального синтаксису не потрібно. Усе робиться за допомогою вбудованих функцій func_num_args(), func_get_arg(), func_get_args().

Функція func_num_args() повертає кількість аргументів, переданих у поточну функцію. Ця функція може використовуватися тільки всередині визначення користувацької функції. Якщо вона з'явиться поза функцією, то інтерпретатор видасть попередження.

```
<?php
function DataCheck(){
  $n = func_num_args();
  echo "Число аргументів функції $n";
}
DataCheck();
   // виведе рядок "Число аргументів функції 0"
DataCheck(1,2,3);
   // виведе рядок "Число аргументів функції 3"
?>
```

Функція func_get_arg (ціле номер_аргументу) повертає аргумент зі списку переданих у функцію аргументів, порядковий номер якого заданий параметром номер_аргументу. Аргументи функції рахуються починаючи з нуля. Як і func_num_args(), ця функція може використовуватися лише в середині визначення якої-небудь функції.

Номер_аргументу не може перевищувати кількості аргументів, переданих у функцію. Інакше буде згенеровано попередження, і функція func_get_arg() поверне False.

Створимо функцію для перевірки типу даних, її аргументів. Вважаємо, що перевірка пройшла успішно, якщо перший аргумент функції – ціле число, другий – стрічка.

```
<?
function DataCheck(){
  $check =true;
  $n = func_num_args();
     // число аргументів,переданих у функцію

  /* перевіряємо, чи є перший
  переданий аргумент цілим числом */
  if ($n>=1) if (!is_int(func_get_arg(0)))
      $check = false;

  /* перевіряємо, чи є другий
  переданий аргумент стрічкою */

  if ($n>=2)
     if (!is_string(func_get_arg(1)))
        $check = false;
return $check;
}
if (DataCheck(a123,"text"))
   echo "Перевірка пройшла успішно<br>";
else echo "Дані не задовольняють умовам<br>";
if (DataCheck(324))
   echo "Перевірка пройшла успішно<br>";
else echo "Дані не задовольняють умовам<br>";
?>
```

Результатом роботи буде наступне.

```
Дані не задовольняють умовам
Перевірка пройшла успішно
```

Функція func_get_args() повертає масив, що складається зі списку аргументів, переданих функції. Кожен елемент масиву відповідає аргументові, переданому функції. Якщо функція використовується поза визначенням користувацької функції, то генерується попередження.

Перепишемо попередній приклад, використовуючи цю функцію. Будемо перевіряти, чи є цілим числом кожен парний аргумент, переданий функції:

```
<?
function DataCheck(){
  $check =true;
  $n = func_num_args();
  // число аргументів, переданих у функцію
  $args = func_get_args();
  // масив аргументів функції
  for ($i=0;$i<$n;$i++){
    $v = $args[$i];
    if ($i % 2 == 0){
      if (!is_int($v)) $check = false;
     // перевіряємо, чи є парний аргумент цілим
    }
  }
return $check;
}
if (DataCheck("text", 324))
   echo "Перевірка пройшла успішно<br>";
else echo "Дані не задовольняють умовам<br>";
?>
```

Як бачимо, комбінації функцій func_num_args(), func_get_arg() і func_get_args() використовується для того, щоб функції могли мати змінний список аргументів. Ці функції були додані тільки в PHP 4. У PHP3 для того, щоб домогтися подібного ефекту, можна використовувати як аргумент функції масив. Наприклад, от так можна написати скрипт, що перевіряє, чи є кожен непарний параметр функції цілим числом:

```
<?
function DataCheck($params){
     $check =true;
     $n = count($params);
  // число аргументів,переданих у функцію
  for ($i=0;$i<$n;$i++){
    $v = $params[$i];
    if ($i % 2 !== 0){
      // перевіряємо, чи є непарний аргумент цілим
if (!is_int($v)) $check = false;
    }
  }
return $check;
}
if (DataCheck("text", 324))
    echo "Перевірка пройшла успішно<br>";
else echo "Дані не задовольняють умовам<br>";
?>
```

4. Використання змінних всередині функції.

Глобальні змінні

Щоб використовувати всередині функції змінні, задані поза нею, ці змінні потрібно оголосити як глобальні. Для цього в тілі функції варто перелічити їхні імена після ключового слова global: global $var1, $var2;

```
<?
$a=1;
function Test_g(){
global $a;
  $a = $a*2;
  echo 'у результаті роботи функції $a=',$a;
}
echo 'поза функцією $a=',$a,', ';
Test_g();
echo "<br>";
echo 'поза функцією $a=',$a,', ';
Test_g();
?>
```

У результаті роботи цього скрипта одержимо:

```
поза функцією $a=1, у результаті роботи функції $a=2
поза функцією $a=2, у результаті роботи функції $a=4
```

Статичні змінні

Щоб використовувати змінні тільки всередині функції, при цьому зберігаючи їхні значення і після виходу з функції, потрібно оголосити ці змінні як статичні. Статичні змінні видно тільки всередині функції і не втрачають свого значення, якщо виконання програми виходить за межі функції. Оголошення таких змінних створюється за допомогою ключового слова static: static $var1, $var2;

Статичної змінної може бути присвоєне будь-яке значення, але не посилання.

```
<?
function Test_s(){
static $a = 1;
// не можна присвоїти вираз або посилання
  $a = $a*2;
  echo $a;
}
Test_s(); // виведе 2
echo $a;  // нічого не виведе, тому що $a доступна тільки
          // усередині функції
Test_s(); // всередині функції $a=2, тому результатом роботи функції
          // буде число 4
?>
```

Значення, що повертаються

Усі функції, наведені вище як приклади, виконували які-небудь дії. Крім подібних дій, будь-яка функція може повертати як результат своєї роботи яке-небудь значення. Це робиться за допомогою твердження return. Значення, що повертається, може бути будь-якого типу, включаючи списки й об'єкти. Коли інтерпретатор зустрічає команду return у тілі функції, він негайно припиняє її виконання і переходить на той рядок, з якого була викликана функція.

Наприклад, складемо функцію, що повертає вік людини. Якщо людина не вмерла, то вік вважається щодо поточного року.

```
<?php
/* якщо другий параметр обчислюється
як true, то він розглядається як
дата смерті, */
function Age($birth, $is_dead){
  if ($is_dead) return $is_dead-$birth;
  else return date("Y")-$birth;
}
echo Age(1971, false); // виведе 33
echo Age(1971, 2001); // виведе 30
?>
```

У цьому прикладі можна було і не використовувати функцію return, а просто замінити її функцією виведення echo. Однак якщо ми все-таки робимо так, що функція повертає якесь значення (у даному випадку вік людини), то в програмі ми можемо присвоїти будь-якій змінній значення цієї функції:

```
$my_age = Age(1981, 2004);
```

У результаті роботи функції може бути повернуте тільки одне значення. Кілька значень можна одержати, якщо повертати список значень (одномірний масив). Припустимо, ми хочемо одержати повний вік людини з точністю до дня.

```
<?php
function Full_age($b_day, $b_month, $b_year){
  if (date("m")>$b_month && date("d")>$b_day)
  {
    $day = date ("d") - $b_day;
    $month = date("m") - $b_month;
    $year = date("Y") - $b_year;
  } else {
    $year = date("Y") - $b_year - 1;
    $day = 31 - ($b_day - date ("d"));
    $month = 12 - ($b_month - date("m"));
  }
return array ($day,$month,$year);
}
$age = Full_age("07","08","1974");
echo "Вам $age[2] років, $age[1] місяців і $age[0] днів";
// виведе "Вам 29 років, 11 місяців і 5 днів"
?>
```

Коли функція повертає кілька значень для їхньої обробки в програмі, зручно використовувати мовну конструкцію list(), що дозволяє одною дією присвоїти значення відразу декільком змінним. Наприклад, у попередньому прикладі, залишивши без зміни функцію, обробити значення, що повертаються з неї, можна було так:

```
<?
// завдання функції Full_age()
list($day,$month,$year) = Full_age("07","08","1974");
echo "Вам $year років, $month місяців і $day днів";
?>
```

Взагалі конструкцію list() можна використовувати для присвоєння змінним значень елементів будь-якого масиву.

```
<?
$arr = array("first","second");
list($a,$b) = $arr;
   // змінної $a привласнюється перше
   // значення масиву, $b – друге
echo $a," ",$b;
   // виведе рядок «first second»
?>
```

5. Внутрішні (вбудовані) функції.

Кажучи про функції, створювані користувачем, все-таки не можна не сказати пари слів про вбудовані функції. З деякими з вбудованих функцій, такими як echo(), print(), date(), include(), ми вже познайомилися. Насправді всі перераховані функції, крім date(), є мовними конструкціями. Вони входять у ядро PHP і не вимагають ніяких додаткових налаштувань і модулів. Функція date() теж входить до складу ядра PHP і не вимагає налаштувань. Але є і функції, для роботи з якими потрібно встановити різні бібліотеки і підключити відповідний модуль. Наприклад, для використання функцій роботи з базою даних MySql варто скомпілювати PHP з підтримкою цього розширення. Останнім часом найбільш розповсюджені розширення і відповідно їхні функції споконвічно включають до складу PHP так, щоб з ними можна працювати без будь-яких додаткових налаштувань інтерпретатора.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

1. Описати функцію, яка обчислює факторіал цілого числа, і використати її для створення нового масиву чисел-факторіалів деякого заданого масиву (для задання масиву можна використати форму розроблену в завданні 1 попередньої лабораторної роботи).

2. Описати функції, які знаходять середнє арифметичне додатніх елементів масиву і середнє арифметичне абсолютних значень від'ємних елементів масиву (0 вважати ні додатнім ні від'ємним), і використати їх для порівняння знайдених значень (вивести,яка велечина є більшою).

3. Описати функцію, яка знаходить максимальний елемент масиву, зі всіх парних елементів масиву, і використати її для деякого заданого масиву.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Яким чином можна задати функцію.
2. Що собою являє аргумент функції.
3. Як оголосити глобальні змінні.
4. Що собою являють статичні змінні.

Лабораторна робота №9

Тема: Розробка PHP сценаріїв з використанням string-функцій.
Мета: Ознайомитися з можливостями PHP при роботі зі стрічками та навчится використовувати їх в своїх сценаріях.

ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

1. Стрічки.

В одній з попередніх лекцій ми наводили три способи завдання *стрічок*: за допомогою одинарних лапок, подвійних лапок і за допомогою heredoc–синтаксису. Відзначали ми й основні відмінності між цими способами. В основному вони стосуються обробки змінних і керуючих послідовностей в середині стрічки.

```
<?php
echo 'У такій стрічці НЕ обробляються змінні і більшість послідовностей';
echo "Тут змінні і послідовності обробляються";
echo <<<EOT
```

```
Тут теж обробляються як змінні, так і керуючі послідовності.
І крім того, можна вводити символи лапок
без їхнього екранування зворотним слешем.
EOT;
?>
```

Не одноразово, починаючи з найпершої лекції по PHP, ми використовували функцію echo. Насправді, echo – не функція, а мовна конструкція, тому використовувати при її виклику круглі дужки не обов'язково. Echo дозволяє виводити на екран стрічки, передані їй як параметри. Параметрів у echo може бути скільки завгодно. Їх розділяють комами або поєднують за допомогою оператора конкатенації і ніколи не беруть у круглі дужки.

```
<?
echo "Прийшов ", "побачив ", "переміг ";
// виведе Стрічка "Прийшов побачив переміг"
// багато хто воліє передавати більше
// параметрів у echo за допомогою конкатенації
echo "Прийшов " . "побачив " . "переміг ";
// теж виведе Стрічку
// "Прийшов побачив переміг"
echo ("Прийшов ", "побачив ", "переміг ");
// видасть помилку: unexpected ','
?>
```

Існує скорочений синтаксис для команди echo:

```
<?=Стрічка_для_виведення ?>
```

Тут параметр Стрічка_для_виведеня містить стрічку, задану будь-яким з відомих способів, що повинна бути виведена на екран. Наприклад, такий скрипт виведе на екран червоним кольором "Мене кличуть Вася":

```
<? $name="Вася" ?>
<font color=red>Мене кличуть <?=$name?></font>
```

Крім мовної конструкції echo існує ряд функцій для виведення стрічок. Це в першу чергу функція print і її різновиди printf, sprintf і т.п.

Функція print дозволяє виводити на екран тільки одну стрічку і, як і echo, не може бути викликана за допомогою змінних функцій, оскільки є мовною конструкцією.

Функція print_r не відноситься до стрічкових функцій, як можна було б подумати. Вона відображає інформацію про змінну у формі, зрозумілої користувачеві.

Функції sprintf і printf обробляють передану їм стрічку відповідно до заданого формату. Але про них ми говорити не будемо. А поговоримо про те, як можна здійснювати пошук у тексті, представленому у вигляді стрічки.

2. Пошук елемента в стрічці.

Для того щоб визначити, чи входить дана підстрічка до складу стрічки, використовується функція strpos(). Синтаксис strpos() такий:

```
strpos (задана стрічка,стрічка для пошуку [,з якого символу шукати])
```

Вона повертає позицію входження шуканої стрічки у задану стріку або повертає логічне false, якщо входження не знайдене. Додатковий аргумент дозволяє задавати символ,

починаючи з якого буде виконуватися пошук. Крім логічного false ця функція може повертати й інші значення, що зводяться до false (наприклад, 0 або ""). Тому для того, щоб перевірити, чи знайдена шукана стрічка, рекомендують використовувати оператор еквівалентності «===».

```
<? $str = "Ідея наносити дані на перфокарти і потім зчитувати й
обробляти їх
автоматично належала Джону Биллингсу,а її технічне рішення здійснив
Герман Холлерит.
Перфокарта Холлерита виявилася настільки вдалою, що без найменших змін
проіснувала
до наших днів.";
$pos = strpos($str,"Холлерит");
if ($pos !== false) echo "Шукана стрічка зустрінута в позиції номер
$pos ";
else echo "Шукана стрічка не знайдена";
/* помітимо, що ми перевіряємо значення $pos на еквівалентність з
false.
Інакше Стрічка, що знаходиться в першій позиції,не була б знайдена,
тому що 0
інтерпретується як false. */
?>
```

Якщо значення параметра стрічка_для_пошуку не є стрічкою, то воно перетворюється в цілий тип і розглядається як ASCII-код символу. Щоб одержати ASCII-код будь-якого символу в PHP, можна використати функцію ord("символ")

Наприклад, якщо ми напишемо $pos = strpos($str,228); то інтерпретатор буде вважати, що ми шукаємо символ «д». Якщо додати цей рядок у наведений вище приклад і вивести результат, то одержимо повідомлення, що шукана стрічка знайдена у позиції 1.

Функція, зворотна за змістом ord, – це chr (код символу). Вона по ASCII-коду виводить символ, що відповідає цьому кодові.

За допомогою функції strpos можна знайти номер тільки першого входження стрічки у задану стрічку. Природно, існують функції, що дозволяють обчислити номер останньої появи стрічки у заданій стрічці. Це функція strrpos(). Її синтаксис такий:

```
strrpos (задана стрічка, символ для пошуку)
```

На відміну від strpos() ця функція дозволяє знайти позицію останнього входження в стрічку зазначеного символу. Не можна шукати позицію стрічки, лише символу.

Бувають ситуації, коли знати позицію, де знаходиться шукана стрічка, не обов'язково, а потрібно просто одержати всі символи, що розташовані після входження цієї стрічки. Можна, звичайно, скористатися і наведеними вище функціями strpos() і strrpos(), але можна зробити і простіше – вирізати підстрічку за допомогою призначених саме для цього функцій.

3. Вирізання під стрічки.

Функція strstr

Говорячи про вирізання підстрічки із шуканої стрічки в мові PHP, у першу чергу варто відзначити функцію strstr():

```
strstr (задана стрічка, стрічка для пошуку)
```

Вона знаходить перше входження шуканої стрічки і повертає підстрічку, починаючи з цієї шуканої стрічки до кінця заданої стрічки.

Якщо стрічка для пошуку не знайдена, то функція поверне false. Якщо стрічка для пошуку не належить стічковому типові даних, то вона перетворюється в ціле число і розглядається як код символу. Крім того, ця функція чуттєва до регістра, тобто якщо ми будемо паралельно шукати входження слів «Ідея» і «ідея», то результати будуть різними. Замість strstr() можна використовувати абсолютно ідентичну їй функцію strchr().

Приклад. Виріжемо із стрічки, що містить назву й автора дослідження, підстрічку, що починається зі слова «Назва»:

```
<?
$str = "Автор: Іванов Іван (<a href=mailto:van@mail.ru>написати
лист</a>),
   Назва: 'Дослідження мов програмування' ";
echo "<b>Задана стрічка: </b>",$str;
if (!strstr($str, "Назва"))
   echo "Стрічка не знайдена<br>";
else echo "<p><b>Отримана підстрічка: </b>",strstr($str, "Назва");
?>
```

У результаті отримаємо:

```
Задана стрічка:
Автор: Іванов Іван (написати лист), Назва: 'Дослідження мов
програмування'
Отримана підстрічка: Назва: 'Дослідження мов програмування'
```

Для реалізації регістронезалежного пошуку підстрічки існує відповідний аналог цієї функції – функція stristr (задана стрічка, шукана стрічка). Діє і використовується вона аналогічно, як і strstr(), за винятком того, що регістр, у якому записані символи шуканої стрічки, не відіграє ролі при пошуку.
Очевидно, що функція strstr() не надто часто використовується – на практиці рідко буває потрібно отримувати підстрічку, що починається з визначеного слова або стрічки. Але в деяких випадках і вона може знадобитися. Крім того, у PHP є і більш зручні функції для пошуку входжень. Найбільш могутні з них, звичайно, зв'язані з регулярними виразами.

