МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЦЕНТРАЛЬНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

Методичні рекомендації

**до виконання лабораторних робіт
з дисципліни
«Web-програмування»
для студентів денної та заочної форми навчання
за спеціальностями 122/F3 «Комп'ютерні науки», 123/F7 «Комп'ютерна інженерія»,
125/F5 «Кібербезпека та захист інформації», 172/G5 «Електроніка, електронні комунікації, приладобудування та радіотехніка»**

ЗАТВЕРДЖЕНО
на засіданні кафедри кібербезпеки та програмного забезпечення,
протокол від 25.08.2025 року № 1

Кропивницький
2025

Методичні рекомендації до виконання лабораторних робіт з дисципліни «Web-програмування» для студентів денної та заочної форми навчання за спеціальностями 122/F3 «Комп'ютерні науки», 123/F7 «Комп'ютерна інженерія», 125/F5 «Кібербезпека та захист інформації», 172/G5 «Електроніка, електронні комунікації, приладобудування та радіотехніка» / [уклад. : Є. В. Мелешко, В. В. Босько, Л. В. Константинова] ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програмного забезпечення. - Кропивницький : ЦНТУ, 2025. - 87 с.

Укладачі: Мелешко Є. В., Босько В.В., Константинова Л. В.

Рецензенти: Смірнов О. А., д-р техн. наук, професор;
Дрєєв О. М., канд. техн. наук.

Методичні рекомендації до лабораторних робіт з дисципліни «Web-програмування» для студентів спеціальностей 122/F3 «Комп'ютерні науки», 123/F7 «Комп'ютерна інженерія», 125/F5 «Кібербезпека та захист інформації», 172/G5 «Електроніка, електронні комунікації, приладобудування та радіотехніка». Розглядаються основні засоби HyperText Markup Language (Html), основи **C**ascading **S**tyle **S**heets (CSS), JavaScript, форми та основи PHP. Вони містять в собі інструкції виконання завдання, теоретичний матеріал до лабораторних робіт і призначені для студентів ВУЗів, що вивчають комп'ютерні спеціальності та можуть бути корисними для школярів, вчителів та просто особистостей, що цікавляться Web-програмуванням.

ЗМІСТ

Вступ

Майже кожен сьогодні використовує веб-сайти чи веб-додатки для своїх потреб і кількість таких людей зростає щороку.

Вебпрограмування — галузь веброзробки й різновид дизайну, в завдання якої входить проєктування користувальницьких вебінтерфейсів для сайтів або вебдодатків.

Веб-розробка - це процес створення вебсайта або вебдодатку. Термін включає розробку додатків електронної комерції, веб-дизайн, програмування для веб на стороні клієнта й серверу, а також конфігурування веб-серверу. Розробка сайту з нуля розбивається на такі етапи: дизайн сайту (шаблон); фронтенд розробка; бекенд розробка; робота з базами даних; адміністрування сайту.

Різні веб-сайти можуть мати різну структуру– кількість веб-сторінок та їх типи, внутрішню тематичну організацію, сукупність внутрішніх зв'язків. Спільним для усіх сайтів є наявність у структурі головної сторінки, яка пов'язана з усіма тематичними розділами сайту.

Методичні рекомендації до лабораторних робіт з дисципліни «Web-програмування» для студентів спеціальностей 122/F3 «Комп'ютерні науки», 123/F7 «Комп'ютерна інженерія», 125/F5 «Кібербезпека та захист інформації», 172/G5 «Електроніка, електронні комунікації, приладобудування та радіотехніка». Вони містять в собі інструкції виконання, завдання, теоретичний матеріал до лабораторних робіт і призначені для студентів ВУЗів, що вивчають комп'ютерні спеціальності та можуть бути корисними для школярів, вчителів та просто особистостей, що цікавляться Web-програмуванням.

В методичних рекомендаціях представлено практичний і теоретичний матеріал для виконання завдань лабораторних робіт з дисципліни «Web-програмування», список скорочень і термінів, приклад оформлення звіту з лабораторної роботи, список рекомендованих джерел інформації.

Дані методичні рекомендації починають знайомити з основними засобами HyperText Markup Language (Html), основами **C**ascading **S**tyle **S**heets (CSS), JavaScript та PHP. Приводиться багато практичних прикладів та ілюстрацій, які допомагають краще засвоїти викладений матеріал. Розглядаються основні html-теги, основні елементи форматування тексту веб-сторінок, їх атрибути, елементи розмітки веб-сторінок для проектування сайтів. Представлено також різні способи застосування CSS, адаптивний дизайн, основи JavaScript при створенні Web-сторінок. Розглядаються завдання зі створення форм HTML та елементів керування, а також застосування рядкових функцій в PHP.

Теми лабораторних робіт, що розглядаються та оцінювання знань

Методичні рекомендації містять у собі сім лабораторних робіт за наступними темами (таблиця 1):

Таблиця 1 – Теми лабораторних робіт

№	Теми лабораторних робіт
1	Основні засоби html
2	Основи CSS
3	Застосування JavaScript при створенні Web-Сторінок
4	Застосування JavaScript при створенні Web form
5	Вивчення й практичне застосування форм HTML та елементів керування
6	Робота з рядками в PHP
7	Системи керування контентом сайтів
8	Застосування CMS для розробки сайту

Форма підсумкового контролю: залік.

Студент може отримати максимальну кількість балів у випадку якщо відвідував всі лекції, лабораторні заняття, виконував всі завдання з лабораторних робіт, що демонструвались викладачу й підтверджується звітом, а також виконувались всі завдання з самостійної роботи у встановлений термін та у випадку вчасного проходження контролю.

При виконанні й захисті лабораторних робіт після встановленого терміну, одержані бали перераховуються з коефіцієнтом: для самостійної роботи студента -0,3; лабораторної роботи -0,7.

В якості самостійної роботи необхідно виконати завдання згідно обраної студентом і погодженої з лектором теми.

На початку кожної лабораторної роботи вказано її тему, мету, що необхідно знати і вміти. Коротко описано теоретичний матеріал з даної теми. Вивчивши теоретичний матеріал лекцій, уважно прочитавши завдання, яке розташоване після теоретичного матеріалу, необхідно вибрати свою тематику, згідно варіанту (номер варіанту узгоджується з викладачем). Ви повинні виконати всі завдання до лабораторних робіт, а також відповісти на запитання, що знаходяться вкінці кожної лабораторної роботи. Звіт повинен містити хід виконання завдань, лістинг програми (з коментарями), а також графічні матеріали, що підтверджують виконання цих завдань, відповіді на запитання. Приклад оформлення звіту з лабораторної роботи дивись в Додатку 1.

Для виконання завдань Вам знадобиться текстовий редактор (Notepad++, HTML-Kit, Crimson Editor, PSPad, Gedit, Eclipse, Bluefish, jEdit, Sublime Text і т.д.) та будь-який Інтернет-браузер.

При виборі редактора необхідно враховувати зручність, функціональність, наскільки актуальна остання версія та чи триває підтримка.

В таблиці 2 представлено шкалу оцінювання, за якою вимірюються набуті знання у студентів.

Таблиця 2 - Шкала оцінювання: національна та ECTS

Сума балів за всі види навчальної діяльності	Оцінка ECTS	Оцінка за національною шкалою
		для екзамену, курсової роботи
90 – 100	**A**	відмінно
82-89	**B**	добре
74-81	**C**	
64-73	**D**	задовільно
60-63	**E**	
35-59	**FX**	незадовільно з можливістю повторного складання
0-34	**F**	незадовільно з обов’язковим повторним вивченням дисципліни

Студенти, що вчасно виконують всі завдання та приймають активну участь у конференціях з тематикою дисципліни «Web-програмування» отримують додаткові бали.

Методичні рекомендації до лабораторних робіт з дисципліни «Web-програмування» у комплексі з іншою допоміжною літературою та рекомендованими джерелами інформації допоможе читачам стати затребуваними та досвідченими фахівцями у галузі веброзробки.

Лабораторна робота №1

Тема: Основні засоби html.

Мета: Познайомитись із синтаксисом, основними тегами й атрибутами мови html.

Знати: Структуру html-документу, основні теги мови й методи форматування тексту.

Вміти: Компонувати текстову інформацію при створенні гіпертекстових документів з використанням простих текстових редакторів. Формувати гіпертекстові посилання й списки й створювати на цій основі зв'язані гіпертекстові сторінки.

Теоретичні відомості

HTML (англ. HyperText Markup Language) – стандартна мова розмітки гіпертекстових документів (веб-сторінок) в Інтернеті [1][2][3].

Мова HTML інтерпретується браузерами. Отриманий в результаті інтерпретації відформатований текст відображається на екрані монітора комп'ютера або мобільного пристрою.

Розглянемо деякі можливості **HTML.**

Мета розробки HTML5 – поліпшення рівня підтримки мультимедіа-технологій з одночасним збереженням зворотної сумісності, зручності читання коду для людини і простоти аналізу для парсерів.

Розробкою та впровадженням стандартів для мережі Інтернет, в тому числі й стандартів для мови HTML займається організація W3C (англ. World Wide Web Consortium (https://www.w3.org/) – консорціум Всесвітньої павутини). Переглянути інформацію про html на веб-сайті можна за наступним посиланням:

https://html.spec.whatwg.org/multipage// [4]

В мові для розмітки тексту використовуються спеціальні команди, які мають назву "теги". Теги можуть бути парними та непарними (
, <img>, <hr>).

Розглянемо структуру html-документів та основні html-теги.

Синтаксис html

```
<назва_тегу> контент </назва_тегу>
```

Приклад 1. Приклад HTML-тегу:

```
<p>Мій перший HTML-параграф</p>
```

Приклад 2. Приклад структури HTML5-документу:

```
<!DOCTYPE html>
<html>
   <head>
       <title>Назва сторінки</title>
   </head>
<body>
      <h1>Заголовок статті</h1>
       <p>Абзац тексту</p>
</body>
</html>
```

Для того, щоб побачити результат роботи даного коду, зробіть наступні кроки:

Крок 1: Відкрийте текстовий редактор.

Крок 2: Скопіюйте код з Прикладу 2.

Крок 3: Збережіть файл з розширенням html.

Крок 4: Відкрийте збережений файл у будь-якому Інтернет-браузері.

Для роботи з HTML можна використовувати спеціалізовані текстові редактори, що полегшують роботу з кодом за допомогою додаткових функцій, таких, наприклад, як підсвічування коду. Наведемо деякі безкоштовні редактори для роботи з html-кодом.

Безкоштовні редактори для Windows:

– Notepad++ – Підтримує FTP та SFTP за допомогою плагіну; підсвічування синтаксису.

– HTML-Kit – Підсвічування синтаксису, підтримка FTP.

– Crimson Editor – Легковагий редактор. Підтримка FTP.

– PSPad – Підтримка FTP; підсвічування синтаксису.

– Вбудований редактор Total Commander.

Безкоштовні редактори для Linux:

– Gedit – вільний текстовий редактор робочого середовища GNOME, Mac OS X і Microsoft Windows з підтримкою Юнікода.

– Kate (KDE Advanced Text Editor) – текстовий редактор, який входить у склад середовища робочого столу KDE. Поширюється згідно GNU General Public License.

При виборі необхідно враховувати зручність, функціональність, наскільки свіжа остання версія і чи триває підтримка.

Кросплатформені редактори:

– Eclipse – Проекти PHPEclipse і PHP Development Tools. З додатковими плагінами підтримує SVN, CVS, моделювання баз даних, доступ по SSH/FTP, навігація базою даних, інтеграція з Trac, та інше.

– Bluefish – Багатоцільовий редактор з підтримкою синтаксиса PHP, встроенной PHP документацией и т.д. З GVFS підтримує SFTP, FTP, WebDAV і SMB.

– jEdit –Versatile вільний/open source редактор. Підтримує SFTP і FTP.

– Sublime Text – кросплатформений пропріетарний текстовой редактор. Підтримує плагіни на мові програмування Python.

Розглянемо детальніше html-код з прикладу 2.

DOCTYPE декларує визначення типу документа HTML.

Після оголошення версії й типу документа необхідно позначити його початок і кінець. Це робиться за допомогою тега-контейнера <HTML>. Потім, між тегами <HTML> і </HTML> варто розмістити заголовок і тіло документа.

Текст між <head> та </head> містить інформацію про поточний документ, таку як заголовок, ключові слова, які можуть використовуватися пошуковими машинами, та інші дані, які не вважаються вмістом документа.

Текст між <title> та</title> служить для ідентифікації вмісту документа (зовнішнього заголовка документа).

Текст між <body> і </body> описує видимий вміст сторінки.

Текст між <h1> і </h1> описує найбільший заголовок.

Текст між і визначає параграф.

HTML заголовки

HTML заголовки визначаються тегами від <h1>до <h6>, наприклад:

<h1>This is a heading</h1>

<h2>This is a heading</h2>

<h3>This is a heading</h3>

Цифра визначає розмір заголовку – найбільший заголовок – 1, найменший – 6.

HTML параграфи

HTML параграфи визначаються тегом , наприклад:

Це параграф

Це інший параграф

HTML Посилання

HTML посилання визначаються тегом <a>, наприклад:

Це посилання

HTML Зображення

HTML зображення визначається тегом **<img>.**

Файл зображення (**src**), альтернативний текст (**alt**), розмір зображення (**width** and **height**) надаються як атрибути, наприклад:

<img src="img1.jpg" alt="зображення 1" width="104"height="142">

HTML Атрибути

HTML елементи можуть мати атрибути.

Атрибути містять додаткову інформацію про елемент, завжди визначаються в початковому тегу. Атрибути є парами ім’я/значення як: **name="value"**. Приклади:

Атрибут мови <html lang="en-US">

Атрибут назви

Атрибут посилання Це посилання

Атрибути розміру <img src="site.jpg" width="104" height="142">

Атрибут alt <img src=" img2.jpg" alt="рис.2" width="104" height="142">

В таблиці 3 перелічено Html-атрибути, що використовуються найчастіше, в таблиці 4 – Html-елементи форматування тексту, вибір кольору шрифта html – сторінки показано в таблиці 5.

Таблиця 3 – Html-атрибути

Атрибути	**Опис**
Alt	Вказує альтернативний текст для зображення
Disabled	Вказує, що вхідний елемент повинен бути відключений
href	Вказує URL-адресу (веб-адресу) для зв'язку
Id	Вказує унікальний ідентифікатор для елемента
Src	Вказує URL-адресу (веб-адресу) для зображення
Style	Визначає CSS стиль вбудованого елементу
Title	Визначає додаткову інформацію про елемент (відображається у вигляді підказки)
Value	Визначає значення (текстовий зміст) для вхідного елемента.

Приклад 3. Приклад HTML5-документу з використанням css-елементів для вказання стилю тегів:

```
<!DOCTYPE html>
<html>
<body>
<h2 style="color:red">Я червоний заголовок</h2>
<h2 style="color:blue">Я синій заголовок</h2>
</body>
</html>
<body style="background-color:lightgrey">
<h1 style="font-family:verdana">Це заголовок</h1>
<p style="font-family:courier">Це параграф</p>
<h1 style="font-size:300%">Це заголовок</h1>
<p style="font-size:160%">Це параграф>
<h1 style="text-align:center">Це заголовок</h1> …
```

Таблиця 4 – Html-елементи форматування тексту

Тег	**Опис**
<b>	Жирний текст
	Жирний текст, визначає важливість виділеного тексту
<i>	Курсив
	Курсив, визначає важливість виділеного тексту
<small>	Зменшує розмір шрифту на одну умовну одиницю. В HTML розмір шрифту вимірюється в умовних одиницях від 1 до 7. Середній розмір тексту, що використовується за замовчуванням – 3.Допускається застосування вкладених тегів <small>, при цьому розмір шрифту буде менше з кожним вкладеним рівнем, але не може бути менше, ніж 1.
<sub>	Нижній індекс
<sup>	Верхній індекс

Таблиця 5 - Вибір кольору шрифта html - сторінки

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

HTML-код:	**Відображення у браузері:**
червоний колір Синій колір	Червоний колір Синій колір

Списки

В таблиці 6 представлено Html-теги списків.

Таблиця 6 – Html-теги списків

Тег	**Опис**
	Ненумерований список
	Нумерований список
	Елемент списку

Приклад 4. Приклад ненумерованого списку.

```
<!DOCTYPE html>
<html>
<body>
<h2>Unordered List with Default Bullets</h2>
<!—А ось наш ненумерований список нашого веб-сайту -->

<ul>
 <li> Яблука </li>
 <li> Банани </li>
 <li> Лимони </li>
 <li> Мандарини </li>
</ul>
</body>
</html>
```

Коментар

Щоб закоментувати текст в html-документі використовують <!-- Коментар -->. Так в прикладі 4 застосовується коментар:
<!--А ось наш ненумерований список нашого веб-сайту -->
тому цей текст не буде відображатись у браузері.

Таблиці

В таблиці 7 представлено Html-теги для роботи з таблицями.

Таблиця 7 – Html-теги таблиці

Тег	Опис
<table>	Таблиця
<th>	Комірка заголовку таблиці
<tr>	Рядок таблиці
<td>	Комірка таблиці
<caption>	Заголовок таблиці

Приклад 5. Приклад таблиці з використанням css-елементів для вказання стилю її елементів:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
}
</style>
</head>
<body>
<table style="width:100%">
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
  <tr>
    <td>John</td>
    <td>Doe</td>
    <td>80</td>
  </tr>
</table>
</body>
</html>
```

На рисунку 1 представлено результат роботи коду з прикладу 5.

Jill	**Smith**	**50**
Eve	**Jackson**	**94**
John	**Doe**	**80**

Рисунок 1 – Результат роботи коду з прикладу 5

Теги групування елементів

Таблиця 8 – Html-теги для групування елементів

Тег	Опис
	є блоковим елементом і призначений для виділення фрагмента документу з метою зміни його стилю. Щоб не описувати кожен раз стиль всередині тега, можна виділити стиль в зовнішню таблицю стилів, а для тегу додати атрибут class або id з ім'ям селектора.
	призначений для визначення стилю малих елементів документа. За допомогою тега можна виділити частину інформації всередині інших тегів і встановити для неї свій стиль. Наприклад, всередині абзацу <p> можна змінити колір і розмір першої літери, якщо додати початковий і кінцевий тег і визначити для нього стиль тексту. Щоб не описувати кожен раз стиль всередині тега, можна виділити стиль в зовнішню таблицю стилів, а для тегу додати атрибут class або id з ім'ям селектора.

Основні складові документа

Веб-сторінки можуть і відрізнятимуться одна від одної, але всі вони переважно складаються з аналогічних стандартних компонентів, якщо тільки сторінка не відображає повноекранне відео або гру, не є частиною якогось художнього проекту або просто погано структурована [5].

Перш ніж почати працювати з будь-яким сайтом, потрібно чітко продумати його структуру. Це безпосередньо впливає на ранжування ресурсу в пошукових системах, і навіть на його сприйняття користувачами.

Приклад 6. Приклад структури сайту (стрілки ілюструють один з можливих варіантів посилань сторінок одна на одну) зображено на рисунку 2:

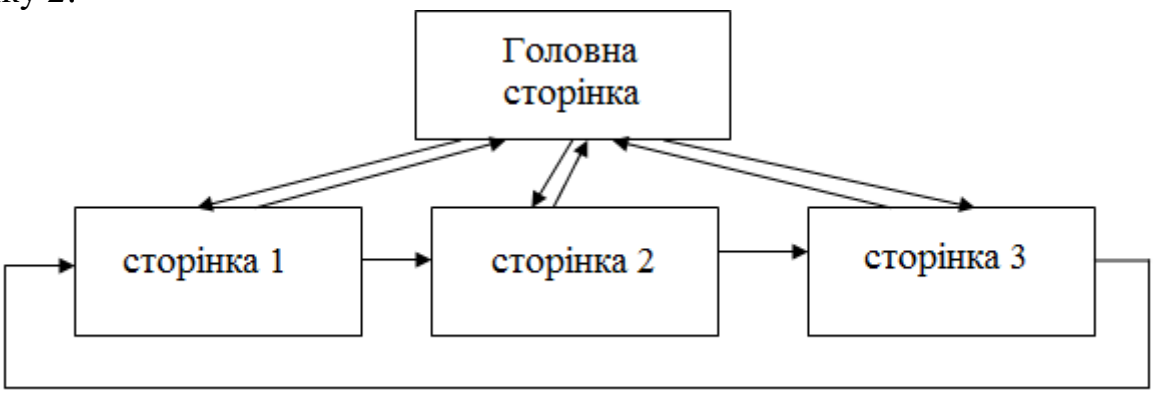


Рисунок 2 – Структура сайту

Елементи розмітки веб-сторінок в HTML5

HTML5 пропонує нові семантичні елементи, які визначають різні частини веб-сторінки, наприклад: <header>, <nav>, <section>, і <footer>.

Приклад 7. Макет сайту у кілька колонок розглянуто на рисунку 3:

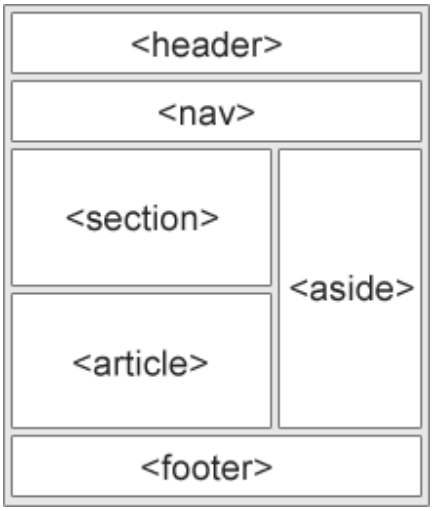


Рисунок 3 – Макет сайту

Таблиця 9 – Елементи розмітки веб-сторінок в HTML5

Тег	**Опис**	
Header	Заголовок документа або розділу. Зазвичай це велика смуга вгорі сторінки, з великим заголовком та/або логотипом. Тут наводиться загальна інформація про веб-сайт, що не змінюється від сторінки до сторінки.	
Nav	Контейнер для навігаційних посилань. Посилання на основні розділи сайту; зазвичай це кнопки, посилання чи вкладки. Як і заголовок, навігація залишається незмінною на всіх сторінках сайту.	
Section	Розділ в документі.	Велика область в центрі сторінки, що містить, в основному, унікальний контент даної сторінки, наприклад відео, розповідь або карту, яку ви хочете переглянути, або заголовки новин і т.д. одна з частин сайту, яка безперечно змінюватиметься від сторінки до сторінки!
Article	Незалежна самодостатня стаття.	
Aside	Бічна панель. Зазвичай містить деяку другорядну інформацію, посилання, цитати, рекламу тощо. Зазвичай вона відноситься до вмісту в основному контенті.	
Footer	Нижній колонтитул документа або розділу. Смуга в нижній частині сторінки, яка зазвичай містить повідомлення про авторські права або контактну інформацію. Це місце для розміщення загальної інформації.	

В таблиці 10 представлено HTML-елементи, що містять інформацію про поточний документ.

Таблиця 10 – HTML-елементи, що містять інформацію про поточний документ

Тег	Опис
<head>	Інформація про документ
<title>	Назва документа
<base>	Адреса за замовчуванням або мета за замовчуванням для всіх посилань на сторінці
<link>	Встановлює зв'язок із зовнішнім документом на зразок файлу зі стилями або з шрифтами.
<meta>	Визначає мета теги, які використовуються для зберігання інформації призначеної для браузерів і пошукових систем. Наприклад, механізми пошукових систем звертаються до метатегів для отримання опису сайту, ключових слів та інших даних.
<script>	Визначає сценарій на стороні клієнта
<style>	Визначає інформацію про стилі для документу

Елемент <video>

Елемент <video> [5] дозволяє застосовувати відео в документ достатньо легко. Наприклад:

```
<video src="rabbit320.webm" controls>
  <p>Ваш браузер не підтримує HTML5 відео. застосуйте <a
href="rabbit320.webm">посилання на відео</a> для доступу.</p>
</video>
```

Опис параметрів: src (source - джерело, схоже з елементом <img>) утримує путь до відео, яке потрібно застосувати. Параграф усередині тегів <video> - називають резервний контент - він відображатиметься, якщо браузер, який показує сторінку, не підтримує елемент <video>.

Формати, такі як MP3, MP4 та WebM, називаються форматами контейнерів. Вони містять різні частини, які складають всю пісню або відео - наприклад, звукову доріжку, відеодоріжку (у разі відео) та метадані для опису представленого носія.

Кожен браузер не лише підтримує свій набір форматів файлів-контейнерів, але й кожен із них підтримує свій вибір кодеків. Щоб максимізувати ймовірність того, що ваш веб-сайт або програма буде працювати в браузері користувача, вам може знадобитися надати кожен медіафайл, який ви використовуєте, у кількох форматах. Якщо ваш сайт та браузер користувача не використовують загальний медіаформат, ваші медіа просто не відтворюватимуться. Тому підтримка мультимедіа частково залежить від того, яке програмне забезпечення встановив користувач. В цьому випадку краще застосувати адаптований приклад:

```
<video controls>
  <source src="rabbit320.mp4" type="video/mp4">
  <source src="rabbit320.webm" type="video/webm">
  <p>Ваш браузер не підтримує HTML5 відео. Ось <a
href="rabbit320.mp4">посилання на відео</a> на випадок.</p>
</video>
```

Елемент <audio>

Елемент <audio> [5] працює схоже з елементом <video>, з невеличкими відмінностями. Наприклад:

```
<audio controls>
  <source src="viper.mp3" type="audio/mp3">
  <source src="viper.ogg" type="audio/ogg">
  <p>Your browser doesn't support HTML5 audio. Here is a <a
href="viper.mp3">link to the audio</a> instead.</p>
</audio>
```

Для отримання додаткової інформації звертайтесь за посиланням [5]: https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Video_and_audio_content.

Завдання:

Створити статичний сайт, що містить мінімум 6 web-сторінок (рисунок 2) з тематики відповідно до свого **варіанту**.

Обов'язково використайте теги:

- форматування тексту,
- створення списків,
- таблиць,
- посилань,
- додавання малюнків,
- елементи розмітки веб-сторінок,
- аудіо та відео.

Варіанти тематик:

Варіант 1

Сайт – тлумачний словник з ілюстраціями

Варіант 2

Сайт про динозаврів

Варіант 3

Сайт з фотографіями та описом квітів з Червоної книги

Варіант 4

Сайт – кулінарна книга з рецептами млинців

Варіант 5

Інтернет-магазин дитячих іграшок

Варіант 6

Інтернет-магазин книг

Варіант 7

Сайт – довідник з музичних термінів

Варіант 8

Сайт-візитка одного з казкових героїв

Варіант 9

Сайт про історію обчислювальної техніки

Варіант 10

Сайт про пташок колібрі

Варіант 11

Сайт про дослідження космосу

Варіант 12

Сайт-візитка одного з римських богів

Варіант 13
Сайт – довідник про кисломолочні продукти
Варіант 14
Сайт – довідник про музичні напрямки
Варіант 15
Сайт – довідник про рослинних шкідників
Варіант 16
Сайт, присвячений улюбленому письменнику
Варіант 17
Сайт, присвячений улюбленій музичній групі чи співаку/співачці
Варіант 18
Сайт наукових новин
Варіант 19
Сайт анекдотів та карикатур
Варіант 20
Сайт віршів та привітань
Варіант 21
Сайт – довідник з мови програмування С++
Варіант 22
Сайт, присвячений певній країні (на вибір студента)
Варіант 23
Сайт, присвячений різним породам котів
Варіант 24
Інтернет-магазин настільних ігор
Варіант 25
Інтернет-магазин квітів
Варіант 26
Сайт, присвячений певній політичній організації (на вибір студента)
Варіант 27
Інтернет-магазин будівельних матеріалів
Варіант 28
Сайт фірми з продажу та ремонту друкуючих пристроїв
Варіант 29
Сайт спортивного клубу
Варіант 30
Сайт кінотеатру
Варіант 31
Сайт краєзнавчого музею
Варіант 32
Сайт для дошкільного навчального закладу

Контрольні питання:

1. Що таке HTML?
2. Яким чином забезпечується форматування тексту в HTML-документі?
3. Яка мета розробки HTML5?
4. Який синтаксис HTML?

5. Структура HTML-документа?
6. Яким чином здійснюється взаємодія між HTML-документом і сервером?
7. Які текстові редактори для роботи з HTML Ви знаєте?
8. Що означає DOCTYPE у HTML-документі?
9. Що таке заголовок HTML-документа?
10. Що таке тіло HTML-документа?
11. Для чого служить текст між тегами <title>?
12. Яким тегом забезпечуються посилання на інші документи HTML і його атрибути?
13. Яким тегом визначаються параграфи?
14. Яким тегом визначаються заголовки?
15. Яким тегом визначається найбільший заголовок?
16. Яким тегом визначається найменший заголовок?
17. Що визначається тегом <img>?
18. Перелічити HTML - елементи форматування тексту.
19. Які теги списків ви знаєте?
20. Що визначає тег <table>?
21. Яким тегом визначається заголовок таблиці?
22. Для чого використовують тег ?
23. Для чого використовують тег ?
24. Що визначають <header>, <nav>, <section>, і <footer> елементи?
25. Які елементи дозволяють застосовувати аудіо та відео контент?

Лабораторна робота №2

Тема: Основи CSS.

Мета: Познайомитися із синтаксисом та основними елементами CSS.

Знати: Способи створення різних дизайнів сайту за допомогою CSS.

Вміти: Встановлювати стилі для HTML-елементів за допомогою селекторів CSS.

Теоретичні відомості

CSS (англ. Cascading Style Sheets, каскадні таблиці стилів) – спеціальна мова, що використовується для візуального оформлення сторінок, написаних HTML та XHTML.

CSS – каскадна (або блочна верстка), що прийшла на заміну табличній верстці вебсторінок. Головна перевага блочної верстки – розділення змісту сторінки (даних) та їх візуального оформлення.

Розробкою та впровадженням стандартів для мережі Інтернет, в тому числі і стандартів для мови CSS займається організація W3C (англ. World Wide Web Consortium – консорціум Всесвітньої павутини). Переглянути інформацію про CSS на їхньому веб-сайті можна за наступним посиланням:

https://www.w3.org/Style/CSS/

Використовуючи CSS **можливо**:

- використовувати поля, кеглі й гарнітури шрифтів, індивідуальні кольори тексту й фону для окремих абзаців, слів і навіть букв. Оформляти нові рядки, буквиці та ін.;
- змінювати форматування сторінок і всього сайту в цілому, не доторкаючись до HTML файлів;
- зменшити кількість тегів в HTML тексті й зробити його кращім для сприйняття;
- визначати варіації оформлення за рахунок спадкування класів. Наприклад, визначивши дизайн для тега **<P>**, можна пронаслідувати від нього теги **<P.exclamation>**, **<P.question>** і т.д., які будуть виглядати як **<P>**, за винятком тих атрибутів, які явно перевизначені в похідних стилях;
- гнучко управляти розміщенням елементів документа, включаючи накладення їх один на одного, й інші ефекти;
- керувати форматуванням сторінок під час друку HTML документа;
- створювати різні ефекти для тексту й зображень (тіні, димку, і т.д.).

Синтаксис CSS:

```
Селектор {
властивість1:  значення;
властивість2:  значення;
...
властивістьN:  значення;
}
```

Селектор – ім'я стилю, у якому зазначені параметри форматування.

Типи селекторів:

- селектори тегів,
- селектори ідентифікаторів,
- селектори класів тощо.

Приклад 1

Запишемо блок, що задає червоний колір і кегль в 36 пунктів для всіх заголовків **<H1>** і голубий колір і кегль в 18 пунктів для заголовків **<H2>**.

```
H1 {
    font-size: 36pt;
    color: red
}
H2 {
    font-size: 18pt;
    color: blue
}
```

Втавка в документ CSS-стилів

Існує три способи вставити стиль в HTML документ.

1. Посиланням в HTML документі на .css файл, що містить визначення стилів. При такому підході Ви можете посилатися на той самий файл стилів з будь-якої кількості документів. Тоді, при зміні стилів у цьому файлі всі документи поміняють свій вигляд;

Приклад 2

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

2. Визначити всі стилі, що використовуються на початку сторінки, а потім використовувати. При такому підході Ви можете змінювати стилі на початку документа (один раз) і Ваші зміни відіб'ються у всіх місцях сторінки, де ці стилі використовуються.

Приклад 3

```
<head>
<style type="text/css">
     H1 {
         font-size: 120%;
         font-family: Verdana, Arial, Helvetica, sans-serif;
         color: #336
    }
</style>
</head>
```

3. Вставляти вказівки на правила форматування безпосередньо в тег. Звичайно це використовується для унікального форматування, що більше (крім як у цьому місці документа) ніде не потрібно.

Приклад 4. Додається кольоровий заголовок третім способом.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Hello!</title>
    </head>
    <body>
        <h1 style="color: blue; text-align: center;">A Colorful
Heading!</h1>
        Hello, world!
    </body>
<html>
```

Властивості CSS

Існує дуже багато властивостей CSS, всі ми розглянути не зможемо, але так само, як і елементи HTML, як правило, їх легко знайти за запитом, наприклад, «змінити шрифт на синій CSS». Деякі з найпоширеніших:

- color: колір тексту;
- text-align: де елементи розміщуються на сторінці;
- background-color: можна встановити будь-який колір;
- width: ширина у пікселях або відсотках від сторінки;
- height: висота у пікселях або відсотках сторінки;
- padding: скільки місця слід залишити всередині елемента;
- margin: скільки місця слід залишити поза елементом;
- font-family: тип шрифту для тексту на сторінці;
- font-size: кегль шрифту у пікселях;
- border: тип межі (суцільна, штрихова тощо) та колір.

Селектор типу - це селектор, який визначає стиль для всіх елементів з даним ім'ям. У стилях як селектор у цьому випадку виступає ім'я тегу.

Селектор id (ідентифікатор) – призначений для застосування стилю до унікальних елементів на веб-сторінці. Унікальність означає, що елемент із цим оформленням буде використовуватися на сторінці лише один раз. У ролі таких елементів можуть бути: шапка сайту, нижній колонтитул, меню навігації тощо.

Приклад 5 з селектором id.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Назва документа</title>
    <style>
      #para1 {
        text-align: center;
        color: red;
      }
    </style>
  </head>

  <body>
    <p id="para1">Вітаю!</p>
    <p>Цей абзац буде без стилю.</p>
  </body>
</html>
```

Селектор class - дозволяє (як і селектор id) стилізувати конкретний елемент сторінки, але на відміну від id, селектор class дозволяє застосувати свій стиль кількох елементів на веб-сторінці.

Приємною властивістю стилів є те, що вони можуть успадковувати властивості один одного. У програмуванні об'єкти, що успадковують властивості "батьків", називаються ***класами***.

Можна вказати, що всі теги **<P>** у документі будуть задавати абзаци, написані шрифтом розміром в 20 пунктів блакитного кольору. Потім вказати, що ще будуть спеціальні абзаци, такі ж, але зсунуті вліво на півдюйма та ще абзаци третього виду – написані похилими буквами. Рішення цього завдання може виглядати, наприклад, так:

Приклад 6

```
<style type="text/css">
  p { font-size: 20pt; color: blue}
  p.m { margin-left: 0.5in}
  p.i { font-style: italic}
</style>
<p> Це звичайний абзац. </p>
<p class=m> Це зрушений абзац. </p>
<p class=i> Це похилий абзац. </p>
```

Псевдокласи. Існують заздалегідь визначені, так звані псевдокласи. Наприклад, для стилів посилань – :link - натиснуте посилання, :visited - відвідане посилання, тощо.

Приклад 7. Виконується так, щоб посилання були висотою в 20 пунктів, перекреслені, відвідані посилання - зелені. Не відвідані - червоні.

```
<style type="text/css">
      a { font-size: 20pt; color: green; text-decoration: line-
through }
      a:visited { font-size: 20pt; color: red; text-decoration: line-
through }
</style>
<p><a href="http://www.site.com/1/"> не відвідане посилання </a></p>
<p><a href=" http://www.site.com/2/">відвідане посилання</a></p>
```

Спадкування

HTML документ має ієрархічну структуру. Тіло документа вкладене в теги **<BODY>...</BODY>**. В середині документа можуть бути розділи, вкладені в теги **<DIV>...</DIV>**. Усередині розділів можуть бути ще розділи й т.д.

Ідея спадкування полягає в тому, що всі формати, які визначаються для **<BODY>**, будуть успадковуватися всіма розділами. Можливо перевизначити в розділі деякі формати, а всі інші він успадкує від **<BODY>**.

Таким способом, розділ, вкладений в інший розділ, успадковує формати розділу, що його містить, але може щось перевизначити для себе.

Приклад 8

```
<DIV STYLE=" font-size: 16pt; color=black;">
```

Це перший розділ. Чорні букви розміром в 16 пунктів.

```
<DIV STYLE=" text-decoration: underline; margin-left: 1cm">
```

Це другий розділ усередині першого. Усе успадковано від першого, але букви підкреслені й все зсунуто на 1 см ліворуч від першого розділу.

```
<DIV STYLE="color: blue; margin-left: 1cm">
```

Це третій розділ усередині другого. Усе успадковано від другого, але букви - блакитні й все зсунуто на 1 см ліворуч від другого розділу.

```
</DIV>
```

Це триває другий розділ (третій закінчився)

```
</DIV>
```

Це триває перший розділ (другий закінчився)

```
</DIV>
```

Вкладені розділи успадковують форматування батьків і можуть додавати своє (а якщо треба, то й змінювати батьківське). Однак, як тільки вкладений розділ завершився й триває батьківський, всі його формати відновлюються.

CSS має наступний **пріоритет застосування**:

1. рядковий стиль;
2. id;

3. клас;
4. тип елемента.

Всі макети сайтів, тобто типи розмітки сторінок можна розділити на такі принципові групи [7]:–фіксовані (fixed),–гумові (fluid),–респонсивні (responsive),–адаптивні (adaptive).

Для підлаштування макету під розмір екрану пристрою використовують респонсивний та адаптивний типи макетів.

Адаптивний дизайн сегментує користувачів на категорії за типом пристрою (desktop, tablet, mobile). На сервері в залежності від цього готується відповідна версія дизайну.

Респонсивний дизайн будується за допомогою медіазапитів (mediaqueries), флексбокс (flexbox) елементів та на так званих решітках (grid-layouts).

Важливо переконатися, що вебсайт читабельний для всіх пристроїв, незалежно від розміру екрану.

Область перегляду - це частина екрану, яка фактично видна користувачеві постійно. За замовчуванням багато вебсторінок вважають, що область перегляду однакова на будь-якому пристрої, що призводить до того, що з багатьма сайтами (особливо старими) важко взаємодіяти на мобільних пристроях.

Способи покращення зовнішнього вигляду сайта:

- Додати наведений далі рядок у заголовок HTML-файлів.

<meta name="**viewport**" content="width=device-width, initial-scale=1.0">

Цей рядок вказує мобільному пристрою використовувати область перегляду тієї ж ширини, що і ширина пристрою, який використовується, а не значно більшу.

- Застосування медіазапитів. **Медіазапити** – це способи зміни стилю сторінки на основі того, як саме переглядається сторінка (Приклад 8).

- Використання атрибута CSS - **flexbox**. Це дозволяє нам легко перенести елементи до наступного рядка, якщо вони не вміщуються горизонтально (Приклад 9) та ін.

Приклад 8

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Screen Size</title>
        <style>
            @media (min-width: 600px) {
                body {
                    background-color: red;
                }
            }
            @media (max-width: 599px) {
                body {
                    background-color: blue;
                }
            }
        </style>
    </head>
    <body>
        <h1>Welcome to the page!</h1>
    </body>
</html>
```

Приклад 9

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Screen Size</title>
        <style>
            #container {
                display: flex;
                flex-wrap: wrap;
            }
            #container > div {
                background-color: green;
                font-size: 20px;
                margin: 20px;
                padding: 20px;
                width: 200px;
            }
        </style>
    </head>
    <body>
        <div id="container">
            <div>Some text 1!</div>
            <div>Some text 2!</div>
            <div>Some text 3!</div>
            <div>Some text 4!</div>
            <div>Some text 5!</div>
            <div>Some text 6!</div>
            <div>Some text 7!</div>
            <div>Some text 8!</div>
            <div>Some text 9!</div>
            <div>Some text 10!</div>
            <div>Some text 11!</div>
            <div>Some text 12!</div>
        </div>
    </body>
</html>
```

Валідність HTML-верстки –це її відповідність стандартам організації TheWorldWideWebConsortium (W3C). Відсутність помилок в розмітці документу - це один з головних показників якості розмітки. Перевірка розмітки на помилки та відповідність стандарту може бути проведена автоматично за допомогою он-лайн сервісу W3C чи за допомогою різних програм «валідаторів»

Завдання:

1. Продумайте та створіть дизайн для свого сайту з лабораторної роботи №1.

2. Застосувати CSS для візуального оформлення сторінок:

- Всі стилі сайту мають бути прописані у файлі style.css.
- Застосувати селектори тегів.
- Застосувати селектори id.
- Застосувати селектори класів.

3. Встановити для сторінки `background-color: olive;` для пристроїв з шириною області перегляду від 600 px; `background-color: yellow;` - <600 px (Застосувати медіа запити, приклад 8).

4. Навчитися позиціювати блочні елементи сторінок. Створити 12 блокових елементів з текстом згідно варіанту, встановити `background-color: lime;`. Застосувати атрибут CSS - flexbox для перенесення елементів до

наступного рядка, в залежності від зміни розміру ширини області перегляду (приклад 9).

5. Перевірити відображення макету в різних браузерах та виправити за необхідності помилки. При роботі для налагодження макету скористатися вбудованими в браузери інспекторами коду.

6. Провести валідацію макету.

Варіанти тематик:

Варіант 1
Сайт – тлумачний словник з ілюстраціями

Варіант 2
Сайт про динозаврів

Варіант 3
Сайт з фотографіями та описом квітів з Червоної книги

Варіант 4
Сайт – кулінарна книга з рецептами млинців

Варіант 5
Інтернет-магазин дитячих іграшок

Варіант 6
Інтернет-магазин книг

Варіант 7
Сайт – довідник з музичних термінів

Варіант 8
Сайт-візитка одного з казкових героїв

Варіант 9
Сайт про історію обчислювальної техніки

Варіант 10
Сайт про пташок колібрі

Варіант 11
Сайт про дослідження космосу

Варіант 12
Сайт-візитка одного з римських богів

Варіант 13
Сайт – довідник про кисломолочні продукти

Варіант 14
Сайт – довідник про музичні напрямки

Варіант 15
Сайт – довідник про рослинних шкідників

Варіант 16
Сайт, присвячений улюбленому письменнику

Варіант 17
Сайт, присвячений улюбленій музичній групі чи співаку/співачці

Варіант 18
Сайт наукових новин

Варіант 19
Сайт анекдотів та карикатур

Варіант 20
Сайт віршів та привітань

Варіант 21
Сайт – довідник з мови програмування C++
Варіант 22
Сайт, присвячений певній країні (на вибір студента)
Варіант 23
Сайт, присвячений різним породам котів
Варіант 24
Інтернет-магазин настільних ігор
Варіант 25
Інтернет-магазин квітів
Варіант 26
Сайт, присвячений певній політичній організації (на вибір студента)
Варіант 27
Інтернет-магазин будівельних матеріалів
Варіант 28
Сайт фірми з продажу та ремонту друкуючих пристроїв
Варіант 29
Сайт спортивного клубу
Варіант 30
Сайт кінотеатру
Варіант 31
Сайт краєзнавчого музею
Варіант 32
Сайт для дошкільного навчального закладу

Контрольні питання:

1) Що представляє собою CSS?
2) Для чого використовують CSS?
3) Яка основна перевага блочної верстки?
4) Які існують способи вставки CSS у документ?
5) Який синтаксис CSS записів?
6) Які бувають типи селекторів?
7) Для чого необхідні класи та як їх визначити?
8) Селектори id – для чого використовують?
9) Що означає термін область перегляду?
10) Які способи адаптивного дизайну ви знаєте?
11) Що означають медіазапити?
12) Що означає валідація макету?

Лабораторна робота №3

Тема: Застосування JavaScript при створенні Web-сторінок
Мета: Познайомитися із синтаксисом, основними елементами мови JavaScript.
Знати: Структуру HTML документа, основні елементи мови JavaScript.

Вміти: Компонувати текстову інформацію при створенні гіпертекстових документів з використанням простих текстових редакторів.

Теоретичні відомості

Вставка в код сторінки

JavaScript являє собою повністю інтерпретуєму мову опису сценаріїв із програмним кодом скрипта, що поставляється користувачеві у відкритому вигляді [6].

Як правило, скрипти вставляються безпосередньо в тіло HTML-документа за допомогою спеціального тегу <SCRIPT>. . .</SCRIPT> . Браузер аналізує вміст, що перебуває між цими тегами й для виконання сценарію пускає в хід той або інший інтерпретатор. Проте, для усунення можливих різночитань і колізій є можливість указувати мову і його версію безпосередньо в тілі тегу скрипта: <SCRIPT language JavaScript 1.1> ... </SCRIPT>.

Альтернативним і в багатьох випадках корисним варіантом є розміщення скриптів в окремих файлах. Цей підхід найбільш прийнятний при використанні скриптів досить великого обсягу, а також для зберігання службових процедур. Якщо ми зберігаємо наш скрипт у файлі з ім'ям filel.js, то включити його в код сторінки ми можемо в такий спосіб: <SCRIPT src="file.JS" X/SCRIPT>. У цьому випадку між тегами опису скрипта нічого вставляти не потрібно - необхідний код перебуває адже у файлі filel.js. Ніщо не заважає нам указувати назву мови і його версію в тегу опису скрипта й у цьому випадку.

Якщо ж браузер користувача не має засобів роботи зі скриптами, то корисно використовувати тег <NOSCRIPT>. . .</NOSCRIPT> для виводу відповідно цьому випадку варіанта вмісту сторінки.

Використання скриптів у тілі HTML-Документа