Функція substr

Іноді ми не знаємо, з яких символів починається шукана стрічка, але знаємо, наприклад, що починається вона з п'ятого символу і закінчується за два символи до кінця заданої стрічки. Як вирізати підстрічку по таким вимогам? Дуже просто, за допомогою функції substr(). Її синтаксис наступний:

```
substr (задана стрічка, позиція початкового символу [, довжина])
```

Ця функція повертає частину стрічки довжиною, заданою параметром довжина, починаючи із символу, зазначеного параметром позиція початкового символу. Позиція, з якої починається вирізувана підстрічка, може бути як додатнім цілим числом, так і від'ємним. В останньому випадку відлік елементів починається з кінця стрічки. Якщо параметр довжина опущений, то substr() повертає підстрічку від зазначеного символу і до кінця заданої стрічки. Довжина вирізуваної підстрічки теж може бути задана від'ємним числом. Це означає, що зазначене число символів відкидається з кінця стрічки.

Приклад. Припустимо, в нас є фраза, виділена жирним шрифтом за допомогою тега <b> мови HTML. Ми хочемо одержати цю ж фразу, але в звичайному стилі. Напишемо таку програму:

```
<?php
$word = "<b>Hello, world!</b>";
echo $word , "<br>";
$pure_str = substr($word, 3, -4);
/* виділяємо підстрічку, починаючи з 3-го символу,
   не включаючи 4 символи з кінця Стрічки */
echo $pure_str;
?>
```

Функція strip_tags

Насправді розв'язати таку задачу можна набагато простіше, за допомогою функції strip_tags:

```
strip_tags (стрічка [, припустимі теги])
```

Ця функція повертає стрічку, з якої вилучені всі html і php-теги. За допомогою додаткового аргументу можна задати теги, що не будуть вилучені з стрічки. Список з декількох тегів вводиться без яких-небудь розділових знаків. Функція видає попередження, якщо зустрічає неправильні або неповні теги.

```
<?php
$string = "<b>Bold text</b>
           <i>Italic text</i>";
$str = strip_tags($string);
   // видаляємо всі теги з Стрічки
$str1 = strip_tags($string, '<i>');
   // видаляємо всі теги крім тега <i>
$str2 = strip_tags($string, '<i><b>');
   // видаляємо всі теги крім тегов <i> і <b>
echo $str,"<br>",$str1,"<br>", $str2;
?>
```

Наведемо інший приклад використання функції substr(). Припустимо, в нас є якесь повідомлення з вітанням і підписом автора. Ми хочемо вирізати спочатку вітання, а потім і підпис, залишивши тільки змістовну частину повідомлення.

```
<?php
$text = "Привіт! Сьогодні ми вивчаємо роботу з Стрічкими. Автор.";
$no_hello = substr($text, 8);
    // забираємо вітання
$content = substr($text, 8, 39);
    // те ж саме, що substr($text, 8, -6).
    // Забираємо підпис.
echo $text, "<br>", $no_hello,
     "<br>", $content;
?>
```

У результаті одержимо:

```
Привіт! Сьогодні ми вивчаємо роботу з Стрічкими. Автор.
Сьогодні ми вивчаємо роботу з Стрічкими. Автор.
Сьогодні ми вивчаємо роботу з Стрічкими.
```

Якщо нам потрібно отримати один конкретний символ із стрічки, знаючи його порядковий номер, то не слід використовувати функції типу substr. Можна скористатися більш простим синтаксисом – записуючи номер символу у фігурних дужках після імені стрічкової змінної. У контексті попереднього приклада букву «р», розташовану другою по рахунку, можна отримати так:

```
echo $text{1}; // виведе символ "р"
```

Відмітимо, що номером цього символу є число один, а не два, тому що нумерація символів стрічки починається з нуля.

Оскільки ми почали говорити про символи в стрічці і їх нумерацію, то мимоволі виникає запитання, скільки всього символів у стрічці і як це обчислити. Кількість символів у стрічці – це довжина стрічки. Обчислити довжину стрічки можна за допомогою функції **strlen (Стрічка)**. Наприклад, довжина стрічки «Розробка інформаційної моделі» обчислюється за допомогою команди: strlen("Розробка інформаційної моделі"); і дорівнює 32 символам.

Отже, як вирізати і знаходити підстрічки, ми розглянули. Тепер навчимося заміняти стрічки, що входять до складу заданої стрічки, на іншу стрічку за нашим вибором.

4. Заміна входження під стрічки.

Функція str_replace

Для заміни підстрічки можна використовувати функцію str_replace(). Це проста і зручна функція, що дозволяє розв’язати безліч задач, що не вимагають особливих тонкостей при виборі замінимої підстрічки. Для того щоб робити заміни з більш складними умовами, використовують механізм регулярних виразів і відповідні функції ereg_replace() і preg_replace(). Синтаксис функції str_replace() такий:

```
str_replace(шукане значення, значення для заміни, об'єкт)
```

Функція str_replace() шукає в розглянутому об'єкті значення і заміняє його значенням, призначеним для заміни. Чому ми говоримо тут не про стрічки для пошуку і заміни і вихідну стрічку, а про значення й об'єкт, в якому відбувається заміна? Справа в тому, що починаючи з PHP 4.0.5 будь-який аргумент цієї функції може бути масивом.

Якщо об'єкт, в якому виконується пошук і заміна, є масивом, то ці дії виконуються для кожного елемента масиву й у результаті повертається новий масив.

```
<?php
$greeting = array("Привіт", "Привіт усім!","Привіт, дорога!"); //
об'єкт
$new_greet = str_replace("Привіт", "Добрий ранок", $greeting);
   // робимо заміну
print_r($new_greet);
/* одержимо: Array ([0]=>Добрий ранок [1]=>Добрий ранок усім!
[2]=>Добрий ранок, дорога!) */
?>
```

Якщо шукане значення і значення для заміни – масиви, то береться по одному значенню з кожного масиву і виконується їхній пошук і заміна в об'єкті. Якщо значень для заміни менше, ніж значень для пошуку, то як нові значення використовується порожня стрічка.

```
<?php
$greeting = array("Привіт", "Привіт усім!","Привіт, дорога!",
```

```
"Здрастуйте","Здрастуйте, товариші", "Ні");    // об'єкт
$search = array ("Привіт", "Здрастуйте", "Ні"); // значення, що будемо
заміняти
$replace = array ("Добрий ранок", "День добрий"); // значення, якими
будемо заміняти
$new_greet  =  str_replace($search,  $replace,  $greeting);//  робимо
заміну
print_r($new_greet); //виводимо отриманий масив
?>
```

У результаті одержимо такий масив:

```
Array ([0] => Добрий ранок
       [1] => Добрий ранок усім!
       [2] => Добрий ранок, дорога!
       [3] => День добрий
       [4] => День добрий, товариші
       [5] =>)
```

Якщо значення для пошуку – масив, а значення для заміни – стрічка, то ця стрічка буде використана для заміни всіх знайдених значень.

```
<?php $greeting = array("Привіт", "Привіт усім!", "Привіт, дорога!",
"Здрастуйте",
   "Здрастуйте, товариші");    // об'єкт
$search  =  array  ("Привіт","Здрастуйте");  //  значення,  що  будемо
заміняти
$replace = "День добрий"; // значення, яким будемо заміняти
$new_greet  =  str_replace($search,  $replace,  $greeting);  //  робимо
заміну
print_r($new_greet);//виводимо отриманий масив
?>
```

Одержимо:

```
Array ([0] => День добрий
       [1] => День добрий усім!
       [2] => День добрий, дорога!
       [3] => День добрий
       [4] => День добрий, товариші)
```

Функція str_replace() чуттєва до регістра, але існує її регістронезалежний аналог – функція str_ireplace(). Однак ця функція підтримується не у всіх версіях PHP.

Функція substr_replace

Ця функція сполучає у собі властивості двох вже розглянутих нами функцій – функції str_replace() і substr(). Її синтаксис такий:

```
substr_replace (вихідна стрічка, стрічка для заміни,
                                          позиція початкового символу [,
довжина])
```

Ця функція заміняє частину стрічки стрічкою, призначеною для заміни. Заміняється та частина стрічки (тобто підстрічка), що починається з позиції, зазначеної параметром позиція початкового символу. За допомогою додаткового аргументу довжина можна обмежити кількість замінимих символів. Тобто, фактично, ми не вказуємо конкретно стрічку, яку потрібно замінити, ми тільки описуємо, де вона знаходиться і, можливо, яку довжину має. У цьому відмінність функції substr_replace() від str_replace().

Як і у випадку з функцією substr() аргументи позиція початкового символу і довжина можуть бути від'ємними. Якщо позиція початкового символу від'ємна, то заміна виконується, починаючи з цієї позиції щодо кінця стрічки. Від'ємна довжина задає, скільки символів від кінця стрічки не повинна бути заміненою. Якщо довжина не вказується, то заміна відбувається до кінця стрічки.

```
<?php
$text = "Мене кличуть Вася.";
echo "Задана стрічка: $text<hr>\n";
/* Наступні два рядки замінять усю
задану стрічку рядком 'А мене – Петя' */
echo substr_replace($text, 'А мене – Петя', 0) . "<br>\n";
echo substr_replace($text, 'А мене – Петя', 0, strlen($text)) .
"<br>\n";
// Наступна стрічка додасть слово 'Привіт! '
// у початок заданої стрічки
echo substr_replace($text, 'Привіт! ', 0, 0). "<br>\n";
// Наступні два рядки замінять ім'я Вася
// на ім'я Іван у заданій стрічці
echo substr_replace($text, 'Іван', 11, -1) . "<br>\n";
echo substr_replace($text, 'Іван', -5, -1) . "<br>\n";
?>
```

У результаті роботи цього скрипта одержимо:

```
Вихідна стрічка: Мене кличуть Вася.


А мене – Петя
А мене – Петя
Привіт! Мене кличуть Вася.
Мене кличуть Іван.
Мене кличуть Іван.
```

5. Поділ і з'єднання стрічки.

Дуже корисні функції – функція поділу стрічки на частини і зворотна їй функція об'єднання стрічок в одну стрічку. Чому дуже корисні? Наприклад, якщо ви динамічно генеруєте форму за бажанням користувача, можна запропонувати йому вводити елементи для створення списку вибору, розділяючи їх яким-небудь символом. І для того щоб обробити отриманий список значень, саме і знадобиться вміння розбивати стрічку на частини. Для реалізації такої розбивки в PHP можна використовувати кілька функцій:

```
explode(роздільник, задана стрічка [,максимальна кількість елементів])
split (шаблон, задана стрічка [, максимальна кількість елементів])
preg_split (шаблон, задана стрічка [,максимальна кількість
елементів [,прапори]])
```

Останні дві функції працюють з регулярними виразами, тому в даній лекції ми їх розглядати не будемо. Розглянемо більш просту функцію – explode().

Функція explode() поділяє задану стрічку на підстрічки, кожна з яких відділена від сусідньої за допомогою зазначеного роздільника, і повертає масив отриманих стрічок. Якщо задано додатковий параметр максимальна кількість елементів, то кількість елементів у масиві буде не більше цього параметра, в останній елемент записується весь залишок стрічки. Якщо як роздільник зазначено порожню стрічку «""», то функція explode() поверне false. Якщо символу роздільника у заданій стрічці немає, то повертається задана стрічка без змін.

Приклад. Ми хочемо створити елемент форми – список, що випадає, і значення для цього списку повинен ввести користувач, не знайомий з мовою html. Створимо таку форму:

```
<form action=exp.php>
Введіть варіанти для вибору автора статті через двокрапку (":"):<br>
   <input type=text name=author size=40>
   <br>
   <input type=submit value=Створити елемент>
</form>
```

Скрипт, що буде її обробляти (exp.php), може бути таким:

```
<?php
$str = $_GET["author"];
$names = explode(":",$str);
    // розбиваємо стрічку введену, користувачем за допомогою ":"
$s = "<select name=author>";
    // створюємо список, що випадає
foreach ($names as $k => $name) {
   $s .= "<option value=$k>$name";
    // додаємо елементи до списку
}
$s .= "</select>";
echo $s;
?>
```

Крім поділу стрічки на частини іноді, навпаки, виникає необхідність об'єднання декількох стрічок в одне ціле. Функція, пропонована для цього мовою PHP, називається implode():

```
implode (масив стрічок, об’єднуючий елемент)
```

Ця функція об’єднує елементи масиву за допомогою переданого їй об'єднуючого елемента (наприклад, коми). На відміну від функції explode(), порядок аргументів у функції implode() не має значення.

Приклад. Припустимо, ми зберігаємо ім'я, прізвище і по батькові людини по окремості, а виводити їх на сторінці потрібно разом. Щоб з'єднати їх в одну стрічка, можна використовувати функцію implode():

```
<?php
$data = array("Іванов","Іван","Іванович");
$str = implode($data," ");
echo $str;
?>
```

У результаті роботи цього скрипта одержимо стрічку:

```
Іванов Іван Іванович
```

У функції implode() існує аналог – функція join(), тобто ці дві функції відрізняються лише іменами.

Отже, ми завершили знайомство з функціями роботи із стрічкими мови PHP. Звичайно ж, ми торкнулися далеко не всіх існуючих функцій, а розглянули лише малу частину. Ми вивчили функції, що дозволяють знайти набір символів у стрічці, функції, що заміняють усі входження однієї стрічки на іншу, функції поділу стрічки на частини і з'єднання декількох стрічок в одну.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

1. З допомогою форми вводиться стрічка і літера. Розробити сценарій для видалення з даної стрічки всіх введених літер

2. З допомогою форми вводиться стрічка і літера. Розробити сценарій підрахунку кількості входжень заданої літери в дану стрічку

3. З допомогою форми вводиться стрічка такої структури: між сусудніми словами ",", в кінці ".". Розробити сценарій для виведення введеної стрічки, але слова повинні бути розміщені в алфавітному порядку

4. З допомогою форми вводиться стрічка довільних символів. Розробити сценарій, що виводить задану стрічку розбиту комами по два символи

5. Через форму вводиться стрічка формату дати, наприклад (02.04.1978). Розробити сценарій, що перевіряє правильність введеної дати

6. З допомогою форми вводиться стрічка довільних символів. Розробити сценарій, що підраховує кількість різних символів цієї стрічки

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Яким чином можна оголосити стрічкову змінну.
2. Які функцію дозволяють вирізати частини стрічки.
3. Якою функцією можна замінити елементи стрічки.
4. Яким чином можна з’єднати стрічки.

Лабораторна робота №10

Тема: Розробка PHP сценаріїв з використанням файлів.
Мета: Ознайомитися із засобами роботи з файлами мови PHP.

ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Функції роботи з файловою системою PHP.

1.Функція fopen.

Взагалі кажучи, у PHP не існує функції, призначеної саме для створення файлів. Більшість функцій працюють із вже існуючими файлами у файловій системі сервера. Є кілька функцій, що дозволяють створювати тимчасові файли, або, що те ж саме, файли з унікальним для поточної директорії ім'ям. А от для того, щоб створити сам файл, потрібно скористатися функцією, що відкриває локальний або вилучений файл. Називається ця функція fopen(). Що означає «відкриває файл»? Це означає, що fopen пов'язує даний файл із потоком керування програми. Причому зв'язок буває різним у залежності від того, що ми хочемо робити з цим файлом: читати його, записувати в нього дані або робити і те й інше. Синтаксис цієї функції такий:

```
resource fopen ( ім'я_файлу, тип_доступу[, use_include_path])
```

У результаті роботи ця функція повертає вказівник на відкритий нею файл. Як параметри цієї функції передаються: ім'я файлу, який потрібно відкрити, тип доступу до файлу (визначається тим, що ми збираємося робити з ним) і, можливо, параметр, що визначає, чи шукати зазначений файл у include_path. Є ще один параметр, але про нього ми говорити не будемо, щоб не ускладнювати виклад. Обговоримо докладніше кожний з цих трьох параметрів.

Параметр ім'я_файлу повинен бути рядком, що містить вірне локальне ім'я файлу або URL-адреса файлу в мережі. Якщо ім'я файлу починається з вказівки протоколу доступу (наприклад, http://... або ftp://...), то інтерпритатор вважає це ім'я адресою URL і шукає оброблювач зазначеного в URL протоколу. Якщо оброблювач знайдений, то PHP перевіряє, чи дозволено працювати з об'єктами URL як зі звичайними файлами (директива allow_url_fopen). Якщо allow_url_fopen=off, то функція fopen викликає помилку і генерується попередження. Якщо ім'я файлу не починається з протоколу, то вважається, що зазначено ім'я локального файлу. Щоб відкрити локальний файл, потрібно, щоб PHP мав відповідні права доступу до цього файлу.

Параметр use_include_path, встановлений із значенням 1 або TRUE, змушує інтерпретатор шукати зазначений у fopen() файл у include_path. Нагадаємо, що include_path – це директива з файлу налаштувань PHP, що задає список директорій, в яких можуть знаходитися файли для включення. Крім функції fopen() вона використовується функціями include() і require().

Параметр тип_доступу може приймати одне з наступних значень (див. таб.).

Таблиця. Значення, що можуть прийматися параметром тип доступу

Тип доступу	Опис
r	Відкриває файл тільки для читання; установлює покажчик позиції у файлі на початок файлу.
r+	Відкриває файл для читання і запису; установлює покажчик файлу на його початок.
w	Відкриває файл тільки для запису; установлює покажчик файлу на його початок і усікає файл до нульової довжини. Якщо файл не існує, то намагається створити його.
w+	Відкриває файл для запису і для читання; установлює покажчик файлу на його початок і усікає файл до нульової довжини. Якщо файл не існує, то намагається створити його.
a	Відкриває файл тільки для запису; установлює покажчик файлу в його кінець. Якщо файл не існує, то намагається створити його.
a+	Відкриває файл для запису і для читання; установлює покажчик файлу в його кінець. Якщо файл не існує, то намагається створити його.
x	Створює і відкриває файл тільки для запису; поміщає покажчик файлу на його початок. Якщо файл вже існує, то fopen() повертає false і генерується попередження. Якщо файл не існує, то робиться спроба створити його. Цей тип доступу підтримується починаючи з версії PHP 4.3.2 і працює тільки з локальними файлами.
x+	Створює і відкриває файл для запису і для читання; поміщає покажчик файлу на його початок. Якщо файл вже існує, то fopen() повертає false і генерується попередження. Якщо файл не існує, то робиться спроба створити його. Цей тип доступу підтримується, починаючи з версії PHP 4.3.2, і працює тільки з локальними файлами.

Отже, щоб створити файл, потрібно, відкрити неіснуючий файл на запис.

```
<?php
$h = fopen("my_file.html","w");
/* відкриває на запис файл my_file.html,
якщо він існує, або створює порожній файл із таким ім'ям, якщо його ще
немає */
$h = fopen("dir/another_file.txt","w+");
```

```
/* відкриває на запис і читання або створює файл another_file.txt у
директорії dir */
$h = fopen("http://www.server.ru/dir/file.php","r");
/* відкриває на читання файл, що знаходиться по зазначеній адресі*/
?>
```

Створюючи файл, потрібно враховувати, під якою операційною системою ви працюєте, і під якою ОС приблизно цей файл буде читатися. Справа в тому, що різні операційні системи по-різному відзначають кінець рядка. У Unix-подібних ОС кінець рядка позначається \n, у системах типу Windows – \r\n. Windows пропонує спеціальний прапор t для перекладу символів кінця рядка систем типу Unix у свої символи кінця рядка. На противагу цьому існує прапор b, використовуваний найчастіше для бінарних файлів, завдяки якому такої трансляції не відбувається. Використовувати ці прапори можна, просто дописавши їх після останнього символу обраного типу доступу до файлу. Наприклад, відкриваючи файл на читання, замість r варто використовувати rt, щоб перекодувати всі символи кінця рядка в \r\n. Якщо не використовувати прапор b при відкритті бінарних файлів, то можуть з'являтися помилки, пов'язані зі зміною вмісту файлу. З розумінь перенесення програми на різні платформи рекомендується завжди використовувати прапор b при відкритті файлів за допомогою fopen().

Що відбувається, якщо відкрити або створити файл за допомогою fopen не вдається? У цьому випадку PHP генерує попередження, а функція fopen повертає як результат своєї роботи значення false. Такого роду попередження можна «придушити» (заборонити) за допомогою символу @ .
Наприклад, така команда не виведе попередження, навіть якщо відкрити файл не вдалося:

```
$h = @fopen("dir/another_file.txt","w+");
```

Таким чином, функція fopen() дозволяє створити лише порожній файл і зробити його доступним для запису. Як же записати дані в цей файл? Як прочитати дані з вже існуючого файлу?

Перш ніж відповісти на ці питання, розглянемо, як закрити встановлене за допомогою fopen() з'єднання.

2. Закриття з'єднання з файлом.

Після виконання необхідних дій з файлом, чи то читання або запис даних або яке-небудь інше, з'єднання, встановлене з цим файлом функцією fopen(), потрібно закрити. Для цього використовують функцію fclose(). Синтаксис якої наступний:

```
fclose (покажчик на файл)
```

Ця функція повертає TRUE, якщо з'єднання успішно закрито, і FALSE – у противному випадку. Параметр цієї функції повинен вказувати на файл, успішно відкритий, наприклад, за допомогою функції fopen().

```
<?php
$h = fopen("my_file.html","w");
fclose($h);
?>
```

Звичайно, якщо не закривати з'єднання з файлом, ніяких помилок виконання скрипта не відбудеться. Але в цілому для сервера це може мати серйозні наслідки. Наприклад, хакер може скористатися відкритим з'єднанням і записати у файл вірус, не говорячи вже про

зайву витрату ресурсів сервера. Так що радимо завжди закривати з'єднання з файлом після виконання необхідних дій.

3. Запис даних у файл.

Функція fwrite

Для того щоб записати дані у файл, доступ до якого відкритий функцією fopen(), можна використовувати функцію fwrite(). Синтаксис якої наступний:

```
int fwrite ( покажчик на файл, рядок [, довжина])
```

Ця функція записує вміст рядка стрічкою у файл, на який вказує покажчик на файл. Якщо зазначено додатковий аргумент довжину, то запис закінчується після того, як записана кількість символів, рівна значенню цього аргументу, або коли буде досягнутий кінець рядка.

У результаті своєї роботи функція fwrite() повертає кількість записаних байтів або false, у випадку помилки.

Приклад. Нехай у нашій робочій директорії немає файлу my_file.html. Створимо його і запишемо в нього рядок тексту:

```
<?php
$h = fopen("my_file.html","w");
$text = "Цей текст запишемо у файл.";
if (fwrite($h,$text))
  echo "Запис пройшов успішно";
else
  echo "Відбулася помилка при записі даних";
fclose($h);
?>
```

У результаті роботи цього скрипта в браузері ми побачимо повідомлення про те, що запис пройшов успішно, а у файлі my_file.html з'явиться рядок “Цей текст запишемо у файл.”. Якби цей файл існував до того, як ми виконали цей скрипт, усі дані, що знаходилися в ньому були б вилучені.

Якщо ж ми напишемо такий скрипт:

```
<?php
$h = fopen("my_file.html","a");
$add_text = "Додамо текст у файл.";
if(fwrite($h,$add_text,7))
  echo "Додавання тексту пройшло успішно<br>";
else echo "Відбулася помилка при додаванні даних<br>";
fclose($h);
?>
```

то до рядка, що вже існує у файлі my_file.html, додасться ще сім символів з рядка, що міститься в змінній $add_text, тобто слово «Додамо»

Далі ми розглянемо, які методи читання даних з файлу пропонує мова PHP.

4. Читання даних з файлу.

Якщо ми хочемо прочитати дані з існуючого файлу, однієї функції fopen(), як і у випадку з записом даних, недостатньо. Вона лише повертає покажчик на відкритий файл, але не зчитує ні одного рядка з цього файлу. Тому для того, щоб прочитати дані з файлу, потрібно скористатися однією зі спеціальних функцій: file, readfile, file_get_contents, fread, fgets і т.п.

Функція fread

Ця функція здійснює читання даних з файлу. Її можна використовувати і для читання даних з бінарних файлів, не боюючись їхнього пошкодження. Синтаксис fread() такий:

```
string fread (покажчик на файл, довжина)
```

При викликанні цієї функції відбувається читання даних довжиною (у байтах), визначеною параметром довжина, з файлу, на який вказує покажчик на файл. Параметр покажчик на файл повинен бути реально існуючою змінною типу ресурс, що містить у собі зв'язок з файлом, відкритим, наприклад, за допомогою функції fopen(). Читання даних відбувається доти, поки не зустрінеться кінець файлу або поки не буде прочитане зазначене параметром довжина число байтів.

У результаті роботи функція fread() повертає рядок з зчитаною з файлу інформацією.

Як ви помітили, у цій функції параметр довжина – обов'язковий. Отже, якщо ми хочемо зчитати весь файл у стрічку, потрібно знати її довжину. PHP може самостійно обчислити довжину зазначеного файлу. Для цього потрібно скористатися функцією filesize(ім'я файлу). У випадку помилки ця функція поверне false. На жаль, її можна використовувати лише для одержання розміру локальних файлів.

Приклад. Прочитаємо вміст файлу my_file.html

```
<?php
$h = fopen("my_file.html","r+");
// відриваємося файл на запис і читання
$content = fread($h,filesize("my_file.html"));
// зчитуємо вміст файлу в стрічку
fclose($h); // закриваємо з'єднання з файлом
echo $content;
// виводимо вміст файлу на екран браузера
?>
```

Для того щоб зчитати вміст бінарного файлу, наприклад зображення, у таких системах, як Windows, рекомендується відкривати файл за допомогою прапора rb або йому подібного, що містять символ b наприкінці.

Функція filesize() кешує результати своєї роботи. Якщо змінити вміст файлу my_file.html і знову запустити наведений вище скрипт, то результат його роботи не зміниться. Більше того, якщо запустити скрипт, що зчитує дані з цього файлу за допомогою іншої функції (наприклад, fgetss), то результат може виявитися таким, як якби файл не змінився. Щоб цього уникнути, потрібно очистити статичний кэш, додавши в код програми команду clearstatcache();

Функція fgets

За допомогою функції fgets() можна зчитати з файлу стрічку тексту. Синтаксис цієї функції практично такий самий, як і в fread(), за вийнятком того, що довжину стрічки, що зчитується, вказувати необов'язково:

```
string fgets ( покажчик на файл [, довжина])
```

У результаті роботи функція fgets() повертає рядок довжиною (довжина–1) байт із файлу, на який вказує покажчик на файл. Читання закінчується, якщо прочитано (довжину–1) символів і зустрівся символ кінця стрічки або кінець файлу. Нагадаємо, що в PHP один символ – це один байт. Якщо довжина стрічки, що зчитується, не зазначена (дана можливість з'явилася починаючи з PHP 4.2.0), то зчитується 1 Кбайт (1024 байт) тексту або, що те саме, 1024 символу. Починаючи з версії PHP 4.3, якщо параметр довжина не задана, зчитується стрічка цілком. У випадку помилки функція fgets() повертає false. Для версій PHP починаючи з 4.3 ця функція безпечна для двійкових файлів.

```
<?php
$h = fopen("my_file.html","r+");
$content = fgets($h,2);
// зчитує перший символ з першого рядка файлу my_file.html
fclose($h);
echo $content;
?>
```

Обидві функції, fread() і fgets(), припиняють зчитування даних з файлу, якщо зустрічають кінець файлу. У PHP є спеціальна функція, що перевіряє, чи досягнув вказівник позиції файлу на кінець файлу. Це булева функція **feof()**, як параметр якої передається вказівник на з'єднання з файлом.
Наприклад, ось так можна зчитати всі рядки файлу my_file.html:

```
<?php
$h = fopen("my_file.html","r");
while (!feof ($h)) {
    $content = fgets($h);
    echo $content,"<br>";
}
fclose($h);
?>
```

Функція fgetss

Існує різновид функції fgets() – функція fgetss(). Вона теж дозволяє зчитувати стрічкук із зазначеного файлу, але при цьому видаляє з нього всі html-теги, що зустрілися, за винятком, можливо, деяких. Синтаксис fgetss() такий:

```
string fgetss(покажчик на файл, довжина [, припустимі теги])
```

Зверніть увагу, що тут аргумент довжина обов'язковий.

Приклад. Нехай у нас існує файл my_file.html наступного змісту:

```
<h1>Без праці не виймеш і рибку зі ставка</h1>
<b>Тихіше їдеш – далі будеш</b>
У семи няньок<i> дитя без ока</i>.
Виведемо на екран усі стрічки файлу my_file.html,
видаливши з них всі теги, крім <b> і <i>:
<?php
$h = fopen("my_file.html","r");
while (!feof ($h)) {
   $content = fgetss($h,1024,'<b><i>');
   echo $content,"<br>";
}
fclose($h);
```

```
?>
```

У результаті роботи цього скрипта одержимо:

```
Тихіше їдеш – далі  будеш
Без праці не виймеш і рибку зі ставка.
У семи няньок дитя без ока.
```

Функція fgetc

Природньо, якщо можна зчитувати інформацію з файлу пострічково, то можна зчитувати її і посимвольно. Для цього призначена функція fgetc(). Легко здогадатися, що синтаксис у неї наступний:

```
string fgetc ( покажчик на файл )
```

Ця функція повертає символ з файлу, на який посилається вказівник на файл, і значення, що обчислюється як FALSE, якщо зустрінуто кінець рядка.

От так, наприклад, можна зчитати файл по одному символу:

```
<?php
$h = fopen("my_file.html","r");
while (!feof ($h)) {
    $content = fgetc($h);
    echo $content,"<br>";
}
fclose($h);
?>
```

Насправді для того, щоб прочитати вміст файлу, відкривати з'єднання з ним за допомогою функції fopen() зовсім не обов'язково. У PHP є функції, що дозволяють робити це, використовуючи лише ім'я файлу. Це функції readfile( ), file( ) і file_get_contents( ). Розглянемо кожну них них докладніше.

Функція readfile

Синтаксис:

```
int readfile ( ім'я_файлу [, use_include_path])
```

Функція readfile() зчитує файл, ім'я якого передане їй як параметр ім'я_файлу, і виводить його вміст на екран. Якщо додатковий аргумент use_include_path має значення TRUE, то пошук файлу з заданим ім'ям виконується і по директоріях, що входять у include_path.

У програму ця функція повертає кількість зчитатих байтів (символів) файлу, а у випадку помилки – FALSE. Повідомлення про помилку в цій функції можна ”придушити” оператором @ .

Приклад. Наступний скрипт виведе на екран вміст файлу my_file1.html і розмір цього файлу, якщо він існує. У противному випадку виведеться нам повідомлення про помилку – рядок "Error in readfile".

```
<?php
$n = @readfile ("my_file1.html");
/* виводить на екран вміст файлу і записує його розмір у змінну $n */
if (!$n) echo "Error in readfile";
```

```
/* якщо функція readfile() виконалася с помилкою, то $n=false і
виводимо
повідомлення про помилку */
else echo $n;
  // якщо помилки не було, то виводимо число кількості символів
?>
```

За допомогою функції readfile() можна читати вміст віддалених файлів, вказуючи їхню URL-адресу як ім'я файлу, якщо ця опція не відключена в налаштуваннях сервера.

Відразу ж виводити вміст файлу на екран не завжди зручно. Інколи потрібно записати інформацію з файлу в змінну, щоб надалі зробити з нею які-небудь дії. Для цього можна використовувати функцію file() або file_get_contents().

Функція file

Функція file() призначена для зчитування інформації з файлу в змінну типу масив. Синтаксис у неї такий же, як і у функції readfile(), за винятком того, що в результаті роботи вона повертає масив:

```
array file ( ім'я_файлу [, use_include_path])
```

Що ж за масив повертає ця функція? Кожен елемент даного масиву є стрічкою у файлі, інформацію з якого ми зчитуємо (його ім'я задане аргументом ім'я_файлу). Символ нового рядка теж включається в кожен з елементів масиву. У випадку помилки функція file(), як і всі вже розглянуті, повертає false. Додатковий аргумент use_include_path знову ж визначає, чи шукати даний файл у директоріях include_path. Відкривати віддалені файли за допомогою цієї функції теж можна, якщо не заборонено сервером. Починаючи з PHP 4.3 робота з бінарними файлами за допомогою цієї функції стала безпечною.

Наприклад, маємо файл my_file.html наступного змісту:

```
<h1>Без праці не виймеш і рибку зі ставка</h1>
<b>Тихіше їдеш – далі будеш</b>
Прочитаємо його вміст за  допомогою функції file():
<?php
$arr = file ("my_file.html");
foreach($arr as $i => $a) echo $i,": ", htmlspecialchars($a), "<br>";
?>
```

У результаті на екран буде виведене наступне повідомлення:

```
0: <h1>Без праці не виймеш і рибку зі ставка</h1>
1: <b>Тихіше їдеш – далі будеш</b>
```

Функція file_get_contents

У версіях PHP починаючи з 4.3 з'явилася можливість зчитувати вміст файлу в стрічку. Робиться це за допомогою функції file_get_contents(). Як і дві попередні функції, як параметри вона приймає значення імені файлу і, можливо, дати вказівку шукати його в директоріях include_path. Наведемо її синтаксис:

```
string file_get_contents (ім'я_файлу [, use_include_path])
```

Ця функція абсолютно ідентична функції file(), тільки повертає вона вміст файлу у вигляді стрічки. Крім того, вона безпечна для обробки бінарних даних і може зчитувати інформацію з віддалених файлів, якщо це не заборонено налаштуваннями сервера.

5. Перевірка існування файлу.

Отже, створювати файл ми навчилися, записувати дані в нього – навчилися, зчитувати дані з файлу – теж навчилися. Але от питання: а що якщо файлу, з яким ми намагаємося проробити всі ці операції, не існує? Або він недоступний для читання або запису? Очевидно, що в такому випадку жодна з вивчених нами функцій працювати не буде і PHP видасть повідомлення про помилку. Щоб відслідковувати такого роду помилки, можна використовувати функції file_exists(), is_writable(), is_readable().

Функція file_exists

Синтаксис:

```
bool file_exists (ім'я файлу або директорії)
```

Функція file_exist() перевіряє, чи існує файл або директорія, ім'я якої передане їй як аргумент. Якщо директорія або файл у файловій системі сервера існує, то функція повертає TRUE, у противному випадку – FALSE. Результат роботи цієї функції кешується. Відповідно очистити кэш можна, як вже відзначалося, за допомогою функції clearstatcache(). Для нелокальних файлів використовувати функцію file_exists() не можна.

```
<?php
$filename = 'c:/users/files/my_file.html';
if (file_exists($filename)) {
  print "Файл <b>$filename</b> існує";}
else {  print "Файл <b>$filename</b> НЕ існує";
}
?>
```

Функція is_writable

Якщо крім перевірки існування файлу потрібно довідатися ще, чи дозволено записувати інформацію в цей файл, варто використовувати функцію is_writable() або її аналог – функцію is_writeable().
Синтаксис:

```
bool is_writable (ім'я файлу або директорії)
```

Ця функція повертає TRUE, якщо файл (або директорія) існує і доступний для запису. Доступ до файлу здійснюється під тим обліковим записом користувача, під яким працює сервер (найчастіше це користувач nobody або www). Результати роботи функції is_writable кешуються.