```
<HTML>
<HEAD>  <! - - - Тут дані заголовка ---i->  </HEAD>
<BODY>
<! ---i- HTML-Код -i->
<SCRIPT> // Код скрипта </SCRIPT>
<! ---i- HTML-Код-i->
<SCRIPT language VBScript>  // Код скрипта мовою VBScript </SCRIPT>
<! ---i- HTML-Код-i>
<SCRIPT language JavaScript 1.3 > // Код скрипта мовою JavaScript
</SCRIPT>
<! ---i- HTML-Код-i->
<! ---i- Підключаємо скрипт із зовнішнього файлу ---i->
<SCRIPT src=" script-l.js" language JavaScript 1.1 >  </SCRIPT>
<! ---i-А якщо браузер не знає про скриптах ? ---i>
```

```
<NOSCRIPT>
<CENTER> <H1> !!! Ваш браузер не має підтримки скриптов. Радимо
обзавестися новим програмним забезпеченням !!! </H1> </CENTER>
</NOSCRIPT>
</BODY>
</HTML>
```

Синтаксис. Типи змінних. Масиви

JavaScript є справжньою мовою програмування з усіма властивими йому атрибутами: змінними, операторами контролю процесу, командами ввод/виводу й ін. Синтаксис JavaScript багато в чому схожий із синтаксисом C++. У той же час конструкції JavaScript досить спрощені, що дозволяє освоювати його людям, що не є досить глибокими знавцями програмування.

Для написання коду програм на JavaScript використовується стандартний набір латинських символів. У рядкових виразах можна використовувати символи національних алфавітів. Застосовуються також спеціальні символи, такі як \n - новий рядок, \t - табуляція, \b - вибій, \r - повернення каретки. Як коментар використовується послідовність символів //. У назвах операторів й іменах змінних заголовні й малі літери є рівнозначними. Змінні JavaScript задаються в наступному форматі й можуть бути оголошені в будь-якому місці коду скрипта:

var cnt;

Можливе завдання початкового значення змінної при її оголошенні:

var name = "значення";

Тип змінної задається, як правило, форматом її значення. У процесі виконання програми скрипта та сама змінна може мати різний тип залежно від збереженого в ній значення. Змінна name буде мати рядковий тип. У деяких випадках можна створювати типізовані змінні для породження екземплярів того або іншого класу об'єктів:

today=new Date();

Тут породжений екземпляр об'єкта Date, що дозволяє здійснювати роботу з датою.

Зверніть увагу на використання крапки з комою наприкінці виразів JavaScript. Її застосування в багатьох випадках необов'язково, але краще ставити.

Зміна значень змінних відбувається за допомогою використання оператора привласнення =.

Namel = "Вираз!";

"Вираз!" може містити в собі інші змінні й оператори дій над ними. Більшість операторів JavaScript подібні до відповідних в C++:

(+) додавання, зчеплення рядків;

(-) вирахування;

(*) множення;

(/) розподіл;

(%) залишок від цілочисельного розподілу;

(++) інкремент;

(--) декремент;

(a+=b) a=a+b;

(a-a-=b) a= a-b;

(&) логічний AND (довгий варіант);

(&&) логічний AND (короткий варіант);

(|) логічний OR (довгий варіант);

(||) логічний OR (короткий варіант);

(!) логічне заперечення;

(>) більше;

(<) менше;

(>>) зрушення праворуч;

(<<) зрушення ліворуч.

Короткий варіант команд порівняння означає те, що аргументи такої операції не будуть обчислюватися далі, якщо в процесі обчислення аргументів результат порівняння стане, відомий заздалегідь.

Наприклад

(2=3) & (A) результат false; вираз A буде обчислено.

(2=3) && (A) результат false; вираз A не буде обчислено.

Математичні функції реалізуються за допомогою бібліотеки Math:

Math.sin() - синус; Math.cos() - косинус; Math.tan() - тангенс; Math.exp() - експонента; Math, log () - натуральний логарифм; Math, random () - випадкове число від 0 до 1.

Завдання **масивів** відбувається за допомогою оператора Array (масиви є об'єктами):

mas = new Array();

Тут заданий масив mas, що має невизначену розмірність.

masl = new Array(5);// Масив masl містить п'ять елементів.

Нумерація елементів масиву починається з нульового, однак при ініціалізації в дужках вказується розмірність масиву, тобто число елементів, під які виділяється пам'ять. При цьому розмірність і тип елементів масиву може мінятися в процесі виконання програми, тобто в мові JavaScript не буде помилки "вихід за межі масиву". При звертанні до неіснуючих елементів (раніше не заданим), їх вміст буде приймати значення "undefined".

Можна задавати початкові значення елементів масиву в процесі його оголошення (тим самим задається розмірність і тип елементів):

mas2 = new Array("A", "B", "C");

Це масив із трьох рядкових елементів.

Звертання до елементів масиву відбувається шляхом вказівки ім'я масиву й індексу елемента. При цьому вказівка індексу елемента більшого, ніж зазначено при оголошенні масиву веде до збільшення розмірності масиву, а не до помилки:

Mas1[3] = 10; mas1[8] = 18; (Тепер mas1 містить дев'ять елементів).

Поточна розмірність масиву втримується у властивості Length: з = mas.length;

JavaScript надає нам також багато методів роботи з масивами. Так, метод Reverse міняє порядок елементів масиву на зворотний:

masl.reverse;

sort - сортує елементи масиву:

mas3 = masl.sort;

Join - поєднує елементи масиву в рядок, використовуючи при цьому зазначений роздільник:

str = mas2.join(" -> "); Рядок str при цьому прийме наступне значення: "A -> B -> C"

Введення/виведення в JavaScript

У JavaScript **виведення даних** на екран може відбуватися різними способами. При цьому оператори виводу оптимізовані для найбільш зручного їх використання. Найбільш простим є застосування оператора **Alert()**. Аргументом оператора може бути будь-який рядковий вираз. Якщо аргумент має нерядковий тип, то він переводиться в рядковий. Результатом виконання оператора Alert є виведення на екран діалогового вікна, вмістом якого є значення виразу аргументу. При цьому діалогове вікно буде очікувати натискання користувачем кнопки ОК. Тільки після виконання цієї дії виконання програми й відображення сторінки буде продовжено. Виведення за допомогою Alert досить зручно використовувати для контролю значень змінних в програмі при налагодженні. Приведемо код HTML-документа, що приводить до появи такого вікна.

Виведення на екран з використанням Alert

```
<HTML>
<ПОЗНАЧКА content="text/html; charset= windows-1251" http~equiv= Content-Type>
<BODY>
<SCRIPT>
mas2 = new Array ("A", "B", "C");
dd = mas2.join ("->");
alert(dd);
</SCRIPT>
</BODY>
</HTML>
```

Слід зазначити, що функція Alert() є методом об'єкта window, що описує поточне вікно браузера. Тому синтаксично більш коректно викликати цю функцію в такий спосіб: window.alert("Текст повідомлення").

Іншим способом виводу інформації на екран є виведення у тіло документа. Організовується він за допомогою оператора **write**, що є методом об'єкта document.

Оператор document.writeln() відрізняється від оператора document.write() тим, що переносить позицію виведення на новий рядок. Виведення тексту відбувається з поточними атрибутами, які мають місце на момент виклику того або іншого оператора виводу. Вираз, що є аргументом оператора виведення, може містити будь-яку рядкову константу, а також містити в собі різні теги HTML. При виводі подібного виразу ці теги будуть інтерпретуватися відповідним чином. Все це дозволяє будувати HTML-код на злету, залежно від тих або інших параметрів.

Іноді при виведенні на екран засобами JavaScript виникають проблеми з кодуванням російського шрифту. Щоб ці проблеми не

турбували вас, необхідно, щоб у налаштуваннях браузера була обрана опція автоматичного визначення кодування документа.

Виведення у тіло документа засобами JavaScript;

```
<HTML>
<ПОЗНАЧКА content="text/html; charset= windows-1251" http-equiv=
Content-Type>
<BODY bgcolor= "#aa88aa" >
<H2> Це виведення засобами HTML <BR>
<SCRIPT>
document.write ("А це працює оператор Write ()");
document.writeln ("</H2> <H3> <FONT color=#fffffff> " +
"<center> Змінюємо стиль шрифту </center> </H3> <BR> ");
</SCRIPT>
      !!!!!!!!!!!!!
</FONT>
<SCRIPT>
document.writeln (" Ми можемо навіть вставити картинку: <BR>");
document.writeln (" <img src= s37b.jpg> ");</SCRIPT>
</BODY>
</HTML>
```

Розглянемо тепер оператори **введення**. JavaScript надає нам кілька способів організації введення. Перший - використання методу **Prompt** об'єкта window. Він має наступний синтаксис:

d=window.promt("Текст повідомлення", "Значення_за_замовчанням");

У результаті виконання такої команди на екрані з'явиться вікно запиту, де буде виведене запрошення на введення, що є у виразі "Текст повідомлення". Після введення значення привласнюється змінній d. Якщо користувач не ввів нічого, то d буде привласнене значення виразу "Значення_за_замовчанням". Це значення буде виведено у вікні запиту й підсвічено так, що, натиснувши кнопку ОК, користувач введе це значення, а, натиснувши будь-яку іншу кнопку, може приступити до введення своєї інформації. Останній вираз є необов'язковим елементом синтаксису оператора Promt.

Введення за допомогою оператора Promt

```
<HTML>
<ПОЗНАЧКА content="text/html; charset= windows-1251" http-equiv=
Content-Type>
<BODY>
<SCRIPT>
var d = "Моя";
c=prompt("Уведіть слово", "Земля");
s =d+" " +C;
alert (s) ;
</SCRIPT>
</BODY>
</HTML>
```

Введення значень булевого типу зручніше за все здійснювати за допомогою оператора window.**confirm**, що має синтаксис:

b = confirm ("Питання");

У результаті виконання такої команди на екрані з'явиться вікно із запитанням і двома кнопками . Залежно від натискання користувачем тої або іншої кнопки змінна b одержить або значення true (кнопка ОК), або false (кнопка Cancel (Скасування)).

Керування потоком обчислень в JavaScript

У спадщину від мови C++ JavaScript дісталися наступні оператори, що реалізують основні алгоритми керування потоком обчислень (flow control): оператор циклу з кінцевим числом повторень, оператор циклу while, оператор розгалуження.

Оператор розгалуження реалізує вибір тої або іншої послідовності дій залежно від умови (умовний оператор):

```
if (умова) { Послідовність 1 }
else             { Послідовність 2  }
```

Вираз "умова" повинно бути булевого типу. Це може бути комбінація операторів відносини або результат дії оператора confirm. Оператор циклу з кінцевим числом повторень повторює певну послідовність дій задану кількість раз.

for ("вираз", "умова", "операція") { Послідовність дій.}

Тут необхідне використання цілочисельної змінної, початковим значенням якої буде "вираз". Цикл буде повторюватися, поки буде дійсною "умова". При цьому при кожній ітерації циклу над змінною - лічильником буде виконуватися дія "операція".

Приклад обчислення суми елементів деякого масиву:

```
3=0
for (i=0; i<5; i++) { s+=mas[i]; }
```

Цикл while виконує деяку послідовність дій доти, поки вірно деяка умова:

```
while ("умова") {  Послідовність дій }
```

Через те, що перевірка умови передує виконанню послідовності дій, цикл while одержав назву циклу із передумовою.

Для примусового виходу з циклів використовується команда break. Для переходу до наступної ітерації циклу (дострокового виконання послідовності дій усередині циклу) використовується оператор continue.

Керування вікнами перегляду

Засоби контролю за відображенням сторінок в JavaScript доповнені командами, що дозволяють керувати як вікнами браузера, так і їх умістом. Команда document.clear очищує поточне вікно. Це може бути при виводі даних у тіло документа. Команда window.close закриває поточне вікно браузера. Команда Window.open(ur1, місце розташування, атрибути) відкриває Документ за адресою url у вікні або фреймі, заданому у виразі "місце розташування". Параметри вікна описані у виразі "Атрибути". Ця команда широко використовується для відкриття у супутньому основному вікні вікон з рекламою.

***Приклади*:**

Занесення у вибране:

```
<a href="#"
onClick="window.external.addFavorite('http://www.ваш_сайт.ua/',
'Назва сайту');return false;">Додати у вибране</a>
```

Лічильник посилань

```
<script language="JavaScript">
<! ---i
var now = new Date()
fixDate(now)
now.setTime(now.getTime() + 365 * 24 * 60 * 60 * 1000)
var visits = getCookie("counter")
```

```
if (!visits)
        visits = 1
else
        visits = parseInt(visits) + 1
setCookie("counter", visits, now)
 Ви були тут " + visits + " раз(а).")
// ---i>
</script>
```

Навігаційне меню

```
<Form><Input Type="hidden" Name="select value">
 <Select Name="sel" Size="1" OnChange="top.location.href =
this.options[this.selectedIndex].value;">
  <Option selected value=#>Посилання</Option>
  <Option value="http://www.ваш_сайт.ua">Посилання 1</Option>
  <Option value="http://www.ваш_сайт.ua">Посилання 2</Option>
  <Option value="http://www.ваш_сайт.ua">Посилання 3</Option>
  <Option value="http://www.ваш_сайт.ua">Посилання 4</Option>
</Select></Form>
```

Організація переходу

```
<SCRIPT> Str = "OK \n "+
"Скасування \n\n "+
"OK->1.html \n "+
"Cancel->2.html ";
if (confitm(str))
{ location.href = "1.html" }
else { location.href = "2.html" } </SCRIPT>
```

Виділення напису. Текст повільно переливається кольорами.

```
<SCRIPT LANGUAGE="JavaScript">
<! ---i Begin
function setupStrobe() {
text = " Плавно миготливий текст";
var x=navigator.appVersion;
y=x.substring(0,4);
if(y>=4)strobeEffect();
}
var isNav=(navigator.appName.indexOf("Netscape")!=-1);
var colors=new Array(
"FFFFFF","FFFFFF","FFFFFF","FFFFFF","FFFFFF","FFFFFF",
"FFFFFF","F9F9F9","F1F1F1","E9E9E9","E1E1E1","D9D9D9",
"D1D1D1","C9C9C9","C1C1C1","B9B9B9","B1B1B1","A9A9A9",
"A1A1A1","999999","919191","898989","818181","797979",
"717171","696969","616161","595959","515151","494949",
"414141","393939","313131","292929","212121","191919",
"111111","090909","000000")
a=0,b=1;
function strobeEffect() {
color=colors[a];
aa="<font color="+color+">" + text + "</font>"
if(isNav) {
document.object1.document.write(aa);
document.object1.document.close();
}
else object1.innerHTML=aa;
a+=b;
if (a==38) b-b-=2;
if (a==0) b+=2;
xx=setTimeout("strobeEffect()",10);
}
// End ---i>
</SCRIPT>
</HEAD>
<BODY  onLoad="setupStrobe()">
<div id="object1"></div>
```

Інформація користувача

```
<font size="2">* Небагато інформації про вас:<br>
```

```
                <script language="JavaScript">
  var name = navigator.appName;
  var vers = navigator.appVersion;
  var code = navigator.appCodeName;
  var where = document.referrer;
  var platform = navigator.platform;
  document.write('<Dd>Броузер: ' + name +
            '<Dd>Версія броузера: ' + vers +
            '<Dd>Кодова назва броузера: ' + code +
            '<Dd>Ви зайшли з: ' + where +
            '<Dd>Платформа: ' + platform);
</script>
```

Дата останнього оновлення сторінки

```
<SCRIPT LANGUAGE="JavaScript">
var m = "Останнє відновлення " + document.lastModified;
var p = m.length-8;
document.writeln("<center>");
document.write(m.substring(p+8, 0));
document.writeln("</center>");
</SCRIPT>
```

Виведення дати

```
<script language="JavaScript">
<! ---i
time=new Date();
month=(time.getMonth() + 1);
date=time.getDate();
year=time.getYear();
if (month < 10) {month = "0" + month }
if (date < 10) {date = "0" + date }
datastr=( date + " / "  + month +" / " + year )
---i>
</script>
</head>
<body>
<center><font face="Arial" size="3" color="#DC5912"><b>
<script language="JavaScript">
<! ---i
document.write(datastr);
---i>
</script>
```

Стартова сторінка

```
<A class=wmenu onclick="this.style.behavior='url(#default#homepage)';
this.setHomePage('http://www.ваш_сайт.ru');return false;"
href="http://newwave.com.ru/#"> <FONT size=2><B>Зробити
стартовою</B><FONT size=+0></A>
Спливаюче вікно
<Script Language="JavaScript">
  Artel=window.open("frame11.htm","Artel",
    "Width=500, Height=160, Toolbar=0, Location=0",
    "Status=0, Menubar=0, Scrollbars=0, Resizable=0")
</Script>
```

Рядок станів, що біжить

```
<Script Language=JavaScript>
  var scrollCounter = 0;
  var scrollText = "Ваш текст";
  var scrollDelay = 70;
  var i = 0;
  while (i ++ < 80)
  scrollText = " " + scrollText;
  function Scroller()
  {window.status = scrollText.substring(scrollCounter++,
  scrollText.length);
    if (scrollCounter == scrollText.length)
       scrollCounter = 0;
```

```
        setTimeout("Scroller()",
        scrollDelay);}
  Scroller();
</Script>
Показ випадкового банера
<Html>
<Head>
<Script Language="JavaScript">
var imagesarray = new Array(
 "banner1.jpg",
 "banner2.jpg",
 "banner3.jpg");
var commentsarray = new Array(
 "Alt - banner1",
 "Alt - banner2",
 "Alt - banner3");
</Script>
</Head>
<Body>
<Script Language="JavaScript">
  var los = Math.floor(Math.random() * imagesarray.length)
  document.write ("<Img Src='"+imagesarray[los]+"'
Alt='"+commentsarray[los]+"'>");
</Script>
</Body>
</Html>
```

Завдання:

На основі HTML сторінок лабораторних робіт №1,№2, реалізувати JavaScript-код об'єкта.

1. Реалізуйте скрипт, що випадковим чином виводить інформацію дня на екран (цитата, анекдот і.т.п.). Інформацію помістіть в масив і винесіть в окремий файл скрипта (*.js).
2. Використовуючи оброблювачі подій OnMouseOver і OnMouseOut забезпечте збільшення зображення при наведенні на нього курсору й повернення від збільшеної копії до зменшеного після зняття курсору.
3. Реалізуйте багатомовний інтерфейс сайту. (Наприклад, створіть кілька копій HTML сторінок, створених у попередніх лабораторних роботах на різних іноземних мовах.) Напишіть скрипт, що дозволяє користувачам вибирати для переглядів сторінку на одній з мов (мінімальний набір - UA, EN).
4. Створіть скрипт, що буде містити обчислення й виведення на екран основних математичних операцій (+,-,/,*). Для виводу різниці й суми треба скористатися методом write об'єкта document, для виводу ділення й множення - методом alert об'єкта window.
5. Створіть скрипт, що буде містити поле типу "radio" (мінімально 5 варіантів вибору). Забезпечити, залежно від вибору поля, виведення на екран одного з п'яти зображень. Забезпечити повернення від зображення до вихідного HTML-документу, з якого був здійснений виклик зображення. Сформуйте збільшену, зменшену копію того самого зображення (використовуйте атрибути: heigth і width тегу <img>).
6. Створіть статичне меню переходу на інші сторінки. Використовуючи оброблювачі подій, треба передбачити зміну кольору гіперпосилань при наведенні на них курсору.

7. Реалізуйте меню переходу на основі елемента вводу select. В HTML-документі оголосіть форму з одного елемента select. Події onChange цього елемента привласніть користувальницьку функцію, що здійснює перехід на запитувану користувачем сторінку.

Контрольні питання:

1) Що представляє собою JavaScript?
2) Поняття об'єктної моделі JavaScript (властивості, методи, події).
3) Які є способи розміщення коду JavaScript на HTML-сторінці?
4) За допомогою якого оператора здійснюється оголошення масивів?
5) Яким методом можна здійснити очищення поточного вікна?
6) Що виконує команда window.close?
7) Що представляють собою оброблювачі подій?
8) Що представляє собою ієрархія класів?
9) Що можна виконати за допомогою методу alert об'єкта window?

Лабораторна робота №4

Тема: Застосування JavaScript при створенні Web form

Мета: Познайомитися із синтаксисом, основними елементами мови JavaScript

Знати: Структуру HTML документа, основні елементи мови JavaScript

Вміти: Компонувати текстову інформацію при створенні гіпертекстових документів з використанням простих текстових редакторів

Теоретичні відомості

Об'єкт form

Сучасні веб-застосунки використовують клієнт-серверну архітектуру, яка за останні десятиліття стала одним із основних способів організації розподілених програмних систем [7]. При такій організації клієнтом виступає веб-браузер, а у якості серверного за стосунку виступає веб-сервер. Задачею веб-браузера є візуалізація HTML-сторінок та мультимедійних об‘єктів, з яких вони складаються. В свою чергу веб-сервер виконує роль оброблювача запитів від віддалених веб-клієнтів.

Обмін інформацією відбувається із використанням методів HTTP: GET, POST, PUT, HEAD, TRACE, OPTIONS, CONNECT, DELETE.

Веб-форма — це набір текстових полів, списків, кнопок [10] та інших елементів керування, що активуються клацанням миші, за допомогою яких відвідувач сторінки може надати їй той чи інший вид інформації. Форми в Інтернеті всюди — завдяки формам ми можемо створювати облікові записи електронної пошти, переглядати та купувати товари в інтернет-магазинах, здійснювати фінансові транзакції та багато іншого. Найпростіша форма - це єдине текстове поле з пошукових систем, таких як Google.

Елемент HTML form (<form>) являє собою розділ документа, що містить інтерактивні елементи керування, які дозволяють користувачеві відправляти інформацію на веб-сервер [5].

Створення кількох форм [6]. Форми в документі входять до спеціальної колекції document.forms [8]. Кожна форма буде мати своє власне ім'я (для цього використовується значення атрибута name елемента <form>). Щоб одержати доступ до елементів форми потрібно: спочатку даємо ім'я (name="MyForm"), потім одержуємо доступ до будь-яких елементів:

```
var name = document.myForm.custName.value
```

Відомо, що в цього об'єкта є властивість forms, що представляє собою масив форм, який міститься в поточному документі. Наприклад, якщо припустити, що форма myForm з попереднього приклада є першою на сторінці, то до неї можна одержати доступ у такий спосіб:

```
var name = document.forms[0].custName.value
```

Іноді буває необхідно організувати автоматичний доступ до форм. Наприклад, такий спосіб:

```
for (x=0; x<3; x=x+1) {var formName[x] = document.forms[x] name}
```

За допомогою такого циклу for можна всі імена форм записати в масив, тим самим організувавши незалежний доступ до кожної з них (formName[x]).

Форма може містити різні елементи введення HTML: input, textarea, button, select і т.д. Правильно спроектована форма поділяється на логічні фрагменти за допомогою елементу <fieldset>.

Всі об'єкти форми, незалежно від того, яким чином здійснюється доступ до них, мають свої набори властивостей, як і документ, вікно. Властивості об'єкта form, зокрема, що випливають:

- *action.* (URL. По суті, такий, як атрибут action елемента <form>);
- *method.* (такий, як атрибут method елемента <form>);
- *name.* (ім'я форми (такий, як атрибут name елемента <form>));
- *length.* (число елементів input, textarea й select у формі);
- *target.* (цільове вікно або фрейм);
- *elements.* (масив, у якому містяться всі елементи input, textarea і select).

Об'єкт form має, свої методи, серед яких reset( ), submit ().

Фокус. Форма призначена для введення інформації. На жаль, користувачі робити це не зможуть доти, доки не клацнуть мишею по першому полю або виділять його за допомогою клавіші <Tab>, встановивши таким чином фокус на цьому полі. Користувач може допомогти в цьому, встановивши фокус на потрібному початковому полі автоматично за допомогою JS методу focus() необхідного елемента <input>. Але цей підхід вимагає зайвого рядка коду і може викликати дратівливі проблеми.

Краще застосувати HTML5-атрибут autofocus, який можна вставити в елемент <input> або <textarea> (але тільки один елемент форми), наприклад: <input id="name" placeholder="Мороз Тарас" autofocus>.

Обробка помилок на формі

Автори HTML5 розробили систему перевірки на стороні клієнта [9], яка дозволяє вставляти основні правила валідації в будь-яке поле <input>, наприклад: певне поле не можна залишати порожнім, і відвідувач має запровадити у нього хоч щось. У HTML5 це здійснюється за допомогою атрибуту required у відповідному полі:

<input id="name" placeholder=" Мороз Тарас " autofocus required>

Для того,щоб звернути увагу користувача на цю вимогу за допомогою якихось своїх візуальних ознак можна, наприклад, виділити рамку поля кольором і поставити біля нього зірочку.

Найпотужнішим (і найскладнішим) підтримуваним HTML5 типом перевірки є перевірка на основі регулярних виразів [5].

Регулярне вираз — це шаблон для зіставлення із зразком, закодований згідно з певними синтаксичними правилами. Регулярні вирази використовуються для пошуку в тексті рядків, які відповідають певному шаблону. Наприклад, за допомогою регулярного виразу можна перевірити,

що поштовий індекс містить правильну кількість цифр, або в адресі електронної пошти є знак @, а його доменне розширення містить, принаймні, два символи.

JavaScript і його обробники подій ідеально підходять для перевірки даних [6], що вводяться користувачами у формах. Під час набору даних скрипт може непомітно перевіряти коректність кодування, числа символів і т.д.

Наприклад, у лістингу закодована сторінка, що дозволяє її відвідувачеві вводити дані. Скрипт здійснює перевірку правильності вводу поштового індексу.

Лістинг *Перевірка даних у формах за допомогою JavaScript*

```
<html>
<head>
<title>Checking the Zip</title>
<meta http-equiv=" Content-Script-Type" content="text/javascript">
<script>
<! ---i
function zipCheck()
{
 var zipStr = custForm.zipCode.value;
 if (zipStr == "") {
   alert("Please enter a five digit number for your Zip code");
   return(-1);
   }
 if (zipStr.length != 5) {
   alert ("Your Zip code entry should be 5 digits");
   return(-1);
   }
 return(0);
}
function checkSend()
 {
 var passCheck = zipCheck();
 if (passCheck == -1) {
   return;
 }
 else {
   custForm.submit();
 }
}
// end hiding ---i>
</script>
</head>
<body>
<h1>Please fill out the following form:</h1>
<form action=" cgi-bin/address.pl" method="post" name="custForm">
<pre>
Name:  <input type="text" size="20" name="name">
Address: <input type="text" size="50" name="address">
City:  <input type="text" size="30" name="city">
State:  <input type="text" size="2" name="state">
Zip:   <input type="text" size="5" NAME="zipCode"
     onBlur = "zipCheck()">
Email:  <input type="text" size="40" Name="email">

<input type="button" value="Send It" onClick = "checkSend()">
</form>
</body>
</html>
```

Обробка форми починається при виникненні події onBlur, при втраті фокуса поля вводу індексу, або при натисканні клавіші Tab, або за

допомогою миші. Оброблювач запускає функцію zipCheck, що перевіряє уведений поштовий індекс на коректність. Якщо поле залишилося порожнім, видається віконце з нагадуванням про те, що це поле варто заповнити. Це робиться шляхом перевірки значення рядкової змінної:

```
if (zipStr = "") {
```

Якщо введено більше або менше 6 цифр, також видається попередження.

```
if (zipStr.length != 5) {
```

Якщо користувач зробив все правильно, триває нормальна робота. Користувач може проігнорувати попередження або змінити його, але так, що воно залишиться некоректним. Для цього випадку передбачений повторний виклик zipCheck з функції checkSend:

```
var passCheck = zipCheck( ):
```

І якщо індекс як і раніше залишився неправильним, функція zipCheck поверне значення -1, що перевіряється за допомогою checkSend:

```
if (passCheck = -1) {  return;  }
```

Тому, якщо й цього разу індекс введено невірно, то функція checkSend поверне користувача на продовження заповнення форми. Загалом, вийшов багато східчастий захист від неправильного введення даних. Якщо користувач нарешті набрав на клавіатурі шість цифр, форма підтверджується.

Перевірка могла б бути як завгодно складною. Наприклад, можна заборонити використання буквених символів у цьому полі. Це можна зробити, застосувавши метод charAt( ) об'єкта String, що ретельно перевіряє кожну позицію й встановлює, чи дійсно в ній перебуває цифра від 0 до 9.

Розширена функція zipCheck( ) може виглядати так:

```
function zipCheck()
{
 var zipStr = custForm.zipCode.value;

 if (zipStr == "")
   {
   alert("Please enter a five digit number for your Zip code");
   return(-1);
   }
 if (zipStr.length != 6)
   {
   alert ("Your Zip code entry should be 6 digits");
   return(-1);
   }

 for (x=0; x < 6; x++)
 {
   if ((zipStr.charAt(x) < "0") || (zipStr.charAt(x) > "9"))
   {
     alert("All characters in the Zip code should be numbers.");
     return(-1);
   }
  }
 return(0);
}
```

Перевірка починається із циклу for, щоб програма могла зробити необхідну кількість ітерацій, охопивши всю довжину поля вводу:

```
for (x=0; x < 5; x++) {
```

Використовуючи цикл for, можна послідовно перебрати всі значення масиву zipStr з номерами від 0 до 5.

У циклі перебуває умовний вираз if, він і перевіряє, чи є символи цифровими:

```
if ((zipStr.charAt(x) < "0") || (zipStr.charAt(x) > "9")) {
```

Знак || означає логічне «або». Тобто, якщо виконується хоча б одна з умов, то весь вираз приймає значення «істина», і з'являється віконце з попередженням:

```
alert("Індекс повинен складатися тільки з цифр!"); return(-1);
```

Якщо ж вираз if приймає значення «неправда», то команди, що знаходяться всередині, пропускаються, і передбачається, що користувачеві вдалося ввести все правильно.

JavaScript на клієнтській машині

Одною з переваг JavaScript є те, що ця мова вбудовується прямо у веб-сторінку й вам не потрібно піклуватися про CGI-скрипти. У наступному лістингу показаний приклад електронної форми та її обробного механізму.

***Лістинг** Customer Survey Form*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml 1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Customer Survey Form</title>
<script>
<! ---i
function processform() {
   var newline = "\n";
   var result_str = "";
   var Form1 = document.form1;
   result_str += Form1.first_name.value + " " + Form1.last_name.value +
newline;
   result_str += Form1.email.value + newline;

   for (x=0; x<Form1.where.length; x++) {
     if (Form1.where[x].checked) {
     result_str += Form1.where[x].value + newline;
     break;
     }
   }
   if (Form1.desktop.checked) result_str += "desktop computers" + newline;
   if (Form1.notebook.checked) result_str += "notebook computers" +
newline;
   if (Form1.peripherals.checked) result_str += "peripherals" + newline;
   if (Form1.software.checked) result_str += "software" + newline;
   document.form2.results.value = result_str;
    return;
   }
// ---i>
</script>
</head>
<body>
<h1>Web Site Survey</h1>
<form name="form1" id="form1">
<table cellpadding="5">
<tr>
<td>First name:</td> <td><input type="text" name="first_name" size="40"
maxlength="40" /></td>
</tr>
<tr>
```

```
<td>Last name:</td> <td><input type="text" name="last_name" size="40"
maxlength="40" /></td>
</tr>
<tr>
<td> E-mail address:</td> <td><input type="text" name="email" size="40"
maxlength="40" /></td>
</tr>
</table>
<p>
<b>Where you heard about us:</b></p>
<input type="radio" name="where" value="Web" checked="checked">Web Search
or Link</input><br />
<input type="radio" name="where" value="Advertisement">Radio or TV
Ad</input><br />
<input type="radio" name="where" value="Press Mention">Article or press
mention</input><br />
<input type="radio" name="where" value="Other">Other</input><br />
</p>
<p>
<b>What products would you like more information about? (check all that
apply)</b><br />
<input type="checkbox" name="desktop"> desktop computers
<input type="checkbox" name="notebook"> notebook computers
<input type="checkbox" name="peripherals"> peripherals
<input type="checkbox" name="software"> software
</p>
<button name="submit" type="button" onclick="processform()">
<span style=" font-family: Arial, Helvetica; font-variant: small-caps;
font-size: 12pt"> Submit Survey</span>
</button>
<button name="reset" type="reset">
<span style=" font-family: Arial, Helvetica; font-variant: small-caps;
font-size: 12pt"> Clear Page</span>
</button>
</form>
<hr/>
<form name="form2" id="form2" action="mailto:survey@fakecorp.net">
<p>Check the entries below for accuracy. If they're accurate, then enter a
comment on the last line (if desired) and click Send It to send the form
via e-mail.</p>
<textarea name="results" cols="40" rows="10">
</textarea>
<button name="submit" type="submit">
<span style=" font-family: Arial, Helvetica; font-variant: small-caps;
font-size: 12pt"> Send It!</span>
</button>
</form>
</body>
</html>
```

Скрипт бере значення, введені користувачем у першій формі, і поміщає їх у текстовому вигляді в іншу форму. При натисканні на кнопку SEND IT! (Відправити!) вся накопичена інформація відправляється електронною поштою.

Цей приклад показує, як важливо одержувати коректні значення прапорців і перемикачів. У лістингу демонструються два способи, якими це можна зробити. Для початку про перемикачі. Потрібно визначити, яке положення селективної кнопки обрано, тоді ми зможемо працювати з відповідними даними.

Значення перемикачів зберігаються в спеціальних масивах, а отже, можливо одержати доступ до них за допомогою циклу:

```
for (x = 0; x < Form1.where.length; x++) {
     if (Form1.where[x].checked) {
```

```
                    result_str += Form1.where[x].value + newline;break;
        }
}
```

В свою чергу, всі перемикачі теж зберігаються в масиві об'єктів, імена яких визначаються за допомогою атрибута name, що вказується при їх створенні. У нашому прикладі name="where". З кожним екземпляром where можна використовувати властивість checked для того, щоб визначити, у якому стані перебуває перемикач. Коли ми знайдемо той об'єкт where, властивість checked якого дорівнює true, це буде означати, що саме його value нам і потрібно.

Тому фактично наведений раніше цикл займається визначенням стану перемикача. Коли він його знаходить, цикл for завершується (командою break), а зі значенням стаються певні дії. У цьому випадку воно додається до рядка result_str.

Друге, що в прикладі, - це робота з наборами прапорців. І тут теж потрібно перевіряти стан властивості checked. Але, оскільки кожний прапорець має своє власне ім'я (name), то немає необхідності зберігати їх у масиві або використовувати цикл для їх перебору. До кожного прапорця потрібен підхід, кожен з них особисто потрібно перевірити на checked. Якщо це так, то використовується його значення:

```
   if (Form1.desktop.checked)        result_str += "desktop computers" +
newline;
   if (Form1.notebook.checked)      result_str += "notebook computers" +
newline;
   if (Form1.peripherals.checked) result_str += "peripherals" + newline;
   if (Form1.software.checked)    result_str += "software" + newline;
```

Метод addEventListener дозволяє призначити на елементи обробники подій. З його допомогою можна вказати, наприклад, що робити при натисканні кнопки, або що робити при наборі тексту в текстовому полі, наприклад:

```
button.addEventListener('click', function() { alert('message'); });
```

Протокол Mailto призначений для відправлення листів через поштову систему користувача за схемою: mailto:<адреса>?<параметри>.

Завдання:

На основі HTML сторінок лабораторних робіт №1, №2, №3 і на основі прикладу в теоретичних відомостях (Customer Survey Form). Створити дві скрипт-форми.

1 форма.

Форма опитування відвідувачів сайту (згідно тематики лабораторних робіт) з перевіркою даних, що вводяться, на стороні клієнта. Містять активні теги: Radio, Checkbox, Text Area, а також кнопку підтвердження введення даних і кнопку очищення полів введення даних. При натисканні на кнопку форми реалізувати проходження перевірки введених даних. При позитивному результаті помістити результати в текстове поле 2 форми.

2 форма.

Форма повідомлення користувача. Має активний тег textarea і кнопку підтвердження відправлення текстової інформації. При натисканні на

кнопку форми, сформувати лист поштовому клієнтові операційної системи.

Контрольні питання:

1) Що представляє собою об’єкт форма?
2) Що представляє собою властивість forms об’єкта form?
3) Як здійснювати перевірку введених даних у формах?
4) Які елементи можуть бути присутні на формах?
5) Як встановити фокус у формі?
6) Як перевірити, чи коректно користувач ввів інформацію у форму?
7) Що таке регулярний вираз?
8) Що представляють собою елементи <input> форми?
9) Які дії необхідні щоб визначити, у якому стані перебуває перемикач?
10) Що визначається за допомогою атрибута type елемента <input> форми?
11) Що означає type="radio" елемента <input> форми?

Лабораторна робота №5

Тема: Вивчення й практичне застосування форм HTML та елементів керування.

Мета: Навчитися передавати дані серверу за допомогою форм.

Знати: Методи передачі даних. Елементи керування форм. Роботу механізму регулярних виразів.

Вміти: Застосовувати для передачі інформації серверу всі типи елементів керування формою. Застосовувати плагін jQuery Validation.

Теоретичні відомості:

Способи передачі параметрів сценарію

Найпоширенішими методами передачі даних між браузером і сценарієм є GET і POST. Однак вручну задавати рядок параметрів для сценаріїв і URL-кодувати їх утомливо [6].

Розглянемо метод GET для передачі запитів серверу.

Існують зручні можливості мови HTML, які, підтримуються браузерами.

Отже, нехай на сервері в кореневому каталозі розміщений файл сценарію script.php. Цей сценарій розпізнає 2 параметри: name і age. Де ці параметри задаються, ми поки не вирішили. При переході за адресою `http://www.somehost.com/script.php` він повинен відробити й вивести наступну HTML-Сторінку:

```
<html><body>
      Привіт, name! Вам age років!
</body></html>
```

При генерації сторінки потрібно name і age замінити на відповідні значення, передані в параметрах.

Передача параметрів через адресний рядок браузера

Спробуємо включати параметри прямо в URL, у рядок параметрів. Таким чином, якщо запустити в браузері `http://www.somehost.com/script.cgi?name=Vasily&age=20` ми одержимо сторінку з потрібним результатом:

```
<html><body>
      Привіт, Vasily! Я знаю, Вам 20 років!
</body></html>
```

Зверніть увагу, що розділяються параметри символом **&**, а також використовується знак рівності **=**.

Використання HTML-Форм

Необхідно зробити, щоб користувач міг у зручній формі ввести своє ім'я й вік. Необхідно інтерактивне вікно, у яке здійснюється введення наших параметрів, при чому через браузер.

Отже, знадобиться звичайний HTML-документ (наприклад, з ім'ям form.html і розташований у кореневому каталозі) з елементами діалогу - полями введення тексту й кнопкою, при натисканні на яку запуститься скрипт script.php. Текст документа form.html:

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

```
  <form action="script.php">
      Введіть ім'я: <input type=text name="name"><br>
      Введіть вік: <input type=text name="age"><br>
      <input type=submit value="GO!">
</form>
</body></html>
```

Завантажимо наш документ у браузер. Тепер, якщо ввести в поле з ім'ям своє ім'я, а в поле для віку — свій вік і натиснути кнопку, браузер автоматично звернеться до сценарію hello.php і передасть через ? всі атрибути, розташовані в тегах *<input>* у формі й розділені символом & у рядку параметрів. Помітьте, що в атрибуті action тега *<form>* задається відносний шлях, тобто сценарій hello.php буде шукатися браузером у тому ж самому каталозі, що й файл form.html.

Всі перекодування й перетворення, які потрібні для URL-кодування даних, здійснюються браузером автоматично.

Використання форм дозволяє не навантажувати користувача такою інформацією, як ім'я сценарію, його параметри й т.д. Він завжди буде мати справу тільки з полями, перемикачами й кнопками форми.

Залишилось визначитись, як можна витягти $name і $age з рядка параметрів і обробити їх.

Обробка параметрів запитів

Веб-програмування в більшій частині являє собою саме обробку різних даних, введених користувачем — тобто, обробку HTML-форм.

PHP набагато полегшує завдання обробки й розбору зовнішніх змінних, тобто змінних, які надійшли з HTML-форм (із браузера користувача). В мову PHP вбудовані всі необхідні можливості, щоб не прийшлось замислюватися над особливостями протоколу HTTP і міркувати, як відбувається відправлення й прийом POST-форм або навіть завантаження файлів. В PHP все передбачено.

А тепер спробуємо написати сценарій, що приймає в параметрах ім'я користувача та його вік і виводить:

"Привіт, <ім'я>! Вам <вік> років!"

Тобто, нам потрібно передати в скрипт 2 параметри: name і age.

Тепер напишемо скрипт script.php, що приймає два параметри: name і age, а також HTML-документ із формою, що ці два параметри буде передавати в новий скрипт:

```
<?php
      echo "Привіт, $_GET['name'] ! Вам $_GET['age'] років !";
 ?>
```

А ось HTML-документ send.html, за допомогою якого параметри name і age передаються нашому скрипту:

```
<html><body>
 <form action="script.php">
  Введіть ім'я: <input type=text name="name"><br>
  Введіть вік: <input type=text name="age"><br>
  <input type=submit value="GO!">
 </form>
 </body></html>
```

Тепер наш скрипт приймає два параметри name і age і виводить у браузер результат формату: "Привіт, <ім'я>! Вам <вік> років!".