Функція is_readable

Якщо крім перевірки існування файлу потрібно довідатися ще, чи дозволено читати інформацію з нього, потрібно використовувати функцію is_readable().

Синтаксис:

```
bool is_readable (ім'я файлу)
```

Ця функція працює подібно функції is_writable().

```
<?php
```

```
$filename = 'c:/users/files/my_file.html';
if (is_readable($filename)) {
  print "Файл <b>$filename</b> існує і доступний для читання";}
else {  print "Файл <b>$filename</b> НЕ існує або НЕ доступний для
читання";
}
?>
```

6. Видалення файлу.

Останнє, що ми хочемо вивчити з дій над файлами, – це видалення файлів. Для того щоб видалити файл за допомогою мови PHP, потрібно скористатися функцією unlink(). Синтаксис цієї функції наступний:

```
bool unlink ( ім'я_файлу)
```

Дана функція знищує файл, що має ім'я ім'я_файлу, повертає TRUE у випадку успіху цієї операції і FALSE – у випадку помилки. Щоб видалити файл, потрібно теж мати відповідні права доступу до нього (наприклад, доступу тільки на читання для видалення файлу недостатньо).

7. Завантаження файлу на сервер.

Тепер розв'яжемо більш складну і часто виникаючу на практиці задачу завантаження файлу на сервер. Перше, що потрібно зробити, щоб завантажити файл на сервер, це створити html-форму. Для того щоб за допомогою цієї форми можна було завантажувати файли, вона повинна містити атрибут enctype в тезі form зі значенням multipart/form-data, а також елемент input типу file.

Приклад.

```
<form enctype="multipart/form-data" action="parse.php" method="post">
  <input type="hidden" name="MAX_FILE_SIZE" value="30000" />
  Завантажити файл: <input type="file" name="myfile" /><br>
  <input type="submit" value="Відправити файл" />
</form>
```

Відмітимо, що ми додали у формі сховане поле, що містить у собі максимально допустимий розмір файлу, що завантажується, у байтах. При спробі завантажити файл, розмір якого більше зазначеного в цьому полі значення, буде зафіксована помилка. У браузері створена нами форма буде виглядати як стрічка для введення тексту з додатковою кнопкою для вибору файлу з локального диска.

Тепер потрібно написати скрипт, що буде обробляти отриманий файл.

Вся інформація про завантажений на сервер файл міститься в глобальному масиві $_FILES. Цей масив з'явився починаючи з PHP 4.1.0. Якщо вімкнено директиву register_globals, то значення переданих змінних доступні просто по їхніх іменах.

Якщо ми завантажили з комп'ютера-клієнта файл з ім'ям critics.htm розміром 15136 байт, то скрипт із єдиною командою print_r($_FILES); виведе на екран наступне:

```
Array ( [myfile] =>  Array ( [name] => critics.htm
                             [type] => text/html
                             [tmp_name] => C:\WINDOWS\TEMP\php49F.tmp
                             [error] => 0
                             [size] => 15136 )
)
```

Взагалі кажучи, масив $_FILES завжди має наступні елементи:

- $_FILES['myfile']['name'] – ім'я, що мав файл на машині клієнта.
- $_FILES['myfile']['type'] – mime-тип відправленого файлу, якщо браузер надав цю інформацію. У нашому прикладі це text/html.
- $_FILES['myfile']['size'] – розмір завантаженого файлу в байтах.
- $_FILES['myfile']['tmp_name'] – тимчасове ім'я файлу, під яким він був збережений на сервері.
- $_FILES['myfile']['error'] – код помилки, що з'явилася при завантаженні.

Тут 'myfile' – це ім'я елемента форми, за допомогою якого було зроблене завантаження файлу на сервер. Тобто воно може бути іншим, якщо елемент форми назвати інакше. Але от інші ключі (name, type і т.д.) залишаються незмінними для будь-якої форми.

Якщо **register_globals=On**, то доступні також додаткові змінні, такі як $myfile_name, що еквівалентна $_FILES['myfile']['name'], і т.п.

Помилок при завантаженні в PHP виділяють п'ять типів і відповідно $_FILES['myfile']['error'] може приймати п'ять значень:

- 0 – помилки не відбулося, файл завантажений успішно
- 1 – файл, що завантажується, перевищує розмір, встановлений директивою upload_max_filesize у файлі налаштувань php.ini
- 2 – файл, що завантажується, перевищує розмір, встановлений елементом MAX_FILE_SIZE форми html
- 3 – файл був завантажений частково
- 4 – файл завантажений не був

По замовчуванні завантажені файли зберігаються в тимчасовій директорії сервера, якщо інша директорія не зазначена за допомогою опції upload_tmp_dir у файлі налаштувань php.ini. Перемістити завантажений файл у потрібну директорію можна за допомогою функції move_uploaded_file().

Функція move_uploaded_file() має наступний синтаксис:

```
bool move_uploaded_file (тимчасове_ім'я_файлу, місце_призначення)
```

Ця функція перевіряє, чи дійсно файл, позначений рядком тимчасове_ім'я_файлу, був завантажений через механізм завантаження HTTP методом POST. Якщо це так, то файл переміщається у файл, заданий параметром місце_призначення (цей параметр містить як шлях до нової директорії для збереження, так і нове ім'я файлу).

Якщо тимчасове_ім'я_файлу задає неправильний завантажений файл, то ніяких дій зроблено не буде, і move_uploaded_file() поверне FALSE. Те ж саме відбудеться, якщо файл із якихось причин не може бути переміщений. У цьому випадку інтерпретатор виведе відповідне попередження. Якщо файл, заданий параметром місце_призначення, існує, то функція move_uploaded_file() перезапише його.

Підведемо підсумки. У цій лекції ми вивчили, як створювати файли за допомогою мови PHP, як записувати дані у файли за допомогою PHP, як зчитувати з них інформацію різними способами, як перевіряти існування і приступність файлу для запису і читання. Крім того, ми розглянули задачу завантаження файлу на сервер і обговорили основні зв'язані з нею перемінні і функції мови PHP.

Для роботи із файловою системою Web-сервера в мові реалізовано наступні процедури та функції:

file_exists(string назва_файлу) – повертає значення істини, якщо файл із заданим ім'ям існує.

is_file (string назва_файлу) – перевіряє існування вказаного файлу, а також можливість виконання з ним операцій читання-запису.

filesize(string назва_файлу) – повертає розмір файлу в байтах.

fopen (string назва_файлу, string режим доступу) – відкриває вказаний файл у вказаному режимі, "r" – читання, "w" –запису, "a" – додання даних, а також повертає цілочисельне значення (ідентифікатор відкритого файлу). Зазначений файл може бути розміщений на комп'ютері, на якому виконується Web-сервер, на іншому HHTP чи FTP-сервері.

fclose (int ідентифікатор відкритого файлу) – закриває файл із вказаним ідентифікатором.

fread(int ідентифікатор, int кількість_байт) – зчитує із файлу із заданим ідентифікатором вказану кількість байт.

fgetc(int ідентифікатор) – зчитує із файлу один символ.

fgets(int ідентифікатор, int довжина) – зчитує із файлу стрічку вказаної довжини.

file(назва_файлу) –завантажує файл в індексований масив, кожен елемент якого відповідає одній стрічці файлу.

int fwrite(int ідентифікатор, string стрічка) – здійснює запис значення рядкової змінної у файл.

int readfile (string назва_файлу) – зчитує вміст файлу і виводить його у вікно браузера.

Приклад: описати сценарій-лічильник, який при кожному своєму запуску, збільшує збільшував число, записане у файлі counter.dat

```
<?php
   $f_name="counter.dat";
   if (file_exists($f_name)){
     $mas=file($f_name);
     $count=$mas[0]+1;
   }
   else {
     $count=1;
   }
   $f=fopen("counter.dat","w");
     fwrite($f,$count);
     fclose($f);
   echo "Кількість відвідувачів:";
     readfile($f_name);
 ?>
```

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

1. Розробити сценарій, що перевірятиме чи в даному файлі міститься введена стрічка.
2. Розробити сценарій для виконання авторизації користувача. Дані про користувачів зберігати у файлі виду:
   3. назва_користувача_1:пароль_користувача_1

4. назва_користувача_2:пароль_користувача_2
5. .....
назва_користувача_N:пароль_користувача_N

6. Розробити сценарій, який буде заносити в файл з форми потрібні дані і відразу ж їх виводити в необхідному форматі. (див приклад)

   Вказівка: повинно бути два файли: перший повинен містити саму форму і виводити вміст, другий повенен додавати дані в файл і переадресовуватись відразу на перший (**Header("Location: файл на який переадресовується");**)

7. Розробити сценарій з допомогою якого можна проводити опитування і перегляд його результатів

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Опишіть етапи роботи із файлами.
2. Що таке дескриптор файлу?
3. Опишіть результат виклику функцій fopen("f.dat","r") та fopen("f.dat","w") у випадку коли файл f.dat не існує.
4. Якими, на Вашу думку, можуть бути наслідки не закриття файлу
   a. на локальному комп'ютері?
   b. на HHTP чи FTP сервері
5. Опишіть відмінності функцій fgets() та file(). Який недолік має використання функції fgets()?

Лабораторна робота №11

Тема: Розробка PHP сценаріїв з застосуванням сесій .
Мета: Ознайомитися із засобами механізму сесій мови PHP.

ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Авторизація доступу за допомогою сесій .

Ця частина лекції присвячена вивченню питань забезпечення безпеки в мережі і використанню для цих цілей механізму сесій. Розглядаються: ініціалізація сесій, передача ідентифікатора користувача, реєстрація змінних сесії, знищення сесії. На завершення наводиться приклад авторизації користувача за допомогою механізму сесій.

Також ми розберемо, що таке сесії й у чому їхня специфіка в PHP, розв'яжемо одну з основних задач, що виникає при побудові більш-менш складних інформаційних систем (сайтів) - задачу авторизації доступу користувачів до ресурсів системи, а також обговоримо безпеку побудованого розв'язку.

1. Авторизація доступу.

Що таке авторизація доступу? Спробуємо пояснити на прикладі зі звичайного життя. Ви хочете взяти в бібліотеці книгу. Але ця послуга доступна тільки тим, у кого є читацький квиток. Можна сказати, що за допомогою цього квитка виконується "авторизація доступу" до бібліотечних ресурсів. Бібліотекар після пред'явлення йому читацького квитка знає, хто бере книгу, і в разі потреби (наприклад, книгу довго не повертають) може вжити заходів (подзвонити боржникові додому). Бібліотекар має набагато більше прав, ніж звичайний відвідувач: він може давати або не давати книги визначеному відвідувачеві, може виставляти на показ новинки і забирати в архів книги, що рідко читаються, і т.п.

В інформаційних технологіях усе приблизно так само. У мережі існує величезна кількість ресурсів, тобто безліч "бібліотек". У кожної з них свій "бібліотекар", тобто людина або група людей, що відповідають за зміст ресурсу і надання користувачам інформації. Їх називають адміністраторами. Функції адміністратора, як правило, включають додавання нової інформації, знищення і редагування існуючої, налаштування способів відображення інформації користувачеві. А в функції користувача (простого відвідувача ресурсу) входить тільки пошук і перегляд інформації.

Як же відрізнити користувача від адміністратора? У реальній бібліотеці це очевидно, але якщо ролі бібліотекаря і відвідувача бібліотеки перенести у віртуальну реальність, то ця очевидність зникає. Бібліотекар, як і відвідувач, має доступ до бібліотечних ресурсів через Internet. А відповідно до протоколу HTTP усі клієнти абсолютно рівноправні. Як же зрозуміти, хто зайшов на сайт? Звичайний користувач (відвідувач) чи адміністратор (бібліотекар)? Якщо це простий користувач, то як зберегти цю інформацію, щоб не допустити відвідувача в закриті архіви сайту? Тобто виникає питання, як ідентифікувати клієнта, що надіслав запит, і зберігати відомості про нього, поки він знаходиться на сайті?

Найпростіший варіант, що приходить у голову, - це реєстрація людини в системі і видача йому аналога читацького квитка, а саме логіна і пароля для входу в адміністративну частину системи. Ця інформація зберігається на комп'ютері-сервері, і при вході в систему перевіряється відповідність введених користувачем логіна і пароля тим, що зберігаються в системі. Правда, тут у порівнянні з реальною бібліотекою ситуація змінюється: читацький квиток потрібно бібліотекареві для входу в закриту частину системи, а читач може заходити на сайт вільно. У принципі можна реєструвати і простих відвідувачів. Тоді всіх зареєстрованих користувачів потрібно розділити на групи: бібліотекарі (адміністратори) і читачі (прості користувачі), наділивши їх відповідними правами. Ми не будемо вдаватися в ці тонкості і скористаємося самим простим варіантом, коли введення логіна і пароля потрібно для доступу до деяких сторінок сайта.

Приклад.
У нас є файл index.html - домашня сторінка Васі Петрова

```
<html>
<head><title>My home page</title></head>
<body>
Привіт усім!
Мене звуть Вася Петров і це моя домашня сторінка.
<a href="secret_info.html">Для Петра</a>
</body></html>
```

і файл **secret_info.html**, що містить секретну інформацію, читати яку дозволено тільки другові Василя - Петру.

```
<html>
<head><title>Secret info</title></head>
<body>
Тут я хочу поділитися секретами з другом Петром.</p>
```

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

Якщо залишити обидва ці файли як є, то будь-який відвідувач, кликнувши на посилання "Для Петра", потрапить на секретну сторінку. Щоб цього уникнути, потрібно додати проміжний скрипт, що буде перевіряти, чи дійсно Петя хоче потрапити на секретну сторінку. І зробити так, щоб головний файл посилався не відразу на secret_info.html, а спочатку на цей скрипт.

```
<html>
 <head><title>My home page</title></head>
 <body>
  <p>Привіт усім!
  Мене звуть Вася Петров і це моя домашня сторінка.
  </p>
  <a href="authorize.php">Для Петра</a>
 </body>
</html>
```

Сам скрипт авторизації повинен містити форму для введення логіна і пароля, перевіряти їхню правильність і перенаправляти на секретну сторінку, якщо перевірка пройшла успішно, і видавати повідомлення про помилку в противному випадку.

```
<?
if (!isset($_GET['go'])){
    // перевіряємо, чи відправлені дані формою
    echo "<form>
    // форма для авторизації (введення логіна і пароля)
    Login: <input type=text name=login>
    Password: <input type=password name=password>
    <input type=submit name=go value=Go>
    </form>";}
else {
// якщо форма заповнена, то порівнюємо логін
// і пароль із правильними логіном і  паролем
    if ($_GET['login']=="pit" && $_GET['passwd']=="123") {
    Header("Location: secret_info.html");
    //і перенаправляємо на секретну сторінку }
else echo "Невірне введення, спробуйте ще раз<br>";}
?>
```

Начебто усе досить просто. Але припустимо, у нас не одна секретна сторінка, а декілька. Причому вони пов'язані між собою перехресними посиланнями. Тоді виникає необхідність постійно пам'ятати пароль і логін відвідувача сайта (якщо він такий має). Щоб розв’язати цю проблему, можна в кожну сторінку вмонтувати скрипт, що буде передавати логін і пароль від сторінки до сторінки як сховані параметри форми. Але такий спосіб не зовсім безпечний: ці параметри можна перехопити і підробити. У PHP існує більш зручний і безпечний метод розв’язання такої проблеми збереження даних про відвідувача протягом сеансу його роботи із сайтом - це механізм сесій.

2. Механізм сесії.

Сесії - це механізм, що дозволяє створювати і використовувати змінні, що зберігають своє значення протягом усього часу роботи користувача із сайтом.

Ці змінні для кожного користувача мають різні значення і можуть використовуватися на будь-якій сторінці сайта до виходу користувача із системи. При цьому щораз, заходячи на сайт, користувач одержує нові значення змінних, що дозволяють ідентифікувати його протягом цього сеансу або сесії роботи із сайтом. Звідси і назва механізму - сесії.

Задача ідентифікації користувача розв’язується шляхом присвоєння кожному користувачеві унікального номера, так званого ідентифікатора сесії (SID, Session IDentifier). Він генерується PHP у той момент, коли користувач заходить на сайт, і знищується, коли користувач іде із сайта, і являє собою рядок з 32 символів (наприклад, ac4f4a45bdc893434c95dcaffb1c1811). Цей ідентифікатор передається на сервер разом з кожним запитом клієнта і повертається назад разом з відповіддю сервера.

Існує кілька способів передачі ідентифікатора сесії:

- За допомогою cookies.
  Cookies були створені спеціально як метод однозначної ідентифікації клієнтів і являють собою розширення протоколу HTTP. У цьому випадку ідентифікатор сесії зберігається в тимчасовому файлі на комп'ютері клієнта, що послав запит. Метод, безсумнівно, гарний, але багато користувачів відключають підтримку cookies на своєму комп'ютері через проблеми з безпекою.

- За допомогою параметрів командного рядка.
  У цьому випадку ідентифікатор сесії автоматично вбудовується в усі запити (URL), передані серверові, і зберігається на стороні сервера.

Наприклад: адреса
http://green.nsu.ru/test.php перетворюється на адресу
http://green.nsu.ru/test.php?PHPSESSID=ac4f4a45bdc893434c95dcaffb1c1811

Цей спосіб передачі ідентифікатора використовується автоматично, якщо в браузера, що відправив запит, вимкнені cookies. Він досить надійний - передавати параметри в командному рядку можна завжди. З іншого боку, ідентифікатор сесії можна підглянути, скористатися збереженим варіантом у рядку браузера або підробити. Хоча, звичайно, усі ці проблеми або надумані або їх можна розв’язати. Наприклад, хто зможе запам'ятати рядок з 32 різних символів? А якщо правильно організувати роботу із сесіями (вчасно їх знищувати), то навіть збережений у браузері номер сесії нічого не дасть. До питань безпеки ми ще повернемося наприкінці лекції.

Крім перерахованих варіантів передачі ідентифікатора сесії, відомо ще декілька, але ми їх розглядати не будемо через їхню складність.

3. Створення сесії.

Перше, що потрібно зробити для роботи із сесіями, це запустити механізм сесій. Якщо в налаштуваннях сервера змінна session.auto_start встановлена в значення "0" (якщо session.auto_start=1, то сесії запускаються автоматично), то будь-який скрипт, у якому потрібно використовувати дані сесії, повинен починатися з команди

```
session_start();
```

Одержавши таку команду, сервер створює нову сесію або відновлює поточну, ґрунтуючись на ідентифікаторі сесії, переданому по запиті. Як це робиться? Інтерпретатор PHP шукає змінну, в якій зберігається ідентифікатор сесії (за замовчуванням це PHPSESSID) спочатку в cookies, потім у змінних, переданих за допомогою POST- і GET-запитів. Якщо ідентифікатор знайдений, то користувач вважається ідентифікованим, виконується зміна всіх URL і виставляння cookies. У противному випадку користувач вважається новим, для нього генерується новий унікальний ідентифікатор, потім виконується зміна URL і виставляння cookies.

Команду session_start() потрібно викликати у всіх скриптах, у яких має бути використані змінні сесії, причому до виведення яких-небудь даних у браузер. Це пов'язано з тим, що cookies виставляються тільки до виведення інформації на екран.

Одержати ідентифікатор поточної сесії можна за допомогою функції session_id().

Для наочності сесії можна задати ім'я за допомогою функції session_name([ім'я_сесії]). Робити це потрібно ще до ініціалізації сесії. Одержати ім'я поточної сесії можна за допомогою цієї ж функції, викликаної без параметрів: session_name();

Приклад. Створення сесії

Переіменуємо наш файл index.html, щоб оброблялися php-скрипти, наприклад у Index.php, створимо сесію і подивимося, який вона одержить ідентифікатор і ім'я.

```
<? session_start();
    // створюємо нову сесію або відновлюємо поточну
echo session_id();
 // виводимо ідентифікатор сесії
?>
<html>
  <head><title>My home page</title></head>
... // домашня сторінка
</html>
<?
  echo session_name();
    // виводимо ім'я поточної сесії.
    // У даному випадку це PHPSESSID
?>
```

Якщо проробити те ж саме з файлом authorize.php, то значення виведених змінних (id сесії і її ім'я) будуть такими ж, якщо перейти на нього з index.php і не закривати перед цим вікно браузера (тоді ідентифікатор сесії зміниться).

4. Реєстрація змінних сесії.

Однак від самих ідентифікатора й імені сесії нам користі для розв’язання наших задач небагато. Ми ж хочемо передавати і зберігати протягом сесії наші власні змінні (наприклад, логін і пароль). Для того, щоб цього домогтися, потрібно просто зареєструвати свої змінні:

```
session_register(ім'я_змінної1, ім'я_змінної2, ...);
```

Помітимо, що реєструються не значення, а імена змінних. Зареєструвати змінну досить один раз на будь-якій сторінці, де використовуються сесії. Імена змінних передаються функції session_register() без знака $. Усі зареєстровані в такий спосіб змінні стають глобальними (тобто доступними з будь-якої сторінки) протягом даної сесії роботи із сайтом.

Зареєструвати змінну також можна, просто записавши її значення в асоціативний масив $_SESSION, тобто написавши

```
$_SESSION['ім'я_змінної'] = 'значення_зміної';
```

У цьому масиві зберігаються всі зареєстровані (тобто глобальні) змінні сесії.

Доступ до таких змінних здійснюється за допомогою масиву $_SESSION['ім'я_змінної'] (або $HTTP_SESSION_VARS['ім'я_змінної'] для версії PHP 4.0.6 і більш ранніх). Якщо ж у налаштуваннях php вімкнена опція **register_globals**, то до сесійних змінних можна звертатися ще і як до звичайних змінних, наприклад так: $ім'я_змінної.

Якщо **register_globals=off** (відключені), то користуватися session_register() для реєстрації змінних переданих методами POST або GET, не можна, тобто це просто не працює. І взагалі, не рекомендується одночасно використовувати обидва методи реєстрації змінних, $_SESSION і session_register().

Приклад. Реєстрація змінних
Зареєструємо логін і пароль, що вводяться користувачем на сторінці авторизації.

```
<?
session_start();
    // створюємо нову сесію або відновлюємо поточну
if (!isset($_GET['go'])){
  echo "<form>
    Login: <input type=text name=login>
    Password: <input type=password name=passwd>
    <input type=submit name=go value=Go>
  </form>";}
else {   $_SESSION['login']=$_GET['login'];
   // реєструємо змінну login
   $_SESSION['passwd']=$_GET['passwd'];
   // реєструємо змінну passwd
// тепер логін і пароль - глобальні змінні для цієї сесії
    if ($_GET['login']=="pit" && $_GET['passwd']=="123") {
        Header("Location: secret_info.php");
        // перенаправляємо на сторінку secret_info.php}
else echo "Невірне введення,спробуйте ще раз<br>";
}
print_r($_SESSION);
    // виводимо всі змінні сесії
?>
```

Тепер, потрапивши на сторінку secret_info.php, та й на будь-яку іншу сторінку сайта, ми зможемо працювати з введеними користувачем логіном і паролем, що будуть зберігатися в масиві $_SESSION. Таким чином, якщо змінити код секретної сторінки (помітьте, ми перейменували її в secret_info.php) так:

```
<?php
session_start();
    // створюємо нову сесію або відновлюємо поточну
print_r($_SESSION);
    // виводимо всі змінні сесії
?>
<html>
<head><title>Secret info</title></head>
<body>
<p>Тут я хочу поділитися секретами з другом Петром.
</body>
</html>
```

То ми отримаємо в браузері на секретній сторінці наступне:

```
Array ( [login] => pit [passwd] => 123 )
Тут я хочу поділитися секретами з другом Петром.
```

У підсумку отримаємо список змінних, зареєстрованих на authorize.php і, власне, саму секретну сторінку.

Що це нам дає? Припустимо, хакер хоче прочитати секрети Васі і Петра. І він якось довідався, як називається секретна сторінка (або сторінки). Тоді він може спробувати просто ввести її адресу в рядку браузера, минаючи сторінку авторизації (введення пароля). Щоб уникнути такого проникнення в наші таємниці, потрібно дописати усього пару рядків у код секретних сторінок:

```
<?php
session_start();
    // створюємо нову сесію або відновлюємо поточну
print_r($_SESSION);
    // виводимо всі змінні сесії
if (!($_SESSION['login']=="pit" &&  $_SESSION['passwd']==123))
    // перевіряємо правильність пароля-логіна
    Header("Location: authorize.php");
    // якщо помилка, то перенаправляємо на сторінку авторизації
?>
<html>
<head><title>Secret info</title></head>
... // тут розташовується секретна інформація :)
</html>
```

5. Знищення змінних сесії.

Крім вміння реєструвати змінні сесії (тобто робити їх глобальними протягом усього сеансу роботи), корисно також вміти знищувати такі змінні і сесію в цілому.

Функція session_unregister(ім'я_змінної) знищує глобальну перемінну з поточної сесії (тобто знищує її зі списку зареєстрованих змінних). Якщо реєстрація виконується за допомогою $_SESSION ($HTTP_SESSION_VARS для версії PHP 4.0.6 і більш ранніх), то використовують мовну конструкцію unset(). Вона не повертає ніякого значення, а просто знищує зазначені змінні.

Де це може знадобитися? Наприклад, для знищення даних про відвідувача (зокрема, логіна і пароля) після його виходу із секретної сторінки. Якщо правильні логін і пароль зберігаються і вікно браузера після відвідування сайта не закрили, то будь-який інший користувач цього комп'ютера зможе прочитати закриту інформацію.

Приклад. Знищення змінних сесії
У файл secret_info.php додамо рядок для виходу на головну сторінку:

```
<?php
// ... php код
?>
<html>
<head><title>Secret info</title></head>
... // тут розташовується секретна інформація :)
<a href="index.php">На головну</a>
</html>
```

У Index.php знищимо логін і пароль, введені раніше:

```
<?
session_start();
session_unregister('passwd');
    // знищуємо пароль
```

```
unset($_SESSION['login']);
    // знищуємо логин
print_r($_SESSION);
    // виводимо глобальні змінні сесії
?>
<html>
<head><title>My home page</title></head>
... // домашня сторінка
</html>
```

Тепер, щоб потрапити на секретну сторінку, потрібно буде знову вводити логін і пароль.

Для того, щоб скинути значення всіх змінних сесії, можна використовувати функцію `session_unset();`
Знищити поточну сесію цілком можна командою `session_destroy();` Вона не скидає значення глобальних змінних сесії і не видаляє cookies, а знищує всі дані, асоційовані з поточною сесією.

```
<?
session_start(); // ініціалізуємо сесію
$test = "змінна сесії";
$_SESSION['test']= $test;
// реєструємо змінну $test.
// якщо register_globals=on, то можна використовувати
// session_register('test');
print_r($_SESSION);
// виводимо всі глобальні змінні
echo session_id();
// виводимо ідентифікатор сесії
echo "<hr>";
session_unset();
// знищуємо всі глобальні змінні сесії
print_r($_SESSION);
echo session_id();
echo "<hr>";
session_destroy(); // знищуємо сесію
print_r($_SESSION);
echo session_id();
?>
```

У результаті роботи цього скрипта будуть виведені три рядки: у першому - масив з елементом test і його значенням, а також ідентифікатор сесії, у другому - порожній масив і ідентифікатор сесії, у третьому - порожній масив. Таким чином, видно, що після знищення сесії знищується і її ідентифікатор, і ми більше не можемо ні реєструвати змінні, ні взагалі робити які-небудь дії із сесією.

Безпека

Взагалі кажучи, варто розуміти, що використання механізму сесій не гарантує повної безпеки системи. Для цього потрібно вживати додаткових заходів. Звернемо увагу на проблеми з безпекою, що можуть виникнути при роботі із сесіями і, зокрема, з тими програмами, що ми написали.

По-перше, небезпечно передавати сюди пароль, його можуть перехопити. Крім того, ми зареєстрували його як глобальну змінну сесії, значить, він зберігся в cookies на комп'ютері-клієнті. Це теж погано. І взагалі, паролі і логіни краще повинні зберігатися в базі даних. Нехай інформація про користувачів зберігається в базі даних "test" (у таблиці "users"), а ми маємо до неї доступ під логіном my_user і паролем my_passwd.

По-друге, що робити, якщо хтось написав скрипт підбору пароля для секретної сторінки? У цьому випадку на сторінку авторизації багато разів повинен стукатися якийсь сторонній скрипт. Тому потрібно просто перевіряти, чи з нашого сайту прийшов запит на авторизацію, і якщо ні, то не пускати його далі. Адреса сторінки, з якої надійшов запит, можна одержати за допомогою глобальної змінної $_SERVER['HTTP_REFERER']). Хоча, звичайно, якщо за ламання сайта взялися серйозно, то значення цієї змінної теж підмінять (наприклад, за допомогою того ж PHP). Проте перевірку її значення можна вважати одним з найважливіших кроків на шляху до забезпечення безпеки свого сайта.

Начебто б перші дві проблеми розв'язані. Але є ще одна. Що робити, якщо хакер просто допише в рядок запиту значення якої-небудь глобальної змінної (наприклад, логіна)? Взагалі це можливо, тільки якщо register_globals=On. Просто інакше ми використовуємо для роботи з глобальними змінними масив $_SESSION і з ним такі фокуси не проходять. Усе-таки спробуємо розв'язати і цю проблему. Для цього потрібно очистити рядок запиту перед тим, як порівнювати значення параметрів. Тобто спочатку скинемо значення $user_login. Потім дану змінну потрібно знову зареєструвати, але не як нову, а як вже існуючу. Для цього знак долара при реєстрації НЕ опускається. От що вийшло:

```
<?php
unset($user_login); // знищуємо змінну
session_start(); // створюємо нову сесію або відновлюємо поточну
session_register($user_login);
    // реєструємо змінну як вже існуючу
if (!($user_login=="pit")) // перевіряємо логін
    Header("Location: authorize.php");
    // якщо помилка, то перенаправляємо на сторінку авторизації
?>
<html>
<head><title>Secret info</title></head>
... // тут розташовується секретна інформація :)
</html>
```

Висновок

Отже, ми познайомилися із сесіями й основними способами роботи з ними, проблемами, що виникають при їхньому використанні, і можливими розв'язками цих проблем. Сподіваюся, що після прочитання лекції читачам стало зрозуміло, наскільки зручні і прості у використанні сесії, а наведені приклади знадобляться на практиці.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

1. Розробити сценарій, що використовує механізм сесій і задовільняє нижче переліченим вимогам.
   Створити три сторінки:
   - основна - сторінка авторизації, що містить посилання на першу і другу сторінку.
     (якщо авторизацію пройшла успішно, то здійснюється переадресація на першу сторінку)
   - перша - доступна лише для зареєстрованих користувачів, тобто якщо користувач авторизований, то він має можливість перглядати вмістиме цієї сторінки, якщо ні то здійснюватиметься переадресація на сторінку авторизації
   - друга - частково доступна для не авторизованих користувачів і повністю доступна для авторизованих

   На першій і другій сторінках міститься посилання, з допомогою якого сесія буде закрита

Вказівка: використати завдання №2 попередньої лабораторної роботи, де виконується авторизація користувачів відомості про які містяться відомості в файлі .

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що собою являє авторизація доступу.
2. Який механізм сесії
3. Яим чином відбувається реєстрація змінних в сесії.
4. Як можна знищити змінні в сесії.

Лабораторна робота №12

Тема: Сценарії PHP для виконання основних операцій з базами даних MySQL.
Мета: Набути навиків роботи із сервером СУБД MySQL через сценарії PHP. Удосконалити вміння створювати SQL-запити.

ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Взаємодія PHP і MySQL

Ця частина лекції ознайомить зі способами взаємодії PHP і СУБД MySql. Основна увага буде приділена встановленню з'єднання з базою даних, функціям відправлення запитів і обробці відповідей (mysql_connect, mysql_query, mysql_result, mysql_num_rows, mysql_close). Приклад - створення web -інтерфейсу для адміністрування бази даних віртуального музею історії.

У дистрибутив PHP входить розширення, що містить вбудовані функції для роботи з базою даних MySQL. У цій частині лекції ми познайомимося з деякими основними функціями для

роботи з MySQL, що будуть потрібні для розв’язання задач побудови web-інтерфейсів з метою відображення і наповнення бази даних. Виникає питання, навіщо будувати такі інтерфейси? Для того щоб вносити інформацію в базу даних і переглядати її вміст могли люди, не знайомі з мовою запитів SQL. При роботі з web-інтерфейсом для додавання інформації в базу даних людині потрібно просто ввести ці дані в html-форму і відправити їх на сервер, а наш скрипт зробить все інше. А для перегляду вмісту таблиць досить просто клацнути по посиланню і зайти на потрібну сторінку.

Для наочності будемо будувати ці інтерфейси для таблиці Artifacts, у якій міститься інформація про експонати віртуального музею інформатики. У попередній частині лекції ми вже наводили структуру цієї колекції, а також її зв'язку з колекціями опису персон (Persons) і зображень (Images). Нагадаємо, що кожен експонат у колекції Artifacts описується за допомогою наступних характеристик:

- назва (title);
- автор (author);
- опис (description);
- альтернативна назва (alternative);
- зображення (photo).

Назва й альтернативна назва є рядками менш ніж 255 символів довжиною (тобто мають тип VARCHAR(255)), опис - текстове поле (має тип TEXT), а в полях "автор" і "зображення" містяться ідентифікатори автора з колекції Persons і зображення експоната з колекції Images відповідно.

Отже, у нас є якась таблиця в базі даних. Щоб побудувати інтерфейс для додавання інформації в цю таблицю, потрібно її структуру (тобто набір її полів) відобразити в html-форму.
Розіб'ємо цю задачу на наступні підзадачі:

- установка з'єднання з БД;
- вибір робочої БД;
- одержання списку полів таблиці;
- відображення полів у html-форму.

Після цього дані, введені у форму, потрібно записати в базу даних. Розглянемо всі ці задачі одну за одною.

1. Встановлення з’єднання із БД.

Отже, перше, що потрібно зробити, - це встановити з'єднання з базою даних. Скористаємося функцією mysql_connect.

Синтаксис mysql_connect

```
 mysql_connect ( [рядок server [, рядок username [, рядок password
                                    [,  логічне  new_link  [,  ціле
client_flags]]]]])
```

Дана функція встановлює з'єднання із сервером MySQL і повертає вказівник на це з'єднання або FALSE у випадку невдачі. Для відсутніх параметрів встановлюються наступні значення за замовчуванням:

```
server = 'localhost:3306'
username = ім'я користувача власника процесу сервера
password = порожній пароль
```

Якщо функція викликається двічі з тими самими параметрами, то нове з'єднання не встановлюється, а повертається посилання на старе з'єднання. Щоб цього уникнути, використовують параметр new_link, що змушує в будь-якому випадку відкрити ще одне з'єднання.

Параметр client_flags - це комбінація наступних констант: MYSQL_CLIENT_COMPRESS (використовувати протокол стиску), MYSQL_CLIENT_IGNORE_SPACE (дозволяє вставляти пробіли після імен функцій), MYSQL_CLIENT_INTERACTIVE (чекати interactive_timeout секунд - замість wait_timeout - до закриття з'єднання).

Параметр new_link з'явився в PHP 4.2.0, а параметр client_flags - у PHP 4.3.0.