Зверніть увагу на адресний рядок браузера після передачі параметрів сценарію, він буде виглядати приблизно в такий спосіб (без URL-Кодування кирилиці):

```
http://localhost/script.php?name=Сашко&age=23
```

Залежно від установок вашого інтерпретатора, існує кілька способів доступу до даних з ваших HTML-форм. Наприклад:

```
<?php
// Доступно, починаючи з PHP 4.1.0
 echo $_GET['username'];
echo $_POST['username'];
echo $_REQUEST['username'];
import_request_variables('p', 'p_');
echo $p_username;
//Доступно, починаючи з PHP 3. Починаючи з PHP 5.0.0, ці довгі
//визначені змінні можуть бути відключені директивою
//register_long_arrays.
echo $HTTP_GET_VARS['username'];
// Доступно, якщо директива PHP register_globals = on. Починаючи
// з PHP 4.2.0, значення за замовчуванням register_globals = off.
//Використання/довіра цьому методу непереважна.
echo $username;
?>
```

Елементи HTML форм

Форма в HTML-документі реалізується тегом-контейнером FORM, у якому задаються всі керуючі елементи - поля введення, кнопки й т.д. Якщо керуючі елементи зазначені поза вмістом тегу FORM, то вони не створюють форму, а використовуються для побудови користувальницького інтерфейсу на веб-сторінці, тобто для привнесення в неї різних кнопок, прапорців, полів введення.

Обробка елементів форми виробляється за допомогою скриптів, але вони можуть і взагалі ніяк не оброблятися.

Імена елементам форми привласнюються через їх атрибут NAME.

Кожний елемент форми може мати початкове й кінцеве значення, які є символьними рядками. Початкові значення елементів не змінюються, завдяки чому може здійснюватися скидання значень, зазначених користувачем. Результатом цієї дії буде встановлення всіх керуючих елементів форми у свої первісні, що використовуються за замовчуванням значення.

В HTML визначені наступні типи керуючих елементів:

- Кнопки - задаються за допомогою елементів BUTTON і INPUT. Розрізняють:

- кнопки відправлення - при натисканні на них, вони здійснюють відправлення форми серверу;

- кнопки скидання - при натисканні на них, елементи керування приймають первісні значення;

- інші кнопки - кнопки, для яких не зазначена дія, що виконується за замовчуванням при натисканні на них.

- Залежні перемикачі (перемикачі із залежною фіксацією) - задаються елементом INPUT і являють собою перемикачі "вкл/викл". Якщо кілька залежних перемикачів мають однакові імена, то вони є взаємовиключними (якщо один з них набуває стану "вкл", то всі інші автоматично - "викл". Саме це і є перевагою їх використання.

- Незалежні перемикачі (перемикачі з незалежною фіксацією) - задаються елементом INPUT і являють собою перемикачі "вкл/викл", але на відміну від залежних, незалежні перемикачі можуть приймати й змінювати своє значення незалежно від інших перемикачів. Навіть якщо останні мають таке ж ім'я.
- Меню - реалізується за допомогою елементів SELECT, OPTGROUP і OPTION. Меню надають користувачеві список можливих варіантів вибору.
- Введення тексту - реалізується елементами INPUT, якщо вводиться один рядок, і елементами TEXTAREA - якщо кілька рядків. В обох випадках введений текст стає поточним значенням керуючого елемента.
- Вибір файлів - дозволяє разом з формою відправляти обрані файли, реалізується HTML-елементом INPUT.
- Сховані керуючі елементи - створюються керуючим елементом INPUT.

Дуже багато елементів задаються за допомогою універсального тегу INPUT.

Тег FORM - контейнер форм

Як уже було сказано, форма в HTML-документі реалізується тегом-контейнером FORM. Цей тег своїми атрибутами вказує адресу сценарію (скрипта), якому буде послана форма, спосіб пересилання й характеристику даних, що утримуються у формі.

Атрибути тегу FORM:

- action- єдиний обов'язковий атрибут. У якості його значення вказується URL-адреса запитуваного скрипта, що буде обробляти дані у формі. Припустимо використовувати запис mailto:URL, завдяки якому форма буде послана по електронній пошті. Якщо атрибут ACTION все-таки не зазначений, то вміст форми буде відправлено на URL-адресу, з якої завантажувалася веб-сторінка;
- method - визначає метод HTTP, який використовується для пересилання даних форми від браузера до сервера. Атрибут METHOD може приймати два значення: GET і POST;
- enctype - необов'язковий атрибут. Вказує тип умісту форми, що використовується для визначення формату кодування при її пересиланні. В HTML визначенні два можливих значення для атрибутів ENCTYPE:

- APPLICATION/ WWW-FORM-URLENCODED (використовується за замовчуванням);

- MULTIPART/ FORM-DATA.

Тег INPUT і його методи

Елемент INPUT є найбільш вживаним тегом HTML - форм. За допомогою цього тегу реалізуються основні функції форми. Він дозволяє створювати всередині форми поля введення рядка тексту, ім'я файлу, пароля й.т.ін. Особливість INPUT - у нього немає кінцевого

(завершального) тегу. Атрибути й особливості використання INPUT залежать від способу його застосування. Розглянемо ці способи.

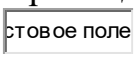
Однорядкові поля введення

Найбільш часто використовуються поля введення - адже навіть кнопка є полем введення інформації. Почнемо з поля введення текстової інформації. Формат тегу INPUT для створення поля введення текстового рядка:

```
<input type=text name=ім'я_параметра [value=значення] [size=розмір_поля] [maxlen=довжина_поля]>
```

Даний тег створює поле ввода з максимально припустимою довжиною тексту maxlen і розміром в size знакомісць. Якщо зазначено атрибут value, то в поле буде постійно відображатися значення даного атрибута. У квадратних дужках [] позначені необов'язкові атрибути.

Приклад однорядкового поля введення:



Поля введення пароля

Ім'я користувача можна ввести за допомогою звичайного текстового поля. А пароль не повинен відображатися на екрані при його введенні, тому для нього - поле введення пароля:

```
<input type=password name=ім'я_параметра [value=значення] [size=розмір] [maxlen=довжина]>
```

Принцип роботи такий же, як і текстового поля. Різниця полягає в тому, що вводиться інформація, що, у полі не відображається, а заміняється зірочками. Не рекомендується встановлювати значення за замовчуванням з міркувань безпеки (value).

От приклад поля введення пароля:

Сховане текстове поле

Для передачі службової інформації (про яку користувач навіть не повинен підозрювати) використовуються сховані поля. За допомогою них, наприклад, передаються параметри налаштування.

```
<input type=hidden name=ім'я_параметра value=значення>
```

Такі поля передаються серверу, але на веб-сторінці не відображаються.

Незалежні перемикачі

Часто користувачеві, що заповнює форму в браузері, необхідно дати можливість вказати свої налаштування за допомогою вибору певних значень, а поруч із ними міститься невелике квадратне поле, у якому можна встановити, або забрати галочку. При цьому значення, відповідно, буде або обрано, або не обрано.

Реалізувати це можна знову ж за допомогою тегу INPUT. Для цього тільки необхідно як значення атрибута type вказати chechbox.

```
<input type=checkbox name=ім'я_параметра value=значення [checked]>
```

Якщо перемикач був включений на момент натискання кнопки відправлення даних, то скрипту буде переданий параметр ім'я=значення.

Якщо ж прапорець виключений, то сценарію взагалі нічого не буде передано - начебто перемикача взагалі немає.

Перемикач за замовчуванням або включений, або виключений. Щоб перемикач був за замовчуванням включений, необхідно для нього вказати атрибут checked.

Перемикач checkbox називається незалежним, тому що його стан не залежить від стану інших перемикачів checkbox. Таким чином, в одній формі може бути одночасно обрано кілька перемикачів.

Приведемо приклад незалежних перемикачів:

- ☐ Перша опція
- ☐ Друга опція
- ☐ Третя опція
- ☐ Четверта опція

Залежні перемикачі

Залежний перемикач, так само як і незалежний перемикач, може бути або включений, або виключений. При цьому перемикач radio є залежним перемикачем, оскільки на формі може бути тільки один включений перемикач типу radio. Точніше, якщо у формі присутні кілька однойменних залежних перемикачів, то включений з них може бути тільки один. При виборі одного перемикача всі однойменні залежні перемикачі автоматично вимикаються. Як ім'я перемикачів сприймається значення атрибута name. Може бути тільки один активний перемикач. Приклад лістингу форми із залежними перемикачами:

```
<form action="http://localhost/script.php" method="GET">
      <input type=radio name=answer value=yes checked>Так
      <input type=radio name=answer value=no>Немає
      <input type=submit value=Відправка>
</form>
```

Ця форма буде виглядати так:

◉ Так ○ Ні [відправка]

Перший перемикач (зі значенням так) активний за замовчуванням (ми встановили атрибут checked).

Як тільки користувач натисне кнопку "Відправка", скрипту script.php буде переданий параметр answer (атрибут name обох перемикачів) зі значенням так або ні (залежно від того, який варіант вибрав користувач).

Кнопка відправлення форми

Ще одним елементом управління типу INPUT є кнопки. Кнопка відправлення служить для відправлення скрипту введених у форму параметрів. Синтаксис тегу INPUT при цьому такий:

```
<input type=submit [name=go] value=Відправити>
```

Атрибут value визначає текст, що буде написаний на кнопці відправлення. Атрибут name визначає ім'я кнопки і є необов'язковим. Якщо значення цього атрибута не вказувати, то скрипту будуть передані введені у форму значення і все. Якщо атрибут name для кнопки буде зазначений,

то додатково до основних даних форми буде відправлена пара ім'я=значення від самої кнопки.

Кнопка скидання параметрів

Крім кнопки submit є ще кнопка reset, що скидає параметри форми, а точніше, налаштовує для всіх елементів форми значення за замовчуванням. Бажано, щоб на формі була така кнопка, особливо, якщо це велика форма. Наявність даної кнопки забезпечує очищення форми, наприклад, у випадку, коли були введені неправильні параметри. Синтаксис кнопки скидання:

```
<input type=reset value=Скидання>
```

Кнопка відправлення з малюнком

Замість кнопки submit можна використовувати малюнок для відправлення даних. Натиск на малюнку дає те ж саме, що й натискання на кнопку submit. Однак, крім цього, сценарію будуть передані координати місця натиску на малюнку у форматі ім'я.x=коор_X, y=коор_Y. Синтаксис кнопки відправлення з малюнком:

```
<input type=image name=ім'я src=малюнок>
```

Багаторядкові текстові поля. Тег TEXTAREA

В HTML Багаторядкові текстові поля створюються за допомогою тегу TEXTAREA. Поле, створюване цим тегом, дозволяє вводити й відправляти не один рядок, а відразу кілька. Синтаксис тегу TEXTAREA:

```
<textarea name=ім'я [cols=ширина_в_символах] [rows=висота_в_символах] wrap=тип_переносу>
текст за замовчуванням
</textarea>
```

Необов'язкові параметри cols і rows бажано все-таки вказувати. Перший з них задає кількість символів у рядку, а другий - кількість рядків в області. Атрибут wrap визначає тип переносу тексту, як буде виглядати текст у полі введення:

- Virtual - праворуч від текстового поля виводиться смуга прокручування. Текст, що вводиться, виглядає розбитим на рядки, а символ нового рядка вставляється при натисканні ENTER;
- Physical - цей тип залежить від типу браузера й виглядає по-різному;
- None - текст виглядає в полі в тому вигляді, у якому користувач його вводить. Якщо текст не вміщується в один рядок, з'являється горизонтальна смуга прокручування.

Варто помітити, що найбільш зручним є тип Virtual.

Списки вибору. Тег SELECT

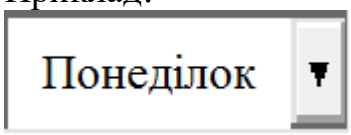
Списки з єдиним вибором

Досить часто існує необхідність представити дані у вигляді списку й передбачити можливість вибору в цьому списку. В HTML списки реалізуються за допомогою тегу SELECT. Список вибору дозволяє вибрати один варіант із безлічі. Приклад списку з єдиним вибором:

```
<select name=day size=1>
```

```
        <option value=1>Понеділок</option>
        <option value=2>Вівторок</option>
        <option value=3 selected>Середа</option>
        <option value=4>Четвер</option>
        <option value=5>П'ятниця</option>
        <option value=6>Субота</option>
        <option value=7>Неділя</option>
</select>
```

Приклад:



Атрибут name визначає ім'я параметра, що буде переданий скрипту. Якщо атрибут size дорівнює 1, то список буде "оснащений" смугою прокручування. Значення, обране в списку за замовчуванням, можна вказати за допомогою атрибута selected для відповідного тегу option. У наведеному прикладі день тижня за замовчуванням - середовище. Атрибут value є необов'язковим. Якщо його не вказати, то буде переданий рядок, укладений в тег option.

Списки множинного вибору

За допомогою тегу SELECT можна також створювати списки множинного вибору. У таких списках можна вибрати не одне, а відразу кілька варіантів значень. Для того, щоб створити список із множинним вибором, необхідно для тегу SELECT вказати атрибут multiple. От практичний приклад такого списку:

```
<select name=day size=7 multiple>
        <option value=1>Понеділок</option>
        <option value=1>Вівторок</option>
        <option value=1>Середа</option>
        <option value=1>Четвер</option>
        <option value=1>П'ятниця</option>
        <option value=1>Субота</option>
        <option value=1>Неділя</option>
</select>
```

Завантаження файлів на сервер

Тег INPUT дозволяє реалізувати ще одну можливість форм, а саме, створювати поле вибору файлу для його відправлення на сервер. Синтаксис наступний:

```
<input type=file name=ім'я [value=ім'я_файлу]>
```

Перед відправкою форм потрібна перевірка введених даних на рівні JavaScript.

Перевірка введених даних

Перевірка форм на заповненість необхідних полів – одна з найважливіших складових у роботі сайту. Сенс перевірки полів форми зводиться до того, що користувач відразу на сторінці отримує інформацію про те, які поля не заповнені, які заповнені, але не правильно (e-mail, телефонні номери, url та ін.) Для таких випадків дуже корисно використовувати плагін jQuery Validation Plugin [16], що вміє перевіряти форму на коректність заповнених даних (jquery.validate.js).

Приклад HTML форми

```
<form method="post" enctype="multipart/form-data" id="contacsform">

    <div class="item">
    <label>Имя</label>
    <input type="text" name="name" id="name"/>
    </div>

    <div class="item">
    <label>Email</label>
    <input type="text" name="email" id="email"/>
    </div>

    <button type="button" class="send">Надіслать</button>
</form>
```

Приклад на jQuery

```
$(document).on('click', '.send', function() {
    var validateForm = $(document).find('#contacsform');
    validateForm.validate({
        rules: {
            name: {
                required: true,
                minlength: 2
            }
            email: {
                required: true,
                email: true,
                minlength: 5
            }
        },
        messages: {
            name: {
                required: "Введіть Ваше Ім'я",
                minlength: "Повинно бути більше 2-х символів"
            }
            phone: {
                required: "Введіть Email",
                minlength: "Поле повинно бути більше 5-х символів",
                email: "Введено Email некоректно!"
            }
        },
        submitHandler: function(form) {
            form.submit();
        },
        errorPlacement: function(error, element) {
            var item = element.parents('.item');
            item.append(error);
        }
    });
    validateForm.submit();
});
```

jQuery validation перевизначення значень за замовчуванням

Змінні, які можете змінити:

```
  jQuery.extend(jQuery.validator.messages, {
    required: "This field is required.",
    remote: "Please fix this field.",
    email: "Please enter a valid email address.",
    url: "Please enter a valid URL.",
    date: "Please enter a valid date.",
    dateISO: "Please enter a valid date (ISO).",
    number: "Please enter a valid number.",
    digits: "Please enter only digits.",
    creditcard: "Please enter a valid credit card number.",
    equalTo: "Please enter the same value again.",
```

```
    accept: "Please enter a value with a valid extension.",
    maxlength: jQuery.validator.format("Please enter no more than {0}
characters."),
    minlength: jQuery.validator.format("Please enter at least {0}
characters."),
    rangelength: jQuery.validator.format("Please enter a value between {0}
and {1} characters long."),
    range: jQuery.validator.format("Please enter a value between {0} and
{1}."),
    max: jQuery.validator.format("Please enter a value less than or equal
to {0}."),
    min: jQuery.validator.format("Please enter a value greater than or
equal to {0}.")
});
```

Докладніше про jQuery validation за посиланням [17]: https://jqueryvalidation.org/documentation/.

У програмуванні **регулярні вирази** надають стислий та гнучкий засіб для ідентифікації рядків тексту, таких як конкретні символи, слова або шаблони символів. Регулярні вирази (скорочено regex або regexp з множинними формами regexps або regexen) записуються офіційною мовою, яка може бути інтерпретована процесором регулярних виразів, програмою, яка або служить генератором parser, або аналізує текст та ідентифікує частини, які відповідають даній специфікації.

Перевірка введення за регулярним виразом, наприклад, тільки латинських літер:

```
jQuery.validator.addMethod("accept", function(value, element, param) {
  return value.match(new RegExp("." + param + "$"));
});

validateForm.validate({
rules: {
    firstname: {
        required: true,
        accept: "accept: "[a-zA-Z]+",",
        minlength: 2
    }
},
messages: {
    firstname: {
        required: "Enter your name",
        accept: "Only letters allowed",
        minlength: "Name must be more than 2 characters"
    }
}
```

Корисні посилання:
https://developer.mozilla.org/ru/docs/Learn/Forms/Form_validation
http://яваскрипт.укр/%D0%BF%D1%83%D0%B1%D0%BB%D1%96%D0%BA%D0%B0%D1%86%D1%96%D1%8F/2
https://uk.education-wiki.com/4461187-javascript-form-validation

https://learn.javascript.ru/form-elements#navigatsiya-formy-i-elementy

Завдання:

1. Скласти програму, що приймає анкетні дані користувача. Намалювати HTML форму. Написати JS-сценарій, який виведе у діалоговому вікні пояснювальну інформацію щодо введення даних до форми.

2. Всі поля обов'язкові для заповнення. Здійснити попередню перевірку коректності даних, які вносяться користувачем у форму у вигляді окремого JS- сценарію, для цього слід застосувати механізм регулярних виразів `RegExp`.

3. Реалізувати аналогічну перевірку коректності даних, які вносяться користувачем у форму із використанням бібліотеки jQuery validate.

4. Перевіряти правильність даних потрібно також на стороні сервера. У випадку помилки знову відображається форма з відповідним повідомленням про помилку. Ті поля, які були заповнені вірно повинні зайняти своє значення. Показати роботу всіх елементів керування (TEXT, SELECT, RADIO, CHECKBOX).

Контрольні питання:

1) Що представляє собою форма, якими атрибутами вона володіє?
2) Яка різниця між GET і POST?
3) Назвіть елементи керування формами?
4) Які функції форми реалізуються тегом INPUT?
5) Для чого використовують залежні перемикачі, чим відрізняються від незалежних перемикачів?
6) Як працює механізм регулярних виразів RegExp?
7) Для чого використовують jQuery Validation Plugin?
8) Як отримати доступ до даних введених з форми на стороні сервера?

Лабораторна робота №6

Тема: Робота з рядками в PHP.
Мета: Навчитися застосовувати функції обробки тексту.

Знати: Основні оператори, стандартні функції PHP, синтаксис основних типів даних.
Вміти: Визначати рядки, застосовувати PHP для роботи з рядками.

Теоретичні відомості:

Рядок в PHP - це набір символів будь-якої довжини. На відміну від Сі, рядки можуть містити в собі також і нульові символи, що ніяк не вплине на програму. Іншими словами, рядки можна використовувати для зберігання бінарних даних. Довжина рядка обмежена тільки розміром оперативної пам'яті [6].

В PHP [15] символ це теж саме, що й байт, це значить, що можливо рівно 256 різних символів. Це також означає, що PHP не має вбудованої підтримки Unicode. Деяку підтримку Unicode забезпечують функції **utf8_encode()** і **utf8_decode()**.

Рядок легко може бути оброблений за допомогою стандартних функцій, можна також безпосередньо звернутися до будь-якого її символу.

Простий приклад рядкової змінної:

```
<?
                $a = "Це просто текст, записаний у рядкову змінну";
        Echo $a; //Виводить 'Це просто текст, записаний у рядкову змінну'
 ?>
```

Синтаксис типу string (рядків)

Рядок може бути визначений трьома різними способами.

- одинарними лапками
- подвійними лапками
- heredoc-синтаксисом

Визначення рядків одинарними лапками:

Найпростіший спосіб визначити рядок - це укласти його в одинарні лапки (символ ').

Щоб використовувати одинарні лапки всередині рядка, як і в багатьох інших мовах, його необхідно випередити символом зворотної косої риси (\), тобто екранувати її. Якщо зворотна коса риса повинна йти перед одинарними лапками або бути наприкінці рядка, вам необхідно продублювати її. Зверніть увагу, що якщо ви спробуєте екранувати будь-який інший символ, зворотна коса риса також буде надрукована! Так що, як правило, немає необхідності екранувати саму зворотну косу рису.

На відміну від двох інших синтаксисів, змінні й послідовності, що екранують, для спеціальних символів, що зустрічаються в рядках, укладених в одинарні лапки**,** не обробляються.

Приведемо приклад використання одинарних лапок [6]:

```
<?php
 echo 'це простий рядок';
 echo 'Також ви можете вставляти в рядки
 символ нового рядка таким чином,
 оскільки це нормально';
```

```
 // Виведе: Один раз Арнольд сказав: "I'll be back"
 echo 'Один раз Арнольд сказав: "I\'ll be back"';
 // Виведе: Ви видалили C:\*.*?
echo 'Ви видалили C:\\*.*?';
 // Виведе: Ви видалили C:\*.*?
echo 'Ви видалили C:\*.*?';
 // Виведе: Це не вставить: \n новий рядок
 echo 'Це не вставить: \n новий рядок';
 // Виведе: Змінні $expand також $either не підставляються
 echo 'Змінні $expand також $either не підставляються';
 ?>
```

Визначення рядків подвійними лапками:

Якщо рядок укладений у подвійні лапки ("), PHP розпізнає більшу кількість керуючих послідовностей для спеціальних символів:

Таблиця керуючих послідовностей [6]:

Послідовність	**Значення**
\n	новий рядок (LF або 0x0A (10) в ASCII)
\r	повернення каретки (CR або 0x0D (13) в ASCII)
\t	горизонтальна табуляція (HT або 0x09 (9) в ASCII)
\\	зворотна коса риса
\$	знак долара
\"	подвійні лапки
\[0-7]{1,3}	послідовність символів, що відповідає регулярному вираженню, символ у восьмеричній системі числення
\x[0-9A-Fa-f]{1,2}	послідовність символів, що відповідає регулярному вираженню, символ у шестнадцятковій системі числення

Визначення рядків heredoc-синтаксисом:

Інший спосіб визначення рядків - це використання heredoc-**синтаксису** ("<<<"). Після <<< необхідно вказати ідентифікатор, потім іде рядок, а потім цей же ідентифікатор, що закриває вставку.

Закриваючий ідентифікатор повинен починатися в першому стовпці рядка. Крім того, ідентифікатор повинен відповідати тим же правилам іменування, що й всі інші мітки в PHP: містити тільки буквено-цифрові символи й знак підкреслення, і повинен починатися з нецифри або знака підкреслення.

Увага! Дуже важливо відзначити, що рядок із закриваючим ідентифікатором не містить інших символів, за винятком, можливо, крапки з коми (**;**). Це означає, що ідентифікатор не повинен вводитися з відступом і, що не може бути ніяких пробілів або знаків табуляції до або після крапки з комою. Важливо також розуміти, що першим символом перед закриваючим ідентифікатором повинен бути символ нового рядка, певний у вашій операційній системі. Наприклад, на Windows це **\r**.

Якщо це правило порушене й закриваючий ідентифікатор не є "чистим", вважається, що закриваючий ідентифікатор відсутній і PHP продовжить його пошук далі. Якщо в цьому випадку вірний закриваючий ідентифікатор так і не буде знайдений, то це викличе помилку в обробці з номером рядка наприкінці скрипта.

Heredoc-Текст поводиться так само, як і рядок у подвійних лапках, при цьому їх не маючи. Це означає, що вам немає необхідності екранувати лапок в heredoc, але ви як і раніше можете використовувати перераховані вище керуючі послідовності. Змінні обробляються, але із застосуванням складних змінних усередині heredoc потрібно бути також уважним, як і при роботі з рядками.

Приклад визначення heredoc-рядка:

```
<?php
 $str = <<<EOD
 Приклад рядка, що
 охоплює кілька рядків,
 з використанням heredoc-синтаксису.
EOD;
 /* Більш складний приклад зі змінними. */
 class foo
 {
      var $foo;
      var $bar;
      function foo()
      {
          $this->foo = 'Foo';
          $this->bar = array('Bar1', 'Bar2', 'Bar3');
      }
 }
 $foo = new foo();
 $name = 'Моє Ім’я';
 echo <<<EOT
 Мене кличуть "$name". Я друкую $foo->foo.
Тепер я виводжу {$foo->bar[1]}.
Це повинно вивести заголовну букву 'A': \x41
 EOT;
 ?>
```

Обробка рядків

Якщо рядок визначається в подвійних лапках, або за допомогою heredoc, змінні усередині його обробляються.

Існує два типи синтаксису: простий і складний. Простий синтаксис більш легкий і зручний. Він дає можливість обробки змінної, значення масиву або властивості об'єкта.

Складний синтаксис був введений в PHP 4 [15] і може бути розпізнаний по фігурних дужках, що оточують вирази.

Простий синтаксис

Якщо інтерпретатор зустрічає знак долара (*$*), він захоплює так багато символів, скільки можливо, щоб сформувати правильне ім'я змінної. Якщо ви хочете точно визначити кінець ім'я, помістіть ім'я змінної у фігурні дужки.

```
<?php
      $beer = 'Heineken';
      echo "$beer's taste is great";
      // працює, "'" це невірний символ для імені змінної
      echo "He drank some $beers";
      // не працює, 's' це вірний символ для імені змінної
       echo "He drank some ${beer}s"; // працює
       echo "He drank some {$beer}s"; // працює
 ?>
```

Таким же чином можуть бути оброблені елемент масиву або властивість об'єкта. В індексах масиву закриваюча квадратна дужка ([]) позначає кінець визначення індексу. Для властивостей об'єкта

застосовуються ті ж правила, що й для простих змінних, хоча з ними неможливий трюк, як зі змінними.

```
<?php
      // Ці приклади специфічно про використання масивів усередині
      // рядків. Поза рядками завжди містіть рядкові ключі вашого
      // масиву в лапки й не використовуйте поза рядками {дужки}.
      // Давайте покажемо всі помилки
      error_reporting(E_ALL);
      $fruits = array('strawberry' => 'red', 'banana' => 'yellow');
      // Працює, але помітьте, що поза лапками рядка це працює по-іншому
      echo "A banana is $fruits[banana].";
      //Працює
      echo "A banana is {$fruits['banana']}.";
      // Працює, але PHP, як описано нижче, спочатку шукає
      // константу banana.
      echo "A banana is {$fruits[banana]}.";
// Не працює, використовуйте фігурні дужки. Це викличе помилку обробки.
      echo "A banana is $fruits['banana'].";
      // Працює
      echo "A banana is " . $fruits['banana'] . ".";
      // Працює
      echo "This square is $square->width meters broad.";
      // Не працює. Для рішення див. складний синтаксис.
      echo "This square is $square->width00 centimeters broad.";
?>
```

Для більш складних завдань використовують складний синтаксис.

Складний (фігурний) синтаксис

Даний синтаксис називається складним не тому, що важкий в розумінні, а тому, що дозволяє використовувати складні вирази.

Фактично, ви можете включити будь-яке значення, що перебуває в просторі ім'я в рядку із цим синтаксисом. Ви просто записуєте вираз в такий же спосіб, як і поза рядком, а потім помістіть його в { і }. Оскільки ви не можете екранувати '{', цей синтаксис буде розпізнаватися тільки коли $ треба безпосередньо за {. (Використовуйте "{\$" або "\{$" щоб відобразити "{$"). Кілька прикладів, що пояснюють:

```
<?php
      // Давайте покажемо всі помилки
      error_reporting(E_ALL);
      $great = 'fantastic';
      // Не працює, виведе: This is { fantastic}
      echo "This is { $great}";
      // Працює, виведе: This is fantastic
      echo "This is {$great}";
      echo "This is ${great}";
      // Працює
      echo "Цей квадрат шириною {$square->width}00 сантиметрів.";
      // Працює
      echo "Це працює: {$arr[4][3]}";
      // Це невірно по тій же причині, що й $foo[bar] невірно поза
      // рядком. Інакше кажучи, це як і раніше буде працювати,
      // але оскільки PHP спочатку шукає константу foo, це викличе
      // помилку рівня E_NOTICE (невизначена константа).
      echo "Це неправильно: {$arr[foo][3]}";
      // Працює. При використанні багатомірних масивів, усередині
      // рядків завжди використовуйте фігурні дужки
      echo "Це працює: {$arr['foo'][3]}";
      // Працює.
      echo "Це працює: " . $arr['foo'][3];
      echo "Ви навіть можете записати {$obj->values[3]->name}";
      echo "Це значення змінної по ім'ю $name: {${$name}}";
?>
```

Доступ до символу в рядку і його зміна:

Символи в рядках можна використовувати й модифікувати, визначивши їхній зсув відносно початку рядка, починаючи з нуля, у фігурних дужках після рядка. Наприклад:

```
<?php
      // Одержання першого символу рядка
       $str = 'Це тест.';
       $first = $str{0};
       // Одержання третього символу рядка
       $third = $str{2};
       // Одержання останнього символу рядка
       $str = 'Це все ще тест.';
       $last = $str{strlen($str)-1};
       // Зміна останнього символу рядка
       $str = 'Подивися на море';
       $str{strlen($str)-1} = 'я';
?>
```

Рядкові функції й оператори

Рядкові оператори

Конкатенація рядків:

У різних мовах програмування використовуються різні оператори конкатенації (об'єднання) рядків. Наприклад, в Pascal використовується оператор "+". Використання в PHP оператора "+" для конкатенації рядків некоректно: якщо рядки містять числа, то замість об'єднання рядків буде виконано операцію додавання двох чисел.

В PHP є два оператори, що виконують конкатенацію [6].

Перший - оператор конкатенації ('.'), що повертає об'єднання лівого й правого аргументу.

Другий - оператор присвоєння з конкатенацією, що приєднує правий аргумент до лівого.

Приведемо конкретний приклад:

```
<?php
       $a = "Hello ";
       $b = $a . "World!"; // $b містить рядок "Hello World!"      -      Це
конкатенація
      $a = "Hello ";
       $a .= "World!";
      // $a містить рядок "Hello World!" - Це присвоєння з конкатенацією
?>
```

Оператори порівняння рядків

Для порівняння рядків не рекомендується використовувати оператори порівняння == і !=, оскільки вони вимагають перетворення типів. Приклад:

```
<?php
       $x=0;
       $y=1;
       if ($x == "") echo "<p>x - порожній рядок</p>";
       if ($y == "") echo "<p>y - порожній рядок</p>";
       // Виводить:
       // x - порожній рядок
 ?>
```

Даний скрипт повідомляє нас, що $x - порожній рядок. Це пов'язане з тим, що порожній рядок ("") трактується насамперед як 0, а тільки потім - як "порожньо". В PHP операнди рівняються, як рядки, тільки в тому

випадку, якщо обоє вони - рядки. У іншому випадку вони визначаються як числа. При цьому будь-який рядок, що PHP не вдається перевести в число (у тому числі й порожній рядок), буде сприйматися як 0.

Приклади порівняння рядків:

```
<?php
      $x="Рядок";
      $y="Рядок";
      $z="Рядок";
      if ($x == $z) echo "<p>Рядок X дорівнює рядку Z</p>";
      if ($x == $y) echo "<p>Рядок X дорівнює рядку Y</p>";
      if ($x != $z) echo "<p>Рядок X НЕ дорівнює рядку Z</p>";
      // Виводить:
      // Рядок X дорівнює рядку Y
      // Рядок X НЕ дорівнює рядку Z
?>
```

Щоб уникнути путанини й перетворення типів, рекомендується користуватися оператором еквівалентності при порівнянні рядків. Оператор еквівалентності дозволяє завжди коректно порівнювати рядки, оскільки порівнює величини й за значенням, і по типу:

```
<?php
     $x="Рядок";
     $y="Рядок";
     $z="Рядок";
     if ($x === $z) echo "<p>Рядок X дорівнює рядку Z</p>";
     if ($x === $y) echo "<p>Рядок X дорівнює рядку Y</p>";
     if ($x !== $z) echo "<p>Рядок X НЕ дорівнює рядку Z</p>";
     // Виводить:
     // Рядок X дорівнює рядку Y
     // Рядок X НЕ дорівнює рядку Z
?>
```

Функції для роботи з рядками

Для роботи з рядками в PHP існує безліч корисних функцій.

Коротко розберемо частину функцій для роботи з рядками.

Базові рядкові функції

strlen(string $st)

Одна з найбільш корисних функцій. Повертає просто довжину рядка, тобто, скільки символів утримується в $st. Рядок може містити будь-які символи, у тому числі й з нульовим кодом (що заборонено в Сі). Приклад:

```
$x = "Hello!";
echo strlen($x); // Виводить 6
```

strpos(string $where, string $what, int $fromwhere=0)

Намагається знайти в рядку $where підстроку (тобто послідовність символів) $what і у випадку успіху повертає позицію (індекс) цієї підстроки в рядку. Необов'язковий параметр $fromwhere можна задавати, якщо пошук потрібно вести не з початку рядка $from, а з якоїсь іншої позиції. У цьому випадку треба цю позицію передати в $fromwhere. Якщо підстроку знайти не вдалося, функція повертає false. Однак будьте уважні, перевіряючи результат виклику strpos() на false - використовуйте для цього тільки оператор ===. Приклад:

```
echo strpos("Hello","el"); // Виводить 1
```

І ще приклад:

```
if (strpos("Norway","rwa") !== false) echo "Рядок rwa є в Norway";
// При порівнянні використовуйте оператори тотожних порівнянь (===)
(!==) щоб уникнути проблем з визначенням типів
```

substr(string $str, int $start [,int $length])

Дана функція теж затребується дуже часто. Її призначення - повертати ділянка рядка $str, починаючи з позиції $start і довжиною $length. Якщо $length не задана, то мається на увазі підстрока від $start до кінця рядка $str. Якщо $start більше, ніж довжина рядка, або ж значення $length дорівнює нулю, то вертається порожня підстрока. Однак ця функція може робити й ще досить корисні речі. Приміром, якщо ми передамо в $start негативне число, то буде вважатися, що це число є індексом підстроки, але тільки відлічуваним від кінця $str (наприклад, -1 означає "починаючи з останнього символу рядка"). Параметр $length, якщо він заданий, теж може бути негативним. У цьому випадку останнім символом повернутої підстроки буде символ з $str з індексом $length, обумовленим від кінця рядка. Приклади:

```
$str = "Programmer";
echo substr($str,0,2); // Виводить Pr
echo substr($str,-3,3); // Виводить mer
```

strcmp(string $str1, string $str2)

Порівнює два рядки посимвольно (точніше, побайтово) і повертає: 0, якщо рядка повністю збігаються; -1, якщо рядок $str1 лексикографічно менше $str2; і 1, якщо, навпаки, $str1 "більше" $str2. Тому що порівняння йде побайтово, те регістр символів впливає на результати порівнянь.

strcasecmp(string $str1, string $str2)

Те ж саме, що й strcmp(), тільки при роботі не враховується регістр букв. Наприклад, з погляду цієї функції "ab" і "AB" рівні.

Функції для роботи із блоками тексту

Перераховані нижче функції найчастіше виявляються корисні, якщо потрібно проводити однотипні операції із багаторядковими блоками тексту, заданими в строкові змінні.

str_replace(string $from, string $to, string $str)

Заміняє в рядку $str всі входження підстроки $from (з урахуванням регістра) на $to і повертає результат. Вихідний рядок, переданий третім параметром, при цьому не міняється. Ця функція працює значно швидше, ніж **ereg_replace()**, що використовується при роботі з регулярними виразами PHP, і її часто використовують, якщо немає необхідності в якихось екзотичних правилах пошуку підстроки. Наприклад, от так ми можемо замістити всі символи переводу рядка на їх HTML еквівалент - тег
:

```
$st=str_replace("\n","<br>\n",$str)
```

Як бачимо, те, що в рядку
\n теж є символ переводу рядка, ніяк не впливає на роботу функції, тобто функція робить лише однократний прохід по рядку. Для рішення описаного завдання також застосовна функція nl2br(), що працює ледве швидше.

string nl2br(string $string)

Заміняє в рядку всі символи нового рядка \n на
\n і повертає результат. Вихідний рядок не змінюється. Зверніть увагу на те, що символи \r, які присутні наприкінці рядка текстових файлів Windows, цією функцією ніяк не враховуються, а тому залишаються на старому місці.

WordWrap(string $str, int $width=75, string $break="\n")

Ця функція, що з'явилася в PHP4, виявляється неймовірно корисної, наприклад, при форматуванні тексту листа перед автоматичним відправленням його адресатові за допомогою mail(). Вона розбиває блок тексту $str на кілька рядків, що завершуються символами $break, так, щоб на одному рядку було не більше $width букв. Розбивка відбувається по границі слова, так що текст залишається що читається. Вертається рядок, що вийшло, із символами переводу рядка, заданими в $break. Приклад використання:

```
<?php
    $str = "Це текст електронного листа, яке потрібно буде відправити адреса
     тові...";
     // Розбиваємо текст по 20 символів
     $str = WordWrap ($str, 20, "<br>");
     echo $str;
     // Виводить:
     /* Це текст
     електронного листа,
     яке потрібно буде
     відправити
     адресатові... */
?>
```

strip_tags (string $str [, string $allowable_tags])

Ще одна корисна функція для роботи з рядками. Ця функція видаляє з рядка всі теги й повертає результат. У параметрі $allowable_tags можна передати теги, які не слід видаляти з рядка. Вони повинні перераховуватися впритул друг до друга. Приклади:

```
$stripped = strip_tags ($str);
// Видаляють всі html - теги з рядка (тексту)
 $stripped = strip_tags($str, "<head><title>");
  // Видалить всі html - теги, крім html - тегів <head> і <title>
```

Функції для роботи з окремими символами

Як і в інших мовах програмування, в PHP можна працювати із символами рядків окремо.

Звернутися до будь-якого символу рядка можна за його індексом:

```
$str = "PHP";
 echo $str[0]; // Виводить 'P'
```

chr(int $code)

Дана функція повертає рядок, що складається із символу з кодом $code. Приклад:

```
echo chr(75); //Виводить K
```

ord($char)

Дана функція повертає код символу $char. Ось приклад:

```
echo ord('A'); // Виводить 65 - код букви 'A'
```

Функції видалення пробілів

Іноді важко навіть представити, якими можуть бути дивними користувачі, якщо дати їм у руки клавіатуру й попросити надрукувати на ній яке-небудь слово. Тому що клавіша пробілу - сама більша, то користувачі мають звичай натискати її в самі неймовірні моменти. Цьому сприяє також і той факт, що символ з кодом 32, що позначає пробіл, як ви знаєте, на екрані не видний. Якщо програма не здатна обробити описану ситуацію, то вона, у найкращому разі після тяжкого мовчання відобразить у браузері що-небудь типу "невірні вхідні дані", а в гіршому - зробить при цьому що-небудь необоротне.

Тим часом, убезпечити себе від паразитних пробілів надзвичайно просто, і розроблювачі PHP надають нам для цього ряд спеціалізованих функцій. Не хвилюйтеся про те, що їхнє застосування сповільнює програму. Ці функції працюють із блискавичною швидкістю, а головне, однаково швидко, незалежно від обсягу переданих їм рядків.

trim(string $str)

Повертає копію $str, тільки з вилученими провідними й кінцевими пробельними символами. Під пробельними символами я тут і далі маю на увазі: пробіл " ", символ переводу рядка \n, символ повернення каретки \r і символ табуляції \t. Наприклад, виклик trim(" test\n ") поверне рядок "test". Ця функція використовується дуже широко. Намагайтеся застосовувати її скрізь, де є хоч найменша підозра на наявність помилкових пробілів. Оскільки працює вона дуже швидко.

ltrim(string $st)

Те ж, що й trim(), тільки видаляє винятково провідні пробіли, а кінцеві не торкає. Використовується набагато рідше.

chop(string $st)

Видаляє тільки кінцеві пробіли, ведучі не торкає.

Функції перетворення символів

Web-Програмування - одна з тих областей, у яких постійно доводиться маніпулювати рядками: розривати їх, додавати й видаляти пробіли, перекодувати в різні кодування, нарешті, URL- кодувати й декодувати. В PHP реалізувати всі ці дії вручну, використовуючи тільки вже описані примітиви, просто неможливо з міркувань швидкодії. Тому-то й існують подібні вбудовані функції.

strtr(string $str, string $from, string $to)

Ця функція застосовується не настільки широко, але все-таки іноді вона буває досить корисною. Вона заміняє в рядку $str всі символи, що зустрічаються в $from, на їх "парні" (тобто розташовані в тих же позиціях, що й в $from) з $to.

Наступні кілька функцій призначені для швидкого URL-Кодування й декодування.

URL-Кодування необхідно для передачі даних через інтернет. Наприклад, таке кодування доцільно, якщо ви передаєте російськомовну інформацію як параметр скрипта. Також подібне кодування можна виконати й для файлу, щоб не виникало колізій через відсутність підтримки 8-бітних кодувань деякими серверами. От ці функції:

UrlEncode(string $str)

Функція URL-Кодує рядок $str і повертає результат. Цю функцію зручно застосовувати, якщо ви, наприклад, хочете динамічно сформувати посилання <a href=...> на якийсь сценарій, але не впевнені, що його параметри містять тільки алфавітно-цифрові символи. У цьому випадку скористайтеся функцією так:

```
echo "<a href=/script.php?param=".UrlEncode($UserData);
```

Тепер, навіть якщо змінна $UserData включає символи =, & або навіть пробіли, однаково сценарію будуть передані коректні дані.

UrlDecode(string $st)

Робить URL-Декодування рядка. У принципі, використовується значно рідше, ніж UrlEncode(), тому що PHP і так уміє перекодувати вхідні дані автоматично.

RawUrlEncode(string $st)

Майже повністю аналогічна UrlEncode(), але тільки пробіли не перетворяться в +, як це робиться при передачі даних з форми, а сприймаються як звичайні неалфавитно-цифрові символи. Втім, цей метод не породжує ніяких додаткових несумісностей у коді.

RawUrlDecode(string $st)

Аналогічна UrlDecode(), але не сприймає + як пробіл.

HtmlSpecialChars(string $str)

Це функція, що звичайно використовується в комбінації з echo. Основне її призначення - гарантувати, що у виведеному рядку жоден ділянка не буде сприйнятий як тег.

Заміняє в рядку деякі символи (такі як амперсант, лапки й знаки "більше" і "менше") на їх HTML-Еквіваленти, так, щоб вони виглядали на сторінці "самими собою". Саме типове застосування цієї функції - формування параметра value у різних елементах форми, щоб не було ніяких проблем з лапками, або ж вивід повідомлення в гостьовій книзі, якщо вставляти теги користувачеві заборонено.

StripSlashes(string $str)

Заміняє в рядку $str деякі випереджені слешем символи на їхні однокодові еквіваленти. Це ставиться до наступних символів: ", ', \ і ніяким іншим.

AddSlashes(string $str)

Вставляє слеши тільки перед наступними символами: ', " і \. Функцію дуже зручно використовувати при виклику **eval()** ця функція виконує рядок, переданий їй у параметрах, так, начебто має справу з невеликий PHP-Програмою.

Функції зміни регістра

Досить часто користувачу доводиться переводити рядки, скажемо, у верхній регістр, тобто робити всі прописні букви в рядку заголовними. Для цієї мети можна було б скористатися функцією strtr(), що розглянута вище, але вона все-таки буде працювати не так швидко, як іноді хотілося б. В PHP є функції, які призначені спеціально для таких потреб. От вони:

strtolower(string $str)

Перетворить рядок у нижній регістр. Повертає результат переводу.

Треба помітити, що при неправильному настроюванні локали (це набір правил по переводу символів з одного регістра в інший, переводу дати й часу, грошових одиниць і т.д.) функція буде видавати неправильні результати при роботі з буквами кирилиці.