З'єднання із сервером закривається при завершенні виконання скрипта, якщо воно до цього не було закрито за допомогою функції mysql_close().

Отже, установлюємо з'єднання з базою даних на локальному сервері для користувача nina з паролем "123":

```
<?
$conn = mysql_connect("localhost", "nina","123")
or die("Неможливо установити з'єднання: ". mysql_error());
echo "З'єднання встановлене";
mysql_close($conn);
?>
```

Дія mysql_connect рівносильна команді `shell>mysql -u nina -p123`

2. Вибір бази даних.

Після встановлення з'єднання потрібно вибрати базу даних, з якою будемо працювати. Наші дані зберігаються в базі даних book. У MySQL вибір бази даних здійснюється за допомогою команди use:

```
mysql>use book;
```

У PHP для цього існує функція `mysql_select_db`.

Синтаксис mysql_select_db:

```
логічне mysql_select_db (рядок database_name [, ресурс
link_identifier])
```

Ця функція повертає TRUE у випадку успішного вибору бази даних і FALSE - у противному випадку.

Зробимо базу даних book робочою:

```
<?
$conn = mysql_connect("localhost","nina","123")
or die("Неможливо установити з'єднання: ". mysql_error());
echo "З'єднання встановлене";
mysql_select_db("book");
?>
```

3. Одержання списку полів таблиці.

Тепер можна зайнятися власне розв’язанням задачі. Як одержати список полів таблиці? Дуже просто. У PHP і на цей випадок є своя команда - mysql_list_fields.

Синтаксис mysql_list_fields

```
ресурс mysql_list_fields (рядок database_name,рядок table_name[,
ресурс link_identifier])
```

Ця функція повертає список полів у таблиці table_name у базі даних database_name. Отже, вибирати базу данних нам було необов'язково, але це знадобиться пізніше. Як можна помітити, результат роботи цієї функції - змінна типу ресурс. Тобто це не зовсім те, що ми хотіли отримати. Це посилання, яке можна використовувати для одержання інформації про поля таблиці, включаючи їх назви, типи і прапори.

Функція mysql_field_name повертає ім'я поля, отриманого в результаті виконання запиту. Функція mysql_field_len повертає довжину поля. Функція mysql_field_type повертає тип поля, а функція mysql_field_flags повертає список прапорів поля, записаних через пробіл. Типи полів можуть бути int, real, string, blob і т.д. Прапори можуть бути not_null, primary_key, unique_key, blob, auto_increment і т.д.

Синтаксис у всіх цих команд однаковий:

```
рядок mysql_field_name (ресурс result, ціле field_offset)
рядок mysql_field_type (ресурс result, ціле field_offset)
рядок mysql_field_flags (ресурс result, ціле field_offset)
рядок mysql_field_len (ресурс result, ціле field_offset)
```

Тут result - це ідентифікатор результату запиту (наприклад, запиту, відправленого функціями mysql_list_fields або mysql_query (про неї буде розказано пізніше)), а field_offset - порядковий номер поля в результаті.

Взагалі кажучи, те, що повертають функції типу mysql_list_fields або mysql_query, являє собою таблицю, а точніше, вказівник на неї. Щоб одержати з цієї таблиці конкретні значення, потрібно задіяти спеціальні функції, що пострічково читають цю таблицю. До таких функцій і відносяться mysql_field_name і т.п. Щоб перебрати всі рядки в таблиці результату виконання запиту, потрібно знати кількість рядків у цій таблиці. Команда mysql_num_rows(ресурс result) повертає кількість рядків у безлічі результатів result.

А тепер спробуємо одержати список полів таблиці Artifacts (колекція експонатів).

```
<?
$conn = mysql_connect("localhost","nina","123")
or die("Неможливо установити з'єднання: ". mysql_error());
echo "З'єднання встановлене";
mysql_select_db("book");
$list_f = mysql_list_fields ("book","Artifacts",$conn);
$n = mysql_num_fields($list_f);
for($i=0;$i<$n; $i++){
  $type = mysql_field_type($list_f, $i);
  $name_f = mysql_field_name($list_f,$i);
  $len = mysql_field_len($list_f, $i);
  $flags_str =  mysql_field_flags ($list_f, $i);
echo "<br>Ім'я поля: ". $name_f;
echo "<br>Тип поля: ". $type;
echo "<br>Довжина поля: ". $len;
echo "<br>Рядок прапорів поля: ". $flags_str . "<hr>";
}
?>
```

У результаті повинно вийти приблизно от що (якщо в таблиці всього два поля, звичайно):

```
Ім'я поля: id
Тип поля: int
Довжина поля: 11
Рядок прапорів поля: not_null primary_key auto_increment
Ім'я поля: title
Тип поля: string
Довжина поля: 255
Рядок прапорів поля:
```

Відображення списку полів у html-формі

Тепер дещо підкоректуємо попередній приклад. Будемо не просто виводити інформацію про поле, а відображати його як потрібний елемент html-форми. Так, елементи типу BLOB переведемо в textarea (помітимо, що поле description, що ми створювали з типом TEXT, відображається як таке, що має тип BLOB), числа і рядки відобразимо як текстові рядки введення <input type=text>, а елемент, що має мітку автоінкремента, взагалі не будемо відображати, оскільки його значення встановлюється автоматично.
Усе це розв'язується досить просто, за винятком виділення зі списку прапорів прапора auto_increment. Для цього потрібно скористатися функцією explode.

Синтаксис explode:

```
масив explode(рядок separator, рядок string [, int limit])
```

Ця функція розбиває рядок string на частини за допомогою роздільника separator і повертає масив отриманих рядків.

У нашому випадку як роздільник потрібно взяти пробіл " ", а як задану стрічку для розбивки - рядок прапорів поля.

Отже, створимо форму для введення даних у таблицю Artifacts:

```
<?
$conn=mysql_connect("localhost","nina","123");      //     встановлюємо
з'єднання
$database = "book";
$table_name = "Artifacts";
mysql_select_db($database); // обираємо базу даних для роботи
$list_f   =   mysql_list_fields($database,$table_name); //   отримуємо
список полів в базі
$n = mysql_num_fields($list_f);
                        // кількість стрічок в результаті попереднього
запиту
                        //тобто   скільки   всього   полів   в   таблиці
Artifacts)
echo "<form method=post action=insert.php>";
                            // створюємо форму для введення даних
echo "<TABLE BORDER=0 CELLSPACING=0 width=50%><tr>
    <TD  BGCOLOR='#005533' align=center>
     <font color='#FFFFFF'>
        <b> Add new row in $table_name</b>
    </font></td></tr>
    <tr><td></td></tr></TABLE>";
echo "<table border=0 CELLSPACING=1 cellpadding=0 width=50% >";
// для кожного поля отримуємо його ім'я, тип, довжину і прапори
for($i=0;$i<$n; $i++){
    $type = mysql_field_type($list_f, $i);
```

```
    $name_f = mysql_field_name ($list_f,$i);
    $len = mysql_field_len($list_f, $i);
    $flags_str = mysql_field_flags ($list_f, $i);
// із стрічки прапорів робимо масив, де кожен елемент масиву – прапор
поля,
    $flags = explode(" ", $flags_str);
    foreach ($flags as $f){
        if ($f == 'auto_increment') $key = $name_f;
            // запам'ятовуємо ім'я інкремента
    }
/* для кожного поля, що не автоінкрементом, в
залежності від його типу виводимо потрібний елемент форми */
if ($key <> $name_f){
echo "<tr><td align=right bgcolor='#C2E3B6'>
      <font size=2><b> ". $name_f ."</b></font>
      </td>";
switch ($type){
        case "string":
             $w = $len/5;
              echo  "<td><input  type=text  name=\"$name_f\"  size  =  $w
></td>";
        break;
        case "int":
            $w =  $len/4;
             echo  "<td><input  type=text  name=\"$name_f\"  size  =  $w
></td>";
        break;
        case "blob":
                echo        "<td><textarea        rows=6        cols=60
name=\"$name_f\"></textarea></td>";
        break;
        }
    }
    echo "</tr>";
echo "</table>";
echo "<input type=submit name='add' value='Add'>";
echo "</form>";
?>
```

4. Запис даних у базу даних.

Отже, форма створена. Тепер потрібно зробити саме головне - відправити дані з цієї форми в нашу базу даних. Як ви вже знаєте, для того щоб записати дані в таблицю, використовується команда INSERT мови SQL. Наприклад:

```
mysql> INSERT INTO Artifacts SET title='Петров';
```

Виникає питання, як можна скористатися такою командою (або будь-якою іншою командою SQL) у PHP скрипті. Для цього існує функція mysql_query().

Синтаксис mysql_query

```
ресурс mysql_query ( рядок query [, ресурс link_identifier])
```

mysql_query() посилає SQL-запит активній базі даних MySQL сервера, що визначається за допомогою вказівника link_identifier (це посилання на якесь з'єднання із сервером MySQL). Якщо параметр link_identifier опущений, використовується останнє відкрите з'єднання. Якщо відкриті з'єднання відсутні, функція намагається з'єднатися із СУБД, аналогічно функції mysql_connect() без параметрів. Результат запиту буферизується.

Зауваження: рядок запиту НЕ повинен закінчуватися крапкою з комою.

Тільки для запитів SELECT, SHOW, EXPLAIN, DESCRIBE, mysql_query() повертає вказівник на результат запиту, або FALSE, якщо запит не був виконаний. В інших випадках mysql_query() повертає TRUE, якщо запит виконаний успішно, і FALSE - у випадку помилки. Значення, не рівне FALSE, говорить про те, що запит був виконаний успішно. Воно не говорить про кількість порушених або повернутих рядів. Цілком можлива ситуація, коли успішний запит не торкнеться жодного ряду. mysql_query() також вважається помилковим і поверне FALSE, якщо в користувача недостатньо прав для роботи з зазначеною в запиті таблицею.

Отже, тепер ми знаємо, як відправити запит на вставку рядків у базу даних. Помітимо, що в попередньому прикладі елементи форми ми назвали іменами полів таблиці. Тому вони будуть доступні в скрипті insert.php, що обробляє дані форми, як змінні виду

```
$_POST['ім'я_поля'].
<?
$conn=mysql_connect("localhost","nina","123");//встановлюємо з'єднання
$database = "book";
$table_name = "Artifacts";
mysql_select_db($database); // обираємо базу даних
$list_f = mysql_list_fields($database,$table_name);
              // отримуємо список полів в базі
$n = mysql_num_fields($list_f);
                // кількість стрічок в результаті попереднього запиту
                // створимо один запит відразу для всіх полів таблиці
$sql = "INSERT INTO $table_name SET ";
     // починаємо створювати запит, перебираємо всі поля таблиці
for($i=0;$i<$n; $i++){
    $name_f = mysql_field_name ($list_f,$i); // визначаємо ім'я поля
    $value = $_POST[$name_f]; // обчислюємо значення поля
    $j = $i + 1;
    $sql = $sql . $name_f." = '$value'";
                      // дописуємо в стрічку $sql пару ім'я=значення
    if ($j <> $n) $sql = $sql . ", ";
                     // якщо поле не останнєв списку, то ставимо кому
}
// перед тим, як записувати що небуть у базу можна подивитися,
//який запит отримаєтьмя
//echo $sql;
$result = mysql_query($sql,$conn); // відправляємо запит
// виводимо повідомлення чи успішно виконано запит
if (!result) echo " Can't add ($table_name) ";
    else echo "Success!<br>";
?>
```

Отже, задачу додавання даних за допомогою web-інтерфейсу ми розв'язали. Однак тут є одна тонкість. При розв'язанні ми не враховували той факт, що значення деяких полів (author, photo) повинні братися з інших таблиць (Persons, Images). Оскільки MySQL із зовнішніми ключами не працює, цей момент залишається на совісті розроблювачів системи. Потрібно дописати програму таким чином, щоб була можливість вводити в такі поля правильні значення. Але ми робити цього не будемо, оскільки завдання лекції полягає в тому, щоб познайомити читача з елементами технології, а не в тому, щоб створити працюючу систему. Тепер звернемося до іншої задачі - відображення даних, що зберігаються в базі даних СУБД MySQL.

5. Відображення даних, що зберігаються в MySQL.

Щоб відобразити якісь дані в браузер за допомогою PHP, потрібно спочатку одержати ці дані у вигляді змінних PHP. При роботі з MySQL без посередника (такого, як PHP) вибірка даних робиться за допомогою команди SELECT мови SQL:

```
mysql> SELECT * FROM Artifacts;
```

У попередньому розділі ми говорили, що будь-який запит, у тому числі і на вибірку, можна відправити на сервер за допомогою функції mysql_query(); Там у нас була дещо інша задача - одержати дані з форми і відправити їх за допомогою запиту на вставку в базу даних. Результатом роботи mysql_query() там міг бути лише один з виразів, TRUE або FALSE. Тепер же потрібно відправити запит на вибірку всіх полів, а результат відобразити в браузері. І тут результат - це ціла таблиця значень, а точніше, вказівник на цю таблицю. Так що потрібні якісь аналоги функції mysql_field_name(), тільки щоб вони витягували з результату запиту не ім'я, а значення поля. Таких функцій у PHP декілька. Найбільш популярні - mysql_result() і mysql_fetch_array().

Синтаксис mysql_result

```
змішане mysql_result (ресурс result, ціле row [, змішане field])
```

mysql_result() повертає значення однієї комірки результату запиту. Аргумент field може бути порядковим номером поля в результаті, ім'ям поля або ім'ям поля з ім'ям таблиці через крапку tablename.fieldname. Якщо для імені поля в запиті застосовувався аліас ('select foo as bar from...'), використовуйте його замість реального імені поля.

Працюючи з великими результатами запитів, варто задіяти одну з функцій, що обробляє відразу цілий ряд результатів (наприклад, mysql_fetch_row(), mysql_fetch_array() і т.д.). Тому що ці функції повертають значення декількох комірок відразу, вони НАБАГАТО швидші mysql_result(). Крім того, потрібно врахувати, що вказівка чисельного зсуву (номера поля) працює набагато швидше, ніж вказівка стовпчика або стовпчиків і таблиці через крапку.

Виклики функції mysql_result() не повинні змішуватися з іншими функціями, що працюють з результатом запиту.

Синтаксис mysql_fetch_array

```
масив mysql_fetch_array (ресурс result [, ціле result_type])
```

Ця функція обробляє ряд результатів запиту, повертаючи масив (асоціативний, чисельний або обидва) з обробленим рядом результатів запиту, або FALSE, якщо рядів більше немає.

mysql_fetch_array() - це розширена версія функції mysql_fetch_row(). Крім збереження значень у масиві з чисельними індексами, функція повертає значення в масиві з індексами за назвою стовпчиків.

Якщо декілька колонок у результаті будуть мати однакові назви, буде повернутий останній стовпчик. Щоб одержати доступ до перших, варто використовувати чисельні індекси масиву або аліаси в запиті. У випадку аліасів саме їх ви не зможете використовувати дійсні імена стовпчиків, як, наприклад, не зможете використовувати "photo" в описаному нижче прикладі.

```
select Artifacts.photo as art_image, Persons.photo as pers_image from
Artifacts, Persons
```

Важливо зауважити, що mysql_fetch_array() працює НЕ повільніше, ніж mysql_fetch_row(), і надає більш зручний доступ до даних.

Другий опційний аргумент result_type у функції mysql_fetch_array() є константою і може приймати наступні значення: MYSQL_ASSOC, MYSQL_NUM і MYSQL_BOTH. Ця можливість додана в PHP 3.0.7. Значенням за замовчуванням є: MYSQL_BOTH.

Використовуючи MYSQL_BOTH, одержимо масив, що складається як з асоціативних індексів, так і з чисельних. MYSQL_ASSOC поверне тільки асоціативні відповідності, а MYSQL_NUM - тільки чисельні.

Зауваження: імена полів, що повертаються цією функцією, регістронезалежні.

У цій лекції ми розглянули дві задачі: додавання даних у базу даних і їхнє відображення в браузері за допомогою мови PHP. Для цього ми розглянули ряд функцій, що дозволяють відправляти SQL-запити до бази даних і обробляти отримані відповіді. Використовуючи наведену тут технологію, можна розв'язати цілий ряд схожих задач, таких як задачі зміни і знищення даних, задачі маніпулювання таблицями бази даних (тобто їхнє створення, зміна і знищення) і т.п. Усе це типові задачі, що виникають при розробці систем керування даними, і вміння їх розв'язувати, як і вміння працювати з базами даних у цілому, дуже важливі для web-програміста.

MySQL – це програма-сервер, що працює на комп'ютері. Клієнтські програми (наприклад, сценарії) відправляють їй спеціальні запити за допомогою мережевих засобів, сервер їх опрацьовує за спеціальними запитом клієнта результат або його частина передає клієнту.

Перш ніж працювати з базою даних, необхідно встановити з нею мережеве з'єднання, а також провести авторизацію користувача. Для цього використовується функція mysql_connect(). **int mysql_connect([сервер] [користувач] [пароль]).** З'єднання з MySQL-сервером буде автоматично закрите після закінчення роботи сценарію, або ж при виклику функції mysql_close().

Для формування запитів до бази даних застосовується функція mysql_query, яка повертає ідентифікатор результуючого набору даних. Результат відразу не пересилається клієнту? Так от, щоб працювати з ним надалі, і використовується цей ідентифікатор. Існує дуже багато функцій, які приймають його як параметр і повертають ті або інші дані.

Синтаксис функції такий: **int mysql_query(string $query [,int $link_identifier])** Параметр $query є запитом-стрічкою, який повинен бути описаний за правилами мови SQL. Необов'язковий параметр $link_identifier використовується для відновлення втраченого зв'язку із сервером MySQL.

Розглянемо детальніше основні конструкції для формування SQL-запитів:

1. **insert into table_name (назва_таблиці1 назва_таблиці2 ...) values('значення1', 'значення2'...)** – запит додає до таблиці назва_таблиці запис, поля якого значення1, набувають значень val1
2. **delete from назва_таблиці where вираз** – Знищує із таблиці записи, для яких є істинним вираз. Параметр вираз є логічним виразом. Наприклад:(id<10) and (age=25)
3. **select * from where вираз [order by назва_поля [desc]]** – проводить пошук записів, які відповідають заданому виразу. Якщо отриманих записів декілька, то при заданій конструкції **order by** вони будуть відсортовані за тим полем **назва_поля** (якщо задано слово **desc**, то сортування буде проведено у зворотному порядку). Символ * вказує на те що слід отримати дані усіх полів, записи, яких відповідають умові. Тобто замість стмволу "*" можна вказувати потрібні поля.

4. **update назва_таблиці SET (назва_поля1='значення1', назва_поля2='значення2', ...) where вираз** – У таблиці для записів, які відповідають умові зазначенні полянабувають відповідних значень

Після виконання запиту запит виконаний і отриманий його ідентифікатор потрібно отримати власне дані – результат. Для цього найбільш часто використовується функція **array mysql_fetch_row(int $result)**, яка повертає масив-список із значеннями полів чергового рядка результату $result. Якщо вказівник поточної позиції результату був встановлений після останнього запису), то функція повертає значення false. Поточна позиція переміщується до наступного запису, так що черговий виклик mysql_fetch_row() поверне наступний рядок результату.

Функція mysql_fetch_array() повертає черговий рядок результату у вигляді асоціативного масиву, де кожному полю відповідає елемент з ключем, що співпадає з ім'ям поля.

Приклад:із таблиці 10А бази даних pupils вивести прізвища та імена учнів (поля surname та name) у алфавітному порядку, середній бал яких більший за 4,5 (поле бал).

```
<?php
   $db=mysql_connect(localhost,root,'');
   mysql_select_db(pupils, $db);
   $sql="SELECT surname,name FROM 10A WHERE bal>4,5 order by
surname";
   $result=mysql_query($sql, $db);
   while ($row=mysql_fetch_array($result1)){
   echo $row[surname];
   echo $row[rname].”\n”;
   }
   mysql_close_db();
?>
```

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

1. Створити таблиці з наступними полями:
   <перші 3 літери вашого прізвища>_user
   id - унікальний ідентифікатор
   login – прізвисько;
   password - пароль;
   name - ім'я;
   lastname – прізвище;

   <перші 3 літери вашого прізвища>_forum
   id - унікальний ідентифікатор
   tema – тема для обговорення;
   autor - автор теми;
   data - дата і час створення теми;

   <перші 3 літери вашого прізвища>_comment
   id - унікальний ідентифікатор
   id_tema – тема до якоъ наледить коментар;
   commentar - текст повыдомлення;
   autor - автор коментаря;
   data - дата і час створення коментаря;

   Типи полів вибрати самостійно

2. Розробити PHP сценарії (в окремій папці **admin_user**), для виводу вмісту таблиці **<...>_user** з можливістю додавання, редагування і знищення її записів.
3. Розробити PHP сценарій (**index.php**) авторизації користувачів, відомості про яких містяться в таблиці **<...>_user**. В разі успішоного проходження авторизації відбувається перехід на сторінку **forum.php.**
4. Розробити PHP сценарій сторінки **forum.php**, в якій виводяться всі записи таблиці **<...>_forum** у такому вигляді:

тема	автор	дата і час	кількість коментарів
назва теми, яка є гіперпосилкою на сторінку **comment.php**			

5. Форму для додавання запису в цю таблицю (форма повинна містити лише одне текстове поле, інша інформація про дату, автора, та унікальний ідентифікатор, повинні шукатись)
6. Розробити PHP сценарій сторінки **comment.php**, яка виводитиме лише ті записи таблиці **<...>_commenet**, які стосуються обраної теми, і на якій знаходиться форма для введення повідомлення на обрану тему (для додавання запису в таблицю **<...>-comment** з **id_tema** на обрану тему)

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Як відбувається з’єднання з базою даних.
2. Якою функцією можна одержати список полів таблиці.
3. Яка функція дозволяє здійснити запит даних у базу даних.
4. Як відобразит дані що зберігаються в MySQL.

Лабораторна робота №13

Тема: JavaScrip. Основні оператори мови.
Мета: Ознайомитися з мовою JavaScript.

ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Javascript. Основні оператори мови

Мова JavaScript була розроблена Бренданом Айком з корпорації Netscape Communications у 1995 році. JavaScript має більше спільного з Self, Lisp і ALGOL ніж з Java.

Комітет TC39 об'єднав програмістів Netscape, Sun, Microsoft, Borland, NOMBAS і інших компаній, які проявляють інтерес до майбутнього мов сценаріїв, і за кілька місяців розробив стандарт ECMA-262, який визначив нову мову сценаріїв з назвою ECMAScript.

Розробники браузерів зі змінним успіхом використовували ECMAScript як основу для реалізації своїх версій JavaScript.

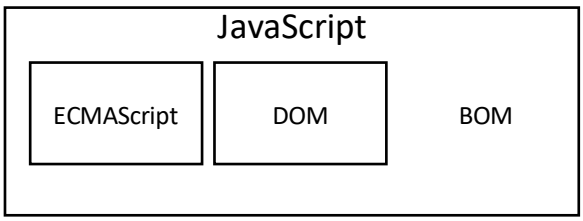
Будемо розглядати мову ECMAScript 6 (ES6) опублікований Ecma в червні 2015 року.

Web-документ, який відображається браузером, – це результат виконання програм, створених на різних мовах. Для опису структури і загального зовнішнього вигляду можна використовувати мову розмітки HTML, для опису особливостей зовнішнього вигляду, відмінного від стандартизованого – краще використовувати мову стилів CSS. Для опису поведінки документа, його реакції на дії користувача використовується мова сценаріїв JavaScript.

JavaScript код виконує браузер. Тобто він працює на стороні клієнта. У нього вбудований інтерпретатор JavaScript. Отже, виконання програми залежить від того, коли цей інтерпретатор отримує управління і який саме це браузер. Останні версії JavaScript можуть не підтримуватися деякими браузерами.

Повна реалізація JavaScript складається з трьох частин:

- ядро (ECMAScript);
- об'єктна модель документа (Document Object Model, DOM);
- об'єктна модель браузера (Browser Object Model, BOM).



На базовому рівні ECMAScript визначає наступні, незалежні від браузера, частини мови:

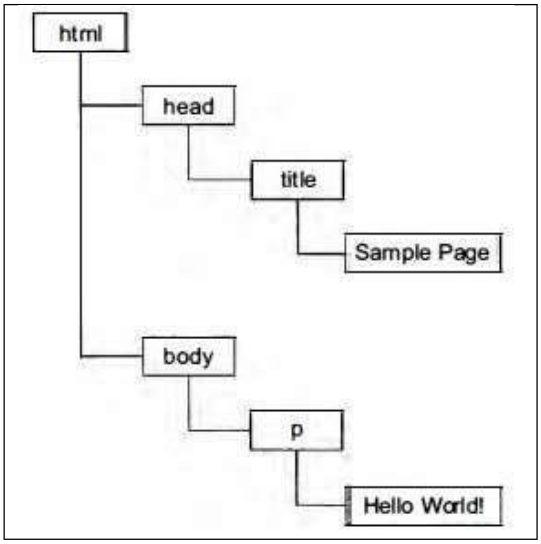
- синтаксис;
- типи;
- інструкції;
- ключові слова;
- зарезервовані слова;
- оператори;
- об'єкти.

Об'єктна модель документа

Об'єктна модель документа (DOM) – це прикладний програмний інтерфейс (Application Programming Interface, API) для XML, застосування якого було розширено на HTML. В DOM вся сторінка представляється як ієрархія вузлів. Кожен елемент HTML- або XML-сторінки є вузлом певного типу, що містить ті чи інші дані. Розглянемо наступну HTM-сторінку:

```
<html>
    <head>
      <title>Sample Page</title>
    </head>
    <body>
      <p>Hello World!</p>
    </body>
</html>
```

Цей код можна зобразити за допомогою DOM як ієрархію вузлів.



Використовуючи DOM API, можна з легкістю видаляти вузли, додавати, замінювати і змінювати їх. Завдяки реалізації динамічного HTML (Dynamic HTML, DHTML), в Internet Explorer 4 і Netscape Navigator 4 розробники вперше змогли змінювати вид і контент веб-сторінок без їх перезавантаження. Це стало важливим етапом розвитку веб-технологій, але виникла серйозна проблема. Netscape і Microsoft вибрали різні шляхи розвитку DHTML, що завершило період, коли можна було писати HTM-сторінки, не замислюючись про веб-браузери.

Для роботи з новими інтерфейсами в DOM Level 2 були представлені нові модулі:

- DOM Views – інтерфейси для відстеження різних представлень документа (наприклад, документ до і після застосування стилів CSS);
- DOM Events – інтерфейси для подій і обробки подій;
- DOM Style – інтерфейси для роботи зі стилізацією елементів за допомогою CSS;
- DOM Traversal and Range – інтерфейси для обходу елементів дерева документа і виконання операцій над ними.

DOM Level 3 доповнила DOM уніфікованими методами завантаження і збереження документів.

Об'єктна модель браузера

В Internet Explorer 3 і Netscape Navigator 3 була представлена об'єктна модель браузера (BOM), яка забезпечує доступ до вікна браузера і дозволяє маніпулювати його елементами. Використовуючи BOM, можна взаємодіяти з браузером поза контекстом відображуваної сторінки. До недавніх пір BOM була єдиною частиною реалізації JavaScript, що не має стандарту, через що при роботі з нею часто виникали проблеми. Формалізація багатьох елементів BOM в HTML5 змінила ситуацію на краще, прояснила багато неясних аспектів моделі.

BOM регламентує роботу з вікном і фреймами браузера, але будь-яке специфічне для браузера JavaScript розширення теж зазвичай вважається частиною BOM. Ось деякі такі розширення:

- функція відображення спливаючих вікон в браузері;
- можливість переміщати, закривати і змінювати розміри вікна браузера;
- об'єкт `navigator`, що надає докладні відомості про браузер;
- об'єкт `location`, що надає докладні відомості про сторінку, завантажену в браузері;
- об'єкт `screen`, що надає докладні відомості про дозвіл екрана;
- підтримка cookie-файлів;
- настроюються настроюються об'єкти, включаючи `XMLHttpRequest`, а також `ActiveXObject` в Internet Explorer.

Підключення JavaScript коду на сторінку

Виконує JavaScript код браузер. У нього вбудований інтерпретатор JavaScript. Отже, виконання програми залежить від того, коли цей інтерпретатор отримує управління. Наведемо кілька способів розміщення коду JavaScript:

1. В контейнері заголовка сторінки `<head>`

```
<head>
    …
    <script type="text/javascript">команди скрипта</script>
    …
</head>
```

2. В контейнері тіла сторінки <body>

```
<body>
    …
    <script>команди скрипта</script>
    …
</body>
```

3. Обробник події вказується безпосередньо в тому тегу, події якого потрібно обробляти, без укладення в теги `<script></script>`

```
<input type="button" value="Натиснути" onClick="window.alert('Натисніть ще раз
')">
```

4. У зовнішньому файлі. За аналогією з тим, як стилі підключаються до сторінки за допомогою елемента `link`, сценарії підключаються за допомогою елемента `script`, тільки файл має розширення не *css*, а *js*.

```
<head>
    …
    <script type="text/javascript" src="my.js"></script>
    …
</head>
```

Виконання операторів сценарію

Існує кілька способів визначення моменту запуску сценарію. Ось деякі з них:

- при завантаженні документа;
- відразу після завантаження документа;
- у відповідь на дії користувача.

Коди елементів `<script>` обробляються в порядку їх розташування на сторінці HTML якщо у цих тегів немає атрибутів `defer` і `async`. Дані атрибути дозволяють відкласти виконання сценарію зовнішнього файлу зі скриптом:

- до повного завантаження і візуалізації web-сторінки,
- паралельно із завантаженням даного скрипта виконувати наступні дії на сторінці.

Велика кількість сценаріїв, виконуваних при завантаженні, наприклад, в заголовку сторінки, може істотно сповільнити візуалізацію документа. Тому

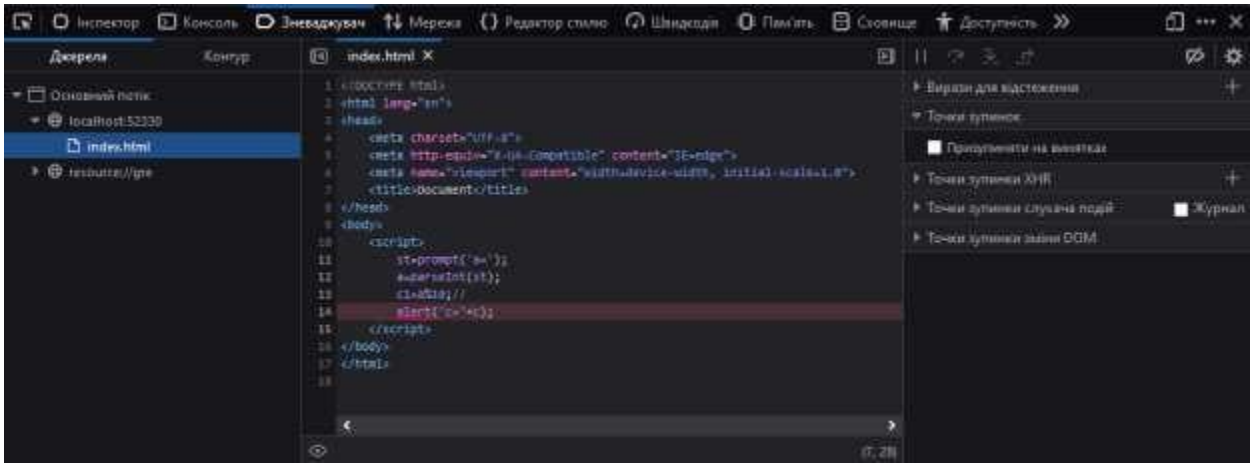
найчастіше ми будемо користуватися сценаріями, виконуваними у відповідь на дії користувача.

Налагодження скрипта

Як правило в браузерах за замовчуванням вже активізований дозвіл використовувати JavaScript. Якщо це не так, то це можна легко зробити в налаштуваннях будь-якого з них.

Помилки при використанні HTML, CSS браузером просто ігноруються. А виводяться лише коректна частина сторінки. Налагодження подібних документів тому ускладнюється. Для контролю синтаксису для HTML, CSS, JavaScript як і раніше можемо використовувати редактори текстів Notepad++. Налагодження JavaScript коду спрощується, тому що у всіх браузерах є спеціальна панель, яка називається «інструменти розробника» (засоби розробника). Цю панель можна запустити за допомогою натискання на клавіатурі сполучення клавіш [Ctrl]+[Shift]+[I] або клавіши [F12].

В Firefox ця панель виглядає наступним чином:



Номер рядка з помилкою, в наведеному прикладі, – 14. У операторі зроблено спробу вивести значення змінної «с», а в програмі значення зберігається в змінній «с1»!

У самому налагоджувачи ми можемо побачити посилання на помилки, але коригувати код потрібно все ж в редакторі.

Вступ в JavaScript

JavaScript – залежить від регістра. Імена JavaScript і Javascript – різні імена!

Всі ключові слова використовують тільки нижній регістр. Вимоги до імен змінних такі ж, як в C, Python.

Оператори розділяються крапкою з комою, яку можна опустити, якщо оператор закінчується символом нового рядка (Enter).

Коментарі:

```
// однорядковий коментар,

/*
багаторядковий
коментар
*/
```

Типи даних

Змінні оголошувати не обов'язково. Однак можна і оголошувати або оголошувати за допомогою оператора `var`. Якщо змінна оголошена в тілі функції, то вона є локальною змінною. Немає суворої типізації змінних.

У момент використання значення змінної вона повинна мати тип і значення. Можливо оголошення с одночасною ініціалізацією.

Приклади:

Оголошується цілочисельна змінна x, що має десяткове значення 123:

```
var s = 123
```

Оголошується змінна з плаваючою точкою, яка має десяткове значення:

```
var d = 3.14
```

Оголошується строкова змінна:

```
var str23 = 'Строкова змінна'
```

Оголошується логічна змінна:

```
var p = true
```

Тип змінної може змінюватися в процесі виконання програми. Якщо у виразі містяться і числові і рядкові змінні, то числові змінні автоматично приводяться до рядкового вигляду.