Можливо, у нескладних програмах, а також якщо немає впевненості в підтримці відповідної локали операційною системою, простіше буде скористатися "ручним" перетворенням символів, задіючи функцію strtr():

```
$st=strtr($st, "АБВГГДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ",
"абвггдеєжзиіїйклмнопрстуфхцчшщьюя");
```

Головні переваги даного способу - те, що у випадку проблем з кодуванням для відновлення працездатності сценарію вам доведеться всього лише перетворити його в те ж кодування, у якому у вас зберігаються документи на сервері.

strtoupper(string $str)

Переводить рядок у верхній регістр. Повертає результат перетворення. Ця функції також прекрасно працює з рядками, складеними з латиниці, але з кирилицею може виникнути все та ж проблема.

Установка локалі (локальних настроювань)

Локалью будемо називати сукупність локальних настроювань системи, таких як формат дати й часу, мова, кодування.

Настроювання локалі сильно залежать від операційної системи.

Для установки локалі використовується функція SetLocale():

SetLocale(string $category, string $locale)

Функція встановлює поточну локаль, з якої будуть працювати функції перетворення регістра символів, виводу дати-часу й.т.д. Загалом кажучи, для кожної категорії функцій локаль визначається окремо й виглядає по-різному. Те, яку саме категорію функцій торкне виклик SetLocale(), задається в параметрі $category. Він може приймати наступні строкові значення:

LC_CTYPE — активізує зазначену локаль для функцій переводу у верх ний/нижній регістри;

LC_NUMERIC — активізує локаль для функцій форматування дробових чисел — а саме, задає роздільник цілої й дробової частини в числах;

LC_TIME — задає формат виводу дати й часу за замовчуванням;

LC_ALL — установлює всі перераховані вище режими.

Тепер про параметр $locale. Як відомо, кожна локаль, установлена в системі, має своє унікальне ім'я, по якому до неї можна звернутися. Саме воно й фіксується в цьому параметрі. Однак, є два важливих виключення із цього правила. По-перше, якщо величина $locale дорівнює порожньому рядку "", те встановлюється та локаль, що зазначена в глобальної змінної оточення з ім'ям, що збігається з ім'ям категорії $category (або LANG - вона практично завжди є присутнім в Unix). У других, якщо в цьому параметрі передається 0, те нова локаль не встановлюється, а просто вертається ім'я поточної локали для зазначеного режиму.

На жаль, імена локалей задаються при настроюванні операційної системи, і для них, очевидно, не існує стандартів. З'ясуєте у свого хостинг-провайдеру, як називаються локали для різних кодувань українських символів. Але, якщо наступний фрагмент працює у вашого хостинг-провайдеру, це не означає, що він заробить, наприклад, під Windows:

```
setlocale('LC_CTYPE','ru_SU.KOI 8-R');
```

Тут виклик установлює таблицю заміни регістра букв відповідно до кодування KOI 8-R.

По правді говорячи, локаль - річ досить непередбачена й досить погано стерпна між операційними системами. Так що, якщо ваш сценарій

не дуже великий, задумайтеся: можливо, краще буде шукати обхідний шлях, наприклад, використовуючи strtr(), а не розраховувати на локаль.

Функції перетворення кодувань [6]

Часто зустрічається ситуація, коли нам потрібно перетворити рядок з одного кодування кирилиці в іншу. Наприклад, ми в програмі перемінили локаль: було кодування windows, а стала - KOI 8-R. Але рядки-те залишилися як і раніше в кодуванні WIN-1251, а виходить, для правильної роботи з ними нам потрібно їх перекодувати в KOI 8-R. Для цього й служить функція перетворення кодувань.

convert_cyr_string(string $str, char $from, char $to);

Функція переводить рядок $str з кодування $from у кодування $to. Звичайно, це має сенс тільки для рядків, що містять "росіяни" букви, тому що латиниця у всіх кодуваннях виглядає однаково. Зрозуміло, кодування $from повинна збігатися із щирим кодуванням рядка, інакше результат вийде невірним. Значення $from і $to - один символ, що визначає кодування:

k — koi 8-r
w — windows-1251
i — iso 8859-5
a — x-cp866
d — x-cp866
m — x-mac-cyrillic

Функція працює досить швидко, так що її цілком можна застосовувати, скажемо, для перекодування листів у потрібну форму перед їхнім відправленням по електронній пошті.

Функції форматних перетворень рядків

Як ми знаємо, змінні в рядках PHP інтерполюються, тому практично завжди завдання "змішування" тексту зі значеннями змінних не є проблемою. Наприклад, ми можемо спокійно написати щось начебто:

```
echo "Привіт, $name! Вам $age років.";
```

У Сі для аналогічних цілей використовується наступний код:

```
printf("Привіт, %s! Вам %s років",name,age);
```

Мова PHP також підтримує ряд функцій, що використовують такий же синтаксис, як і їх Сі -еквіваленти. Бувають випадки, коли їхнє застосування дає найбільш гарне й лаконічне рішення, хоча це й трапляється досить нечасто.

sprintf(string $format [, mixed args, ...])

Ця функція — аналог функції sprintf() у Сі. Вона повертає рядок, складений на основі рядка форматування, що містить деякі спеціальні символи, які будуть згодом замінені на значення відповідних змінних зі списку аргументів.

Рядок форматування $format може містити в собі команди форматування, випереджені символом %. Всі інші символи копіюються у вихідний рядок як є. Кожний специфікатор формату (тобто, символ % і наступні за ним команди) відповідає одному, і тільки одному параметру, зазначеному після параметра $format. Якщо ж потрібно помістити в текст % як звичайний символ, необхідно його подвоїти:

```
echo sprintf("The percentage was %d%%",$percentage);
```

Кожний специфікатор формату включає максимум п'ять елементів (у порядку їхнього проходження після символу %):

>>> Необов'язковий специфікатор розміру поля, що вказує, скільки символів буде відведено під виведену величину. Як символи-заповнювачів (якщо значення має менший розмір, чим розмір поля для його виводу) може використовуватися пробіл або 0, за замовчуванням підставляється пробіл. Можна задати будь-який інший символ-наповнювач, якщо вказати його в рядку форматування, випередивши апострофом '.

>>> Опціональний специфікатор вирівнювання, що визначає, буде результат вирівняний по правому або по лівому краї поля. За замовчуванням виробляється вирівнювання по правому краї, однак можна вказати й ліве вирівнювання, задавши символ - (мінус).

>>> Необов'язкове число, що визначає розмір поля для виводу величини. Якщо результат не буде в поле міститися, то він "вилізе" за краї цього поля, але не буде усічений.

>>> Необов'язкове число, випереджене крапкою ".", що пропонує, скільки знаків після коми буде в результуючому рядку. Цей специфікатор ураховується тільки в тому випадку, якщо відбувається вивід числа із плаваючою крапкою, у противному випадку він ігнорується.

>>> Нарешті, обов'язковий (помітьте - єдиний обов'язковий!) специфікатор типу величини, що буде поміщена у вихідний рядок:

b — черговий аргумент зі списку виводиться як двійкове ціле число;

c — виводиться символ із зазначеним в аргументі кодом;

d — ціле число;

f — число із плаваючою крапкою;

o — восьмеричне ціле число;

s — рядок символів;

x — шестнадцатеричное ціле число з маленькими буквами a-z;

X — шестнадцатеричное число з більшими буквами A-Z.

От як можна вказати точність подання чисел із плаваючою крапкою:

```
$money1 = 68.75;
$money2 = 54.35;
$money = $money1 + $money2;
// echo $money виведе "123.1"...
$formatted = sprintf ("%01.2f", $money);
// echo $formatted виведе "123.10"!
```

От приклад виводу цілого числа, випередженого потрібною кількістю нулів:

```
$isodate=sprintf("%04 d-d-%02 d-d-%02d",$year,$month,$day);
```

printf(string $format [, mixed args, ...])

Робить те ж саме, що й sprintf(), тільки результуючий рядок не вертається, а направляється в браузер користувача.

number_format(float $number, int $decimals, string $dec_point=".", string $thousands_sep=",");

Ця функція форматує число із плаваючою крапкою з поділом його на тріади із зазначеною точністю. Вона може бути викликана із двома або чотирма аргументами, але не із трьома! Параметр $decimals задає, скільки цифр після коми повинне бути в числа у вихідному рядку. Параметр

$dec_point являє собою роздільник цілої й дробової частин, а параметр $thousands_sep - роздільник тріад у числі (якщо вказати на його місці порожній рядок, то тріади не відділяються друг від друга).

В PHP існує ще кілька функцій для виконання форматних перетворень, серед них - **sscanf()** і **fscanf()**, які часто застосовуються в Сі. Однак в PHP їхнє використання досить обмежене: найчастіше для розбору рядків виявляється набагато вигідніше залучити регулярні вираження або функцію **explode()**.

Хеш-Функції

md5(string $str)

Повертає хеш-код рядка $str, заснований на алгоритмі корпорації RSA Data Security за назвою "MD5 Message-Digest Algorithm". Хеш-Код - це просто рядок, практично унікальний для кожного з рядків $str. Тобто ймовірність того, що два різні рядки, передані в $str, дадуть нам однаковий хеш-код, прагне до нуля.

Якщо довжина рядка $str може досягати декількох тисяч символів, то її MD 5-код займає максимум 32 символу.

Для чого потрібний хеш-код і, зокрема, алгоритм MD5? Наприклад, для перевірки паролів на істинність.

Нехай, приміром, у нас є система з багатьма користувачами, кожний з яких має свій пароль. Можна, звичайно, зберігати всі ці паролі у звичайному виді, або зашифрувати їх яким-небудь способом, але тоді велика ймовірність того, що одного чудового дня цей файл із паролями у вас украдуть.

Зробимо так: у файлі паролів будемо зберігати не самі паролі, а їх (MD5) хеш-коди. При спробі якого або користувача ввійти в систему ми обчислимо хеш-код тільки що уведеного їм пароля й зрівняємо його з тим, що записаний у нас у базі даних. Якщо коди збіжаться, виходить, усе в порядку, а якщо немає — що ж, ...

Звичайно, при обчисленні хеш-коду якась частина інформації про рядок $str безповоротно губиться. І саме це дозволяє нам не побоюватися, що зловмисник, що одержав файл паролів, зможе його коли-небудь розшифрувати. Адже в ньому немає самих паролів, немає навіть їхніх зв'язкових частин.

Приклад використання алгоритму хешування MD5:

```
<?php
$pass_a = "MySecret";
     $pass_b = "MySecret";
     // Виводимо хеш-код рядка MySecret ($pass_a) - вихідний пароль
     echo "<b> Хеш-
     Код вихідного пароля '$pass_a':</b><b style=\"color:green\">".md5($pass
     _a)."</b><br>";
     // Виводимо хеш-код рядка MySecret ($pass_b) - верифікований пароль
     echo "<b> Хеш-
     код верифікованого паролю '$pass_b':</b><b style=\"color:green\">".md5(
     $pass_b)."</b><br>";
     // Порівнюємо хеш-коди MD5 вихідного й верифікованого паролю
     echo "<h3>Перевіряємо істинність уведеного пароля:</h3>";

     if (md5($pass_a)===md5($pass_b)) echo "<h3 style=\"color:green\">Пароль
      вірний! ( Хеш-Коди збігаються)</h3>";
     else echo "<h3 style=\"color:red\">Пароль невірний! ( Хеш-
```

```
    Коди не збігаються)</h3>"
     // У даній ситуації виводить: Пароль вірний! ( Хеш-Коди збігаються)
     // Спробуйте змінити значення рядка $pass_b :)
?>
```

crc32(string $str)

Функція crc32() обчислює 32-бітну контрольну суму рядка $str. Тобто, результат її роботи - 32 бітне (4-байтове) ціле число. Ця функція працює набагато швидше md5(), але в той же час видає набагато менш надійні " хеш-коди" для рядка.

Функції скидання буфера виводу

flush()

Ця функція має дуже й дуже віддалене відношення до роботи з рядками, але вона ще далі відстоїть від інших функцій.

Почнемо здалеку: звичайно при використанні echo дані не прямо відразу відправляються клієнтові, а накопичуються в спеціальному буфері, щоб потім транспортуватися великою "пачкою". Так виходить швидше.

Однак, іноді буває потрібно достроково відправити всі дані з буфера користувачеві, наприклад, якщо ви щось виводите в реальному часі (так найчастіше працюють чати). Отут вам і допоможе функція flush(), що відправляє вміст буфера echo у браузер користувача.

Завдання 1:

1. Скласти HTML форму, яка буде передавати текст PHP сценарію, обробляти його й виводити на екран.
2. Введений текст розбити за словами.
3. Букви в словах розгорнути місцями (перша стає останньою, остання - першою, і т.д.)
4. Склеїти оброблені слова, і вивести на екран те, що отримали

P.S. Текст повинен залишитися колишнім, включаючи абзаци, заголовки. Тільки слова читаються навпаки[6].

Завдання 2:

1. Замініть скрізь у тексті дату у форматі 'дд-мм-рррр' на дату у форматі 'рррр.мм.дд'.
2. Користувач вводить у поле пароль. Якщо кількість символів пароля більше 5-ти і менше 10-ти, то виведіть користувачеві повідомлення про те, що пароль підходить, інакше повідомлення про те, що потрібно придумати інший пароль.
3. Дано рядок $str. Замініть смайли, які зустрічаються в цьому рядку на відповідні рисунки. Різних смайлів повинно бути не менше 5.
4. Дана змінна $str, в якій зберігається рядок українського тексту. Напишіть скрипт, який запише трансліт цього тексту в змінну $translit.
5. Дана змінна $str, в якій зберігається будь-який текст. Реалізуйте обрізання довгого тексту за наступним принципом: якщо кількість символів цього тексту більше заданого у змінній $n, то в змінну $result

запишіть перші $n символів рядка $str і додайте в кінець три крапки "...". В іншому випадку в змінну $result запишіть вміст змінної $str.

6. Підрахуйте кількість входжень заданого користувачем слова у тексті.

7. Замініть у тексті всі числа на відповідні слова.

8. Зашифруйте та дешифруйте текст введений користувачем методом Цезаря. Ключ шифрування вказує користувач [6].

Контрольні питання:

1) Що представляє собою рядок в PHP?
2) Який синтаксис рядків (3 способи визначення)?
3) Які є функції для роботи з рядками в PHP?
4) Що таке регулярний вираз?
5) Яким чином, можна видалити прогалини з тексту?
6) Яким чином можна виконати в PHP конкатенацію рядків?
7) Які оператори порівняння рядків є в PHP?

Лабораторна робота №7

Тема: Системи керування контентом сайтів.

Мета: Розглянути та проаналізувати існуючі CMS, виділити їх переваги та недоліки.

Теоретичні відомості:

Система керування контентом (CMS, Content Management System) – програмне забезпечення для організації веб-сайтів чи інших інформаційних ресурсів в Інтернеті чи окремих комп'ютерних мережах [6].

Існують сотні, а може, навіть й тисячі доступних CMS-систем. Система керування контентом бувають як комерційні, так і безкоштовні.

Види CMS-систем:

- *Для управління веб-сайтами* (напр., енциклопедіями, онлайн-виданнями, блогами, форумами, корпоративними чи персональними веб-сторінками та ін.)
- *Транзакційні СКВ* для забезпечення транзакцій у електронній комерції.
- *Інтегровані СКВ* для роботи з документацією на підприємствах.
- *Електронні бібліотеки* (Digital Asset Management) для забезпечення циклу життя файлів електронних медіа (відео, графічн., презентації тощо).
- Системи для забезпечення циклу життя документації (інструкції, довідники, описи).
- *Освітні СКВ* — системи для організації Інтернет курсів та відповідного циклу життя документації.

Приклади CMS-систем:

- *WordPress* – це проста у встановленні та використанні CMS, з відкритим кодом, безкоштовна, яка широко використовується для створення веб-сайтів. Сфера застосування – від блогів до складних веб-сайтів. Є українська локалізація.
- *Joomla* – популярна CMS, з відкритим кодом, безкоштовна, є українська локалізація, дозволяє створювати сайти різної складності;
- *Drupal* – популярна CMS, з відкритим кодом, безкоштовна, є українська локалізація, дозволяє створювати сайти різної складності;
- *Moodle* – освітня CMS, безкоштовна, відкрита, має українську локалізацію, використовується більш ніж 20 ВНЗ України, також використовується у 197 країнах світу;
- *MediaWiki* – програмний рушій для веб-сайтів, що працюють за технологією «вікі», під управлінням якої працює Вікіпедія, безкоштовна відкрита, є українська локалізація.

Завдання:

1. Розглятути відомі CMS
2. Виділити переваги CMS
3. Виділити недоліки CMS. Скласти таблицю.

Лабораторна робота №8

Тема: Застосування CMS для розробки сайту.

Мета: Навчитися застосовувати системи керування контентом для створення різних типів сайтів.

Знати: Основні засоби HTML, CSS, стандартні функції PHP, синтаксис основних типів даних.

Вміти: Компонувати текстову інформацію при створенні гіпертекстових документів з використанням простих текстових редакторів. Формувати гіпертекстові посилання й списки й створювати на цій основі зв'язані гіпертекстові сторінки.

Теоретичні відомості:

Встановлення WordPress

1. Скачайте і розпакуйте дистрибутив WordPress: https://uk.wordpress.org/download/

2. Створіть базу даних для WordPress на вашому веб-сервері, щоб ваш користувач MySQL мав усі права для доступу та внесення змін до неї.

3. Перейменуйте файл wp-config-sample.php файл в wp-config.php.

4. Відкрийте wp-config.php в вашому улюбленому текстовому редакторі і впишіть в нього настройки підключення до бази даних:

```
// ** MySQL settings ** //
define('DB_NAME', 'putyourdbnamehere');    // Ім’я бази даних
define('DB_USER', 'usernamehere');     // Ім’я користувача MySQL
define('DB_PASSWORD', 'yourpasswordhere'); // Пароль користувача MySQL
define('DB_HOST', 'localhost');     // скоріш за все змінювати буде непотрібно
define('DB_CHARSET', 'utf8');
define('DB_COLLATE', '');
```

5. Додайте файли WordPress на Ваш веб-сервер.

6. Перейдіть за посиланням <назва сайту>/wp-admin/install.php використовуючи ваш улюблений веб-браузер.

7. Слідуйте інструкціям на сторінці.

Створення теми для WordPress

Теми WordPress складаються з файлів і стилів, які разом визначають зовнішній вигляд сайту. Вони можуть сильно відрізнятися між собою, дозволяючи швидко змінити дизайн веб-сайту. Тема необхідна, щоб створити унікальний дизайн сайту.

Анатомія теми

Теми WordPress знаходяться в підкаталогах wp-content/themes/. Директорія теми містить таблиці стилів, файли шаблонів, файл додаткового функціоналу (functions.php) і картинки. Наприклад, тема під назвою "test" ймовірно буде розташована в директорії wp-content/themes/test/.

За замовчуванням WordPress містить дві теми: "Classic" і "Default". Ці дві теми відрізняються між собою і використовують різні функції і теги

для створення зовнішнього вигляду і сторінок сайту. Уважно вивчіть їх складові файли, щоб краще зрозуміти, як створити свою власну тему.

Тема WordPress складається з трьох основних типів файлів, на додаток до зображень:

1. Перший – це таблиця стилів під ім'ям style.css, яка контролює зовнішній вигляд сторінок сайту.

2. Другий – файл додаткового функціоналу (functions.php).

3. Інші файли – це файли шаблонів, які визначають, яким чином виводиться інформація з бази даних на веб-сторінку.

Таблиця стилів теми

Крім власне таблиці стилів Вашої теми, файл style.css обов'язково повинен містити інформацію про вашу темі у вигляді коментарів. Кожна тема повинна мати свою власну унікальну інформацію в коментарях заголовка, в іншому випадку виникнуть проблеми в діалозі вибору теми. Створюючи нову тему на основі вже існуючої, в першу чергу змініть коментарі заголовка.

Нижче наведено шаблон заголовка таблиці стилів (style sheet header) Вашої теми Rose. Він повинен знаходиться на початку файлу стилів style.css, в перших рядках:

```
/ *
Theme Name: Rose
Theme URI: домашня-сторінка-теми
Description: короткий-опис-теми
Author: ваше-ім'я
Author URI: ваш-URI
Template: напишіть-тут-найменування-батьківської-теми -
необов'язкове-поле
Version: номер-версії -необов'язкове-поле
.
Розгорнутий опис теми / Ліцензія-якщо-потрібно.
.
* /
```

Найпростіша тема включає в себе тільки файл style.css плюс зображення, якщо вони необхідні. Для створення такої теми необхідно [6] в рядку заголовка Template: вказати ім'я батьківської теми. Наприклад, якщо ваша тема "Rose" успадковується від теми "test", напишіть в заголовку style.css наступний рядок:

Template: test

Після додавання такого рядка в заголовок style.css всі шаблони теми "test" будуть успадковуватися вашою темою "Rose", в якій нічого немає, крім файлу з таблицею стилів style.css, і можливо, малюнків. Ці файли необхідно помістити в каталог wp-content/themes/Rose.

Починаючи з WordPress 2.7 тема-нащадок може містити файли-шаблони, імена яких збігаються з іменами шаблонів в батьківській темі, в адмін-панелі можна вибрати ці шаблони для використання, і вони будуть використовуватися замість батьківських.

Коментарі в заголовку style.css потрібні WordPress для ідентифікації теми і відображення її в Administration Panel підменю Design>Themes як встановленої теми, разом з іншими встановленими темами.

Примітка: Коли ви визначаєте батьківську тему, в рядку коментарів Template:, ви повинні використовувати ім'я каталогу теми. Наприклад, для використання теми "Default Wordpress Theme", не пишіть:

Template: WordPress Default

а пишіть:

Template: default

оскільки "default" - це ім'я каталогу батьківської теми.

Файл додаткового функціоналу

Тема може додатково використовувати файл з необхідними функціями для роботи теми, він повинен розташовуватися в каталозі теми і називатися functions.php. Цей файл в основному працює подібно плагіну, і якщо він присутній в каталозі теми, яку ви використовуєте, то він автоматично завантажується під час ініціалізації WordPress. Це вірно як для сторінок адмінки, так і для інших (зовнішніх до адмінки) сторінок. Цей файл можна використовувати наступним чином:

1. Визначення функцій, використовуваних в файлах шаблонів Вашої теми;

2. Налаштування в адмінці Вашої теми, які надають користувачам варіанти налаштувань кольорів, стилів або інших аспектів вашої теми.