Операції

Арифметичні операції

Операція	**Дія**	**Приклад**	**Значення, яке прийме X**

+	Додавання	`x=100+5`
		`str2='Початок'`
		`str1=str2+' кінець'`
			105
			Початок кінець

-	Віднімання	`x=100-5`	95
*	Множення	`x=2*3`	6
/	Ділення	`x=12/3`	4
%	Залишок від ділення (аналогічно mod)	`x=16%3`	1

++	Значення збільшується на 1	`X=2; X++;`	3
--	Значення зменшується на 1	`X=2; X--;`	1

Операції порівняння

Операція	Дія	Приклад	Аналогічно, в Паскалі
==	Дорівнює	`X==10`	`X=10`
!=	Не дорівнює	`x!=5`	`x<>5`
>	Більше	`x>0`	`x>0`
<	Менше	`x<4`	`x<4`
>=	Більше або дорівнює	`x>=y`	`x>=y`
<=	Менше або дорівнює	`X<=5`	`x<=5`

Логічні операції

Операція	Дія	Приклад	Аналогічно, в Паскалі
&&	Аналогічно логічній операції `and`	`X>=2 && y>=2`	`(X>=2)and(Y>=2)`
||	Аналогічно логічній операції `or`	`x>0 || y>0`	`(x>0)or(y>0)`
!	Аналогічно логічній операції not	`!(1 < x && x < 10)`	`Not((1 < x) and ( x < 10))`

Оператори присвоювання

Операція	Дія	Приклад	Значення, яке прийме X	Аналогічно, в Паскалі
=	Привласнює значення змінній	`X=1000;`	1000	`X:=1000;`
+=	Збільшує значення змінної на зазначену величину	`X=1000;` `X+=100;`	1100	`X:=1000;` `X:=X+100;`
-=	Зменшує значення змінної на зазначену величину	`X=1000;` `X-=12;`	988	`X:=1000;` `X:=X-12;`
*=	Примножує значення змінної на зазначену величину	`X=1000;` `X*=2;`	2000	`X:=1000;` `X:=X*2;`

/=	Поділяє значення змінної на зазначену величину	X=1000; X/=2;	500	X:=1000; X:=X/2;
%=	Ділить значення змінної на зазначену величину і повертає залишок	X=1000; X%=5;	0	X:=1000; X:=X mod 5;

Умовний оператор

JavaScript	Pascal
`if (a==2) z=2; else z=3`	`if a=2 then z:=2 else z:=3;`

`if (x>=2 && x<=6)`
`{`
`y=0; z=1;`
`}`
`else`
`{`
`y=1; z=0;`
`}`
	`if (x>=2)and(x<=6) then`
	`begin`
	`y:=0; z:=1;`
	`end`
	`else`
	`begin`
	`y:=1; z:=0;`
	`end;`

Особливості.

Вся умова береться в дужки. Прості умови в дужки можна не брати. Крапка з комою перед `else` обов'язково ставиться, якщо перед `else` немає фігурної дужки «}». Якщо дужка є, то «;» ставити не можна.

Найпростіший ввод/виведення даних. Діалогові вікна

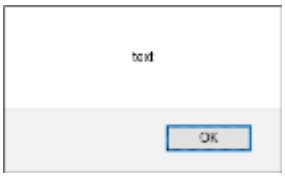
В JavaScript існує 3 найпростіші функції, методу, що дозволяють користувачеві виводити діалогові вікна:

```
alert ("строка")
```

Метод `alert` використовується для виведення найпростішого діалогового вікна, що містить текст повідомлення і єдину кнопку [Ok]. Програма виводить повідомлення і чекає натиснення кнопки. Після натискання на кнопку, програма починає виконуватися далі. Текст повідомлення може зчіплюватися з будь-якою текстовою змінною за допомогою знака «+». Щоб текст виводився в кілька рядків використовують символи «\ n»

Приклад

```
<body>
    <script>
        let t='text';
        alert(t);
    </script>
</body>
```



Confirm (“текст”)

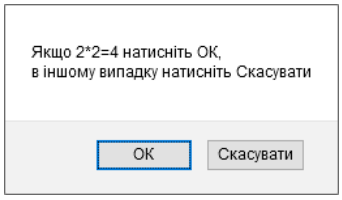
Метод `confirm` використовується в тих випадках, коли користувач повинен зробити вибір. Метод `confirm` дозволяє користувачеві вивести діалогове вікно, що містить текст питання і кнопки **[ОК]** і **[Відмина]**.

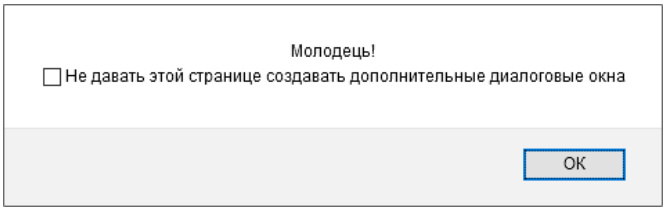
Функція `confirm` повертає логічне значення в залежності від натиснутої користувачем кнопки: **[ОК]** відповідає значенню `true`, **[Відмина]** – значенню `false`.

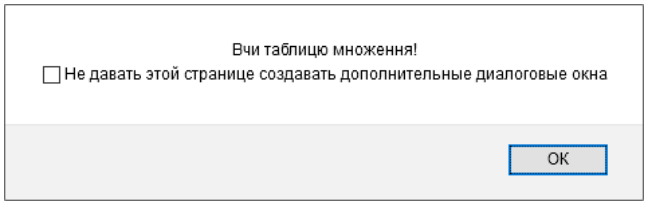
Як правило, результат роботи функції присвоюють логічній змінній, для подальшого аналізу, як це показано в прикладі.

Приклад

```
<body>
    <script>
        var result=confirm("Якщо 2*2=4 натисніть ОК,"+"\n"+"в іншому випадку на
тисніть Скасувати");
        if(result) alert("Молодець!")
        else alert("Вчи таблицю множення!");
    </script>
</body>
```







prompt (“текст1”, “текст2”)

Метод `prompt` використовується в тих випадках, коли користувачеві потрібно ввести значення в змінну. У вікно виводиться повідомлення «***текст1***», в поле введення поміщується значення за замовчування «***текст2***». Цей метод дозволяє вивести діалогове вікно запиту на введення даних.

Результат роботи функції привласнюють змінній рядкового типу.

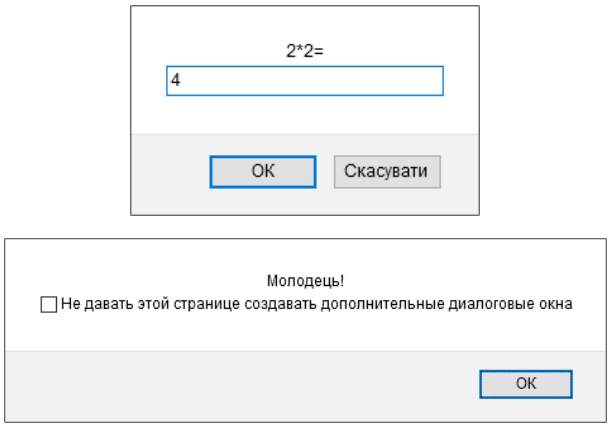
Якщо введені дані потрібно використовувати в арифметичних виразах, необхідно виконати перетворення введеного рядка до числового типу. Це можна зробити за допомогою наступних функцій:

- `parseInt("текст")` – перетворює рядок у ціле число;
- `parseFloat("текст")` – перетворює рядок в дійсне число.

Приклад 1

```
<body>
    <script>
        var str=prompt('2*2=','0');
        if(str=="4") alert('Молодець!')
        else alert('Вчи таблицю множення!');
    </script>
</body>
```

Прибираємо 0 і вводимо 4



Оператор циклу for

Наприклад, цикл нижче виводить значення від 0 до 5:

```
for(початок; умова; крок)
 {
    //...тіло циклу...
 }
```

```
for(i=0; i<5; i++)
 {
     alert(i);
 }
```

Оператор вибору switch

На відміну від умовного оператора `if` відразу організовує розгалуження алгоритму не на 2 частини, а на безліч. Як умова розгалуження можна вибирати вираз, в найпростішому випадку змінну.

Приклад 1

Порівняння значення змінної ***a*** з різними константами.

```
var a = 4;
switch(a)
{
        case 3:   alert('Мало');       break
        case 4:   alert('Точно!');     break
        case 5:   alert('Перебір');    break
  default:  alert('Я таких значень не знаю')
}
```

Результат буде – «Точно!»

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

1 . Обчислити площу прямокутника.
2. Знайти середнє значення двох чисел.
3. Перевірити, чи є число більшим за певне значення.
4. Визначити, чи два числа рівні.
5. Перевірити, чи є число парним і додатнім.
6. Збільшити значення змінної на 1.
7. Відняти певне значення від змінної.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Які існують основні типи даних в JS ?
2. Конструкція циклу for, пояснити складові
3. Пояснити призначення оператору document.write ()
4. Пояснити призначення основних математичних методів
5. Конструкція умовного оператора if/if else/else

Лабораторна робота №14

Тема: Вбудовані об'єкти мови JavaScript.
Мета: Ознайомитися з роботою вбудованих об’єктів мови JavaScript.

ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Основною одиницею в мові JavaScript є об'єкт, який об'єднує в собі дані (властивості) і засоби обробки цих даних (методи).

Всі об'єкти, які використовуються в мові JavaScript, діляться на такі великі групи:

- Вбудовані об'єкти базового мови
  - ✓ Об'єкт `Math`
  - ✓ Об'єкт `Array`
  - ✓ Об'єкт `Date`
  - ✓ Об'єкт `String`
- Об'єкти документа

Вбудовані функції

`parseInt(str)`

Перетворення `str` строки в ціле число.

Якщо `parseInt` стикається з неприпустимим символом, то повертає значення, засноване на підстроки, наступного до цього символу, ігноруючи всі наступні. Якщо перший же символ не допустимий, `parseInt` повертає значення `NaN`.

`parseFloat(str)`

Перетворення рядка `str` в число з плаваючою точкою.

Якщо `parseFloat` стикається з неприпустимим символом, то повертає значення, засноване на підрядку, наступного до цього символу, ігноруючи всі наступні. Якщо перший же символ не допустимий, `parseFloat` повертає `NaN`.

`isNaN(str)`

Якщо рядок `str` не є числом, повертає `true`, інакше – `false`.

Об'єкт `Math`

У мові JavaScript існує спеціальний об'єкт `Math`, в якому зібрані основні математичні функції і константи. Всі властивості і методи об'єкта є статичними, тому сам об'єкт створювати не потрібно. У таблицях нижче наведено список деяких з його методів.

Метод	Опис
`abs(число)`	Повертає модуль (абсолютну величину) числа.

sqrt(число)	Повертає квадратний корінь з числа.
pow(основа, ступінь)	Повертає результат піднесення основи в зазначену ступінь. Наприклад, Math.pow(5, 3) поверне $5^3 = 125$.
random()	Повертає псевдовипадкове число в діапазоні від 0 до 1.
ceil(число)	Проводить округлення в більшу сторону, тобто повертає найменше ціле число, більше або рівне аргументу.
floor(число)	Проводить округлення в меншу сторону, тобто повертає найбільше ціле число, менше або рівне аргументу.
round(число)	Округлює вказаний аргумент до цілочисельного значення.
cos(число)	Повертає косинус числа.
sin(число)	Повертає синус числа.
exp(число)	Зводить число e (основу натурального логарифма) в зазначену ступінь.
log(число)	Повертає натуральний логарифм числа.
PI	Число пи

Щоб задіяти метод (зазвичай це називають викликом методу), в операторі JavaScript потрібно зробити на нього посилання. Для цього використовується синтаксис з використанням точки. Наприклад:

```
x=Math.sqrt(4.1);
```

Об'єкт Array

Об'єкт Array призначений для зберігання масивів даних. Масив – це упорядкований набір елементів. Доступ до окремого елементу проводиться по імені і індексу (номеру). Нумерація елементів в JavaScript починається з нуля.

Приклад:

Заповнити одновимірний цілочисельний масив випадковими числами з (5;20) і вивести на екран

```
<script>
  ar = new Array(10);
  s = "";
  for (i = 0; i < ar.length; i++) {
    x = Math.random() * 15 + 5;
    ar[i] = Math.floor(x);
    s = s + " " + ar[i];
  }
  alert(s);
</script>
```

Об'єкт Date

Об'єкт `Date` призначений для маніпуляцій з датами і часом. Його примітивним значенням є число, яке дорівнює кількості мілісекунд щодо базового часу, рівного півночі 1 січня 1970 р. День складається з 86400000 мілісекунд.

Об'єкт і екземпляр об'єкта

Об'єкт – це шаблон. Примірник об'єкта – це робоча копія. Наприклад, об'єктом є комплект документації на заводі, по якій виготовляються телевізори. Сам телевізор є екземпляром цього об'єкта. Всі телевізори, які сходять з конвеєра, мають одні і ті ж властивості зображення і одні і ті ж методи управління цими властивостями.

Створення екземпляра об'єкта Date

Для створення екземпляра об'єкта використовується ключове слово `new`. Створення екземпляра об'єкта з поточною датою і часом: `new Date()`

```
var a = new Date();
```

В змінній `a` поточна дата і час.

Створення екземпляра об'єкта з датою і часом, заданими аргументами конструктора:

```
new Date(год, месяц, день [,часы [,минуты [,секунды [,мс]?]?]?]?)
```

Тут:

- рік – числовий вираз, що задає повний номер року (наприклад, 1988, а не 88);
- місяць – числовий вираз, що задає номер місяця (0 = січень, 1 = лютий, ..., 11 = грудень);
- день – числовий вираз, що задає номер дня у місяці від 1 до 31;
- години – необов'язковий числовий вираз, що задає номер години від 0 до 23;
- хвилини – необов'язковий числовий вираз, що задає номер хвилини від 0 до 59;
- секунди – необов'язковий числовий вираз, що задає номер секунди від 0 до 59;
- мс – необов'язковий числовий вираз, що задає номер мілісекунди від 0 до 999.

Наприклад,

```
var c = new Date(1958, 4, 21, 10, 15);
```

В змінній «c» дата – 21 травня 1958 року і час 10 годин 15 хв. Нумерація місяців починається з 0.

Методи отримання компонентів об'єкта Date

Метод – це дія яка виконується для об'єкта або з об'єктом. За своєю суттю це команда, але її дії пов'язані з певним об'єктом. У кожного об'єкта може бути багато методів.

Метод	Опис

getTime()	Повертає примітивне значення об'єкта.
getFullYear()	Повертає номер року за місцевим часом
getMonth()	Повертає місяць за місцевим часом. Нумерація місяців з 0 (0 – січ, 1 – лют ...)
getDate()	Повертає число за місцевим часом.
getDay()	Повертає день тижня за місцевим часом. Нумерація днів тижня з 0 (0 – неділя, 1 – понед ...)
getHours()	Повертає години за місцевим часом.
getMinutes()	Повертає хвилини за місцевим часом.
getSeconds()	Повертає секунди за місцевим часом.

Наприклад, виведемо в вікно номер поточного року.

```
<script>
    td = new Date();
    God = td.getFullYear();
    alert(God);
</script>
```

Об'єкт String

Об'єкт `String` призначений для зберігання текстових даних. Доступ до окремого елементу проводиться по імені і індексу (номера). Нумерація символів в рядку в JavaScript починається з нуля. Методи об'єкта:

- Метод `length` повертає довжину рядка.
- Метод `indexOf` повертає перше входження символу
- Метод `lastIndexOf`, який повертає останнє входження символу
- Метод `charAt` повертає символ, що стоїть на зазначеній позиції. Приклад пошуку і виведення довжини рядка:

```
<script>
    str = "Це рядок символів";
    ln = str.length;
    alert('Довжина рядка=' + ln);
</script>
```

Завдання. Вбудовані об'єкти мови JavaScript.

Варіант 1

1. Написати програму, яка водить значення **x**, **y**, а потім обчислює і виводить значення виразу **z**:

$$z = \frac{xy}{x - y} + \sqrt{|y^3 - x|}$$

2. Заповніть цілочисельний масив А з 10 елементів випадковими цілими числами з проміжку (0;50). Знайдіть в цьому масиві числа, які починаються на 3.
3. Дано число, номер місяця і рік. Створити сторінку, яка при завантаженні виводить у вікно цю дату у вигляді: число, назва місяця, рік і назва дня тижня.
4. Дано рядок і символ. Скільки разів зустрічається цей символ в рядку?

Варіант 2.

1. Написати програму, яка вводить значення **x**, **y**, а потім обчислює і виводить значення виразу **z**:

$$z = \frac{|x^3 - y^3|}{(x+y)^2} - \sqrt{\frac{1+|y|}{x}}$$

2. Заповніть цілочисельний масив А з 20 елементів випадковими цілими числами з проміжку (0;60). Знайдіть в цьому масиві числа, які кратні 7.
3. Створити сторінку, яка при завантаженні виводить у вікно сьогоднішню дату у вигляді: число, назва місяця, рік і назва дня тижня.
4. Дан рядок. Скільки разів зустрічаються цифри в цьому рядку?

КОНТРОЛЬНІ ЗАПИТАННЯ

1.Принцип використання типу даних масив.

2. Конструкція циклу while, пояснити складові

3. Конструкція умовного оператора swich

Список літературних джерел.

1. Берко А.Ю., Верес О.М., Пасічник В.В. Системи баз даних та знань. Книга 2. Системи баз даних та знань. / В-во: «Магнолія-2006», 2013 - 584с.

2. Кадемія М.Ю., Козяр В.М., Кобися В.М., Коваль М.С. Соціальні сервіси Веб 2.0 і Веб 3.0. у навчальній діяльності: навчальний посібник. – Вінниця: ТОВ «Планер», 2010. – 230 с.

3. Пасічнік О.Г., Пасічнік О.В., Стеценко І.В. Основи веб-дизайну/ В-во: Вид. група BHV, 2009 - 336с.

4. Пасічник В. В., Пасічник О. В., Угрин Д. І. Веб-технології : підручник. Львів : Магнолія, 2013 - 215 с.

5. О.Б.Проценко. Web-програмування та web-дизайн. Технологія XML. / В-во: Вид-во СумДУ, 2009-127с.

6. Олексій Васильєв. Програмування мовою PHP. / В-во: Ліра-К, 2022 - 368с.

7. Майкл Ховард, Дэвид Лебланк, Джон Виега Як написати безпечний код на C++, Java, Perl, PHP, ASP/В-во:Print2print, 2013 – 288с.

8. Олексій Васильєв. Програмування мовою Java. / В-во: Навчальна книга-Богдан, 2022-696с.

9. Елізабет Робсон, Ерік Фрімен. Head First. Програмування на JavaScript / В-во Фабула, 2022 – 672с.

Зміст