Теми, що поставляються з WordPress і встановлені з коробки, мають файл functions.php, який визначає ряд функцій і налаштувань в адмінці теми, тому можете використовувати його в якості моделі. Файл functions.php в основному працює як плагін, розділ Кодексу Function_Reference - найкраще місце, де можна отримати більш детальну інформацію про те, як ви можете використовувати цей файл.

Файли шаблонів теми

Шаблони – це PHP файли, які використовуються для генерації сторінок або їх частин, що потрібні відвідувачам блогу. Розглянемо різні варіанти шаблонів, які можна використовувати як частину теми.

WordPress дозволяє визначати окремі шаблони для різних потреб і частин вашого блогу, але зовсім не обов'язково, щоб всі ці різні шаблони Вашого блогу одночасно виконувалися для генерації сторінок в повному складі. Шаблони вибираються і використовуються для генерації сторінки на основі ієрархії шаблонів Template Hierarchy, в залежності від того, які шаблони доступні в тій чи іншій темі. Розробник теми, може вибрати кількість налаштувань, які хоче реалізувати з використанням шаблонів. Наприклад, в крайньому випадку, можна використовувати тільки один файл шаблону, з ім'ям index.php як шаблон для всіх сторінок, що створюються і відображаються в блозі. Однак найбільш часто використовують різні файли шаблонів для генерації різного виду сторінок, щоб забезпечити максимальну гнучкість налаштувань.

Мінімальна тема WordPress складається з двох файлів:

1. style.css

2. index.php

Обидва цих файли поміщають в папку теми. Файл шаблону index.php є вельми гнучким. Його можна використовувати так, щоб в нього додати всі посилання на заголовок, бічну панель, підвал, зміст, категорії,

архіви, пошук, сторінки помилок й інші необхідні веб-сторінки. В іншому варіанті побудови теми розробіть для теми інші модульні файли шаблонів, кожен з яких візьме на себе частину функціоналу.

Якщо не додать в тему свої власні файли шаблонів, WordPress буде використовувати вбудовані файли шаблонів за замовчуванням. Наприклад, якщо в темі немає файлів шаблонів ні comments.php, ні comments-popup.php, то WordPress буде автоматично використовувати стандартні wp-comments.php і wp-comments-popup.php файли шаблонів, згідно ієрархії шаблонів Template Hierarchy. Ці шаблони за замовчуванням швидше за все не будуть відповідати стилям вашої теми, так що, ймовірно, доведеться розробити свої власні файли шаблонів.

Основні файли, які зазвичай використовують для поділу вигляду сторінок (і які повинні бути в каталозі теми):

- header.php
- sidebar.php
- footer.php
- comments.php
- comments-popup.php

Якщо ці модульні файли шаблонів розроблено, можете їх додати в index.php (головний файл-шаблон), і результат їх роботи з'явиться на сторінці у користувача.

Перелік стандартних файлів шаблонів теми WordPress

Нижче наведено список стандартних файлів шаблонів, що використовуються WordPress. Звичайно, тема може містити й інші файли стилів, зображень або програм. Але майте на увазі, що наведені нижче імена файлів мають особливе значення для WordPress у відповідності до шаблону ієрархій Template Hierarchy.

style.css

Головний файл таблиці стилів. Файл повинен бути в темі обов'язково і обов'язково повинен містити описані вище коментарі в заголовку файлу.

index.php

Головний файл шаблонів. Якщо в своїй темі використовувувати свої шаблони, цей файл index.php обов'язковий.

comments.php

Шаблон коментарів. Якщо його немає, використовується файл comments.php з теми за замовчуванням.

comments-popup.php

Додає спливаюче меню для коментарів на JS. При відсутності викликається comments-popup.php з теми за замовчуванням.

home.php

Шаблон головної сторінки блогу.

single.php

Шаблон сторінки одиночного поста. Використовується, коли запитаний конкретний пост. Для цього й інших шаблонів за запитом використовується index.php, якщо у темі немає відповідного файлу шаблону.

page.php

Шаблон окремої сторінки, використовується для відображення запитаної сторінки Page.

category.php

Шаблон категорії category template. Використовується при запиті категорії.

author.php

Шаблон автора блогу author template. Використовується при запиті автора блогу.

date.php

Шаблон виведення дати-часу. Використовується при запиті дати і часу. Рік, місяць, день місяця, година, хвилина, секунда.

archive.php

Шаблон архіву. Використовується, коли запитані категорія, автор або дата. Цей шаблон може бути перевизначений шаблонами category.php, author.php і date.php для відповідних типів запитів.

search.php

Шаблон результатів пошуку. Використовується після виконання пошуку.

404.php

Шаблон для повідомлення про помилку 404 Not Found. Використовується, коли WordPress не може знайти повідомлення або сторінку, яка відповідає запиту.

Ці файли мають спеціальне значення в WordPress, оскільки вони використовуються для заміни index.php, коли вони є в каталозі теми, відповідно до ієрархією шаблонів Template Hierarchy, і коли надійшов відповідний запит; або ж відповідно до істинності умовних тегів Conditional Tag, коли функція типу is_*(); повертає 'true'.

Наприклад, якщо потрібно відобразити єдиний пост, функція is_single() повертає 'true', і в каталозі активної теми є файл шаблону single.php, цей шаблон використовується для створення сторінки.

Детальніше про роботу з WordPress можна дізнатися на сайті офіційному довіднику по даній CMS-системі [19]: https://codex.wordpress.org/

Завдання:

Створити веб-сайт та тему для CMS-системи WordPress (чи ін. CMS), використовуючи знання, набуті у всіх попередніх лабораторних роботах. Як контент використати дані з лабораторної роботи №1 (згідно варіанту). Як зразок дизайну для теми використати наробки з лабораторної роботи №2 [6].

Контрольні питання до л.р. 7,8:

1) Що таке CMS? Для чого вони потрібні?
2) Які бувають види CMS? Наведіть приклади CMS.
3) Як встановити WordPress?
4) Що представляє собою тема для WordPress?
5) Як створити тему для WordPress?

Список скорочень і спеціальних термінів

ANSI (American national standards institute) - Американський національний інститут стандартів.

API (Application Programming Interface) - інтерфейс прикладного програмування.

CDN (Content Delivery Network) - географічно розподілена мережева інфраструктура, що дозволяє оптимізувати доправлення та розповсюдження контенту кінцевим користувачам в мережі Інтернет.

CMS (англ. Content Management System) - система керування контентом. CMS часто називають словами «движок», «адмінка» або просто: «на чому побудований сайт». CMS дозволяє управляти контентом сайту, змінювати, додавати і видаляти сторінки сайту, а також коригувати їх вміст.

Всі вони діляться на три основних типи:

• Платні CMS (1С-Бітрікс, UMI)

• Безкоштовні CMS (WordPress, Joomla, Drupal, Blogger, OpenCart)

• Самописні CMS. Серед них зустрічаються цікаві рішення, але дуже часто ідея таких систем, полягає в тому, щоб прив'язати замовника сайту до виконавця. Самописні CMS можна назвати умовно-безкоштовними, тому що замовник не платить за їх використання, але при цьому, в майбутньому, він волею-неволею змушений співпрацювати з тією компанією. При виборі CMS потрібно врахувати три головні чинники:

-Популярність. Популярність CMS - це рівень її поширеності в світі.

-Вартість. Переваги платних CMS полягають, перш за все, в підвищеному рівні безпеки. Крім цього, CMS 1С-Бітрікс досить легко об'єднуються з системами бухгалтерського обліку компанії 1С, що дозволяє робити інтернет-магазини, інтегровані в вашу систему обліку. Однак, досить суттєва вартість ліцензій робить Бітрікс важкодоступним для компаній малого і середнього бізнесу.

-Можливості. Платні CMS мають багату гаму додаткових можливостей. Можливості безкоштовних CMS безпосередньо пов'язані з

популярністю. Чим більше поширена CMS, тим більше безкоштовних і платних надбудов для неї існує. Це забезпечує дуже гнучку роботу з сайтом. Можна легко додавати багато зручні для бізнесу функції, починаючи від систем безпеки, захисту від спаму і SEO, і закінчуючи складними надбудовами для розгортання потужних інтернет-магазинів і навіть соціальних мереж.

CSS (Cascading Style Sheets) - ґаскадні таблиці стилів. Це звід стильових описів, тих чи інших HTML тегів (далі елементів HTML), який може бути застосований як до окремого тегу - елементу, так і одночасно до всіх ідентичних елементів на всіх сторінках сайту.

DHTML – це динамічний HTML – комерційний термін, що описує технології, які були введені в четвертій версії Web – браузерів та дозволяли обходити обмеження HTML. DHTML – представляє собою комбінацію Web-стандартів CSS+JavaScript+DOM+XHTML.

Dom (Document Object Model) - об’єктна модель документа, яка дозволяє динамічно змінювати веб-сторінку, застосовуючи мову написання сценарію. DOM має для кожного елемента або об’єкта, що визначається за допомогою атрибуту ID (ідентифікатора об’єкта), функцію JavaScript, що дозволяють керувати властивостями атрибутів об’єкта, що задаються через CSS.

HTML (від англ. HyperText Markup Language - «мова гіпертекстової розмітки») - стандартна мова розмітки документів у Всесвітній павутині. Більшість веб-сторінок містять опис розмітки на мові HTML (або XHTML). За допомогою HTML необхідно розмітити текст, описати за допомогою тегів його структуру.

HTTP (Hyper Text Transfer Protocol) - протокол передачі даних (гіпер-текстових документів).

HTTPS (Hyper Text Transmission Protocol, Secure - протокол захищеної передачі гіпер-текстових документів) - HTTP в сукупності з SSL (Secure Sockets Layer) - протоколом захищених сокетів.

IDE (англ. Integrated Development Environment) - інтегроване середовище розробки.

JS (JavaScript) - об’єктно-орієнтована мова програмування сценаріїв.

PHP – це мова, програмування призначена для створення сайтів, це скрипт-мова , що вбудовується в HTML, що інтерпретується та виконується на сервері а клієнту передається результат роботи.

SEO (англ. search engine optimization) - процес коригування HTML-коду, текстового наповнення (контенту), структури сайту, контроль зовнішніх чинників для відповідності вимогам алгоритму пошукових систем, з метою підняття позиції сайту в результатах пошуку в цих системах за певними запитами користувачів.

TCP (Transmission Control Protocol) - протокол керування передачею – протокол орієнтований на роботу з підключеннями і передає дані у вигляді потоків байтів.

WWW (World Wide Web) - всесвітня павутина.

Контент (сайту) - це інформація, розташована на сторінках сайту. Інформація може бути як текстова, так і графічна, мультимедійна і т.д. Іншими словами контентом називають все те, що можна знайти за допомогою пошукових систем. У будь-якому випадку найпоширенішим контентом є тексти (статті). Саме текстова інформація є лідером за обсягами і кількістю інформації в інтернеті. (SEO-копірайтинг, рерайтінг, тексти, що продають , постинг)

Правила хорошого тону web-дизайну вимагають щоб уявлення Web - сторінки було відокремлено від її структури (Визначення стилів виноситься в окремі файли).

Сайт - це набір різноманітних інтернет-сторінок, які доступні за певними адресами в інтернеті (URL).

Рекомендовані джерела інформації

1 Трегубенко І.Б. Сучасні технології програмування в Т-66 мережах [Електронний ресурс]/ І.Б.Трегубенко, Г.Т.Олійник, О.М. Панаско, 2-ге вид..- Черкаси: ЧДТУ, 2010.-175с.

2 Пасічник О.Г., Пасічник О.В., Стеценко І.В. Основи веб-дизайну Видавництво: Вид група BHV, 2009, 336с.

3 Манако В., Манако Д., Данилова О., Войченко О.П. Основи будування сайтів Видавництво: Шкільний світ, 2006, 120с.

4 HTML Living Standard [Електронний ресурс] – Режим доступу: https://html.spec.whatwg.org/multipage//(дата звернення: 20.01.2023).

5 Resources for developers, by developers. MDN Web Docs. Structuring documents [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Structuring_content/Structuring_documents (дата звернення: 1.09.2025).

6 Методичні вказівки до лабораторних робіт з дисципліни "Web-програмування" для студентів напряму підготовки 6.050101 “Комп’ютерні науки” всіх форм навчання / уклад. Т. В. Федорончак. – Запоріжжя : ЗНТУ, 2017. – 59 с.

7 Web-програмування. Частина 1 (frontend) : навч. посіб. / В. В. Босько, Л. В. Константинова, К. М. Марченко, О. С. Улічев ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький : ЦНТУ, 2022. - 208 с.

8 Client-side form validation [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation (дата звернення: 20.01.2023).

9 Robin Nixon. Learning PHP, MySQL & JavaScript. 6th Ed. 2021, 825p.

10 Dave Crane, Eric Pascarello, Darren James. Ajax in Action. Manning; 1st edition (November 3, 2005) 680 pages.

11 Ричард Вагнер, Аллен Вайк. JavaScript. Київ: ДиаСофт, 2001 р., 464с.

12 Український веб-довідник [Електронний ресурс] – Режим доступу: https://css.in.ua/html/events (дата звернення: 20.01.2023).

13 Бородкіна І.Л., Бородкін Р. О. Web-технології та Web-дизайн : застосування мови HTML для створення електронних ресурсів. Видавництво: Ліра До, 2020, 212с.

14 PHP Documentation Group [Електронний ресурс] – Режим доступу: https://www.php.net/manual/en/intro-whatis.php (дата звернення: 20.01.2023).

15 jQuery Validation Plugin [Електронний ресурс] – Режим доступу: https://jqueryvalidation.org/documentation/ (дата звернення: 20.01.2022)

16 Web-програмування. Лабораторний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 125 «Кібербезпека» та 113 «Прикладна математика» / А. Ю. Шелестов, Н. М. Куссуль; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 1047 Кбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 61 с.

17 WordPress Codex [Електронний ресурс] – Режим доступу: https://codex.wordpress.org/ (дата звернення: 21.01.2022).

Додаток 1
Приклад оформлення звіту з лабораторної роботи з Web-програмування

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

ЗВІТ
ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ №__
з дисципліни «Web-програмування»

Тема: **Вписати тему лабораторної роботи**

Виконав: студент гр. _-__-_
Мороз Д.І.
Перевірив:викладач
(Вказати ПІБ викладача)

Кропивницький 202_

Мета: Познайомитися із синтаксисом, основними тегами й атрибутами мови html.

Знати: Структуру html-документу, основні теги мови й методи форматування тексту.

Вміти: Компонувати текстову інформацію при створенні гіпертекстових документів з використанням простих текстових редакторів. Формувати гіпертекстові посилання й списки й створювати на цій основі зв'язані гіпертекстові сторінки.

Завдання:

Створити статичний сайт, що містить мінімум 6 web-сторінок з тематики відповідно до свого **варіанту**.

Обов’язково використайте теги:

- форматування тексту,
- створення списків,
- таблиць,
- посилань,
- додавання малюнків,
- елементи розмітки веб-сторінок,
- аудіо та відео.

Варіант 1

Сайт – тлумачний словник з ілюстраціями

Хід роботи:

1 Лістинг лр1.html

```
<html>
<head>
<title>lr1 Прізвище студента </title>

</head>
<body bgcolor="#C0C0C0" >

<h1><p align=center><font color="blue">Тлумачний словник онлайн </font></p></h1>
<h4><p><i>В даному розділі нашого порталу ви зможете вивчити тлумачення слів ігрової тематики.</i></p></h4>
<h5>В цьому Вам допоможе <b>український тлумачний словник</b></h5>
<h2>Словник термінів</h2>
<p>В нашому <b>словнику термінів</b> Ви знайдете тлумачення термінів, понять і значень слів.
Словник містить як слова сучасного тлумачного словника так і слова відомі давно.
Спершу будуть представлені тільки українські слова, але незабаром ми почнемо наповнювати англійський тлумачний словник.
Якщо ви хочете скачати тлумачний словник української мови - тоді зверніть увагу, що викачані словники не оновляються.
```

А в великих онлайн тлумачних словниках можна завжди знайти щось нове, щось що ви довго шукали.

All <tt>good</tt>!

<h3>Тлумачення слів<h3>

Нижче виберіть одну з букв для того, щоб знайти те чи інше тлумачення слів.

<h1>Українська абетка:<h1>

А

Б

В

Г

Д

Е

……..

<img src="img/mini_7.jpg" alt="Це словник!">

</body>

</html>

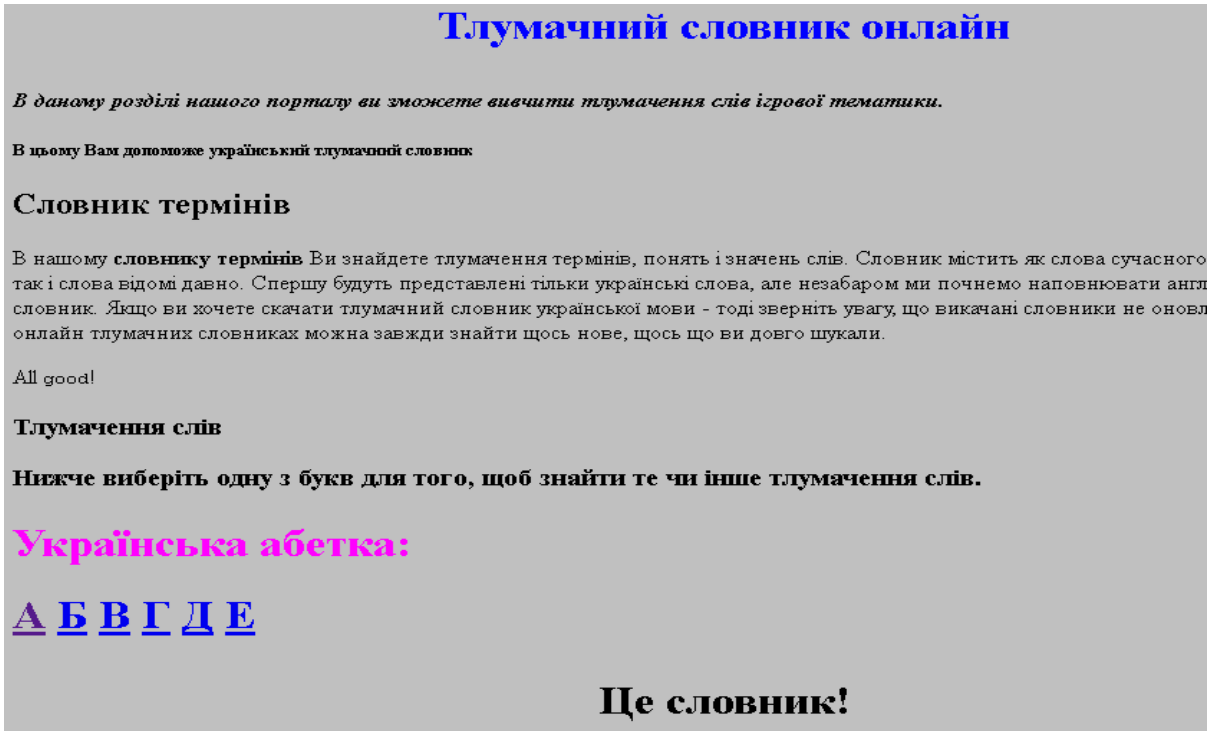


Рисунок 1 - Головна сторінка сайту

2 Лістинг А_лр.html

<html>

<head>

<title>А_lr1 Прізвище студента </title>

</head>

<body bgcolor="#C0C0C0" >

<h1> Тлумачний словник онлайн </h1>

```
Слова на літеру А
<p><font size="+2">В </font>списку нижче <small>представлені слова, які починаються на літеру</small> <big>А.</big></p>

Виберіть одне із слів:
<p><font face="Monotype Corsiva, Arial" size="7" color="lime">Азарт</font></p>
<p><center><img src="img/tigr.jpg" alt="Ось який азарт!"></center></p>
<div align=center><font color="#FF00FF">All <tt>good</tt>! </font></div>


<a align=center href="лр1.html">Повернутися на головну сторінку</a>

….
</body>
</html>
```

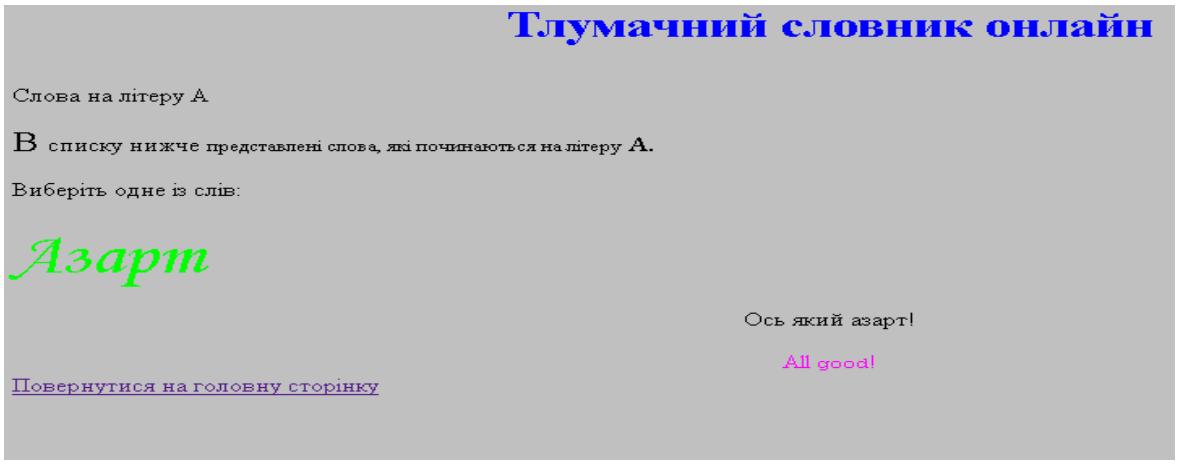


Рисунок 2 – Перехід за посиланням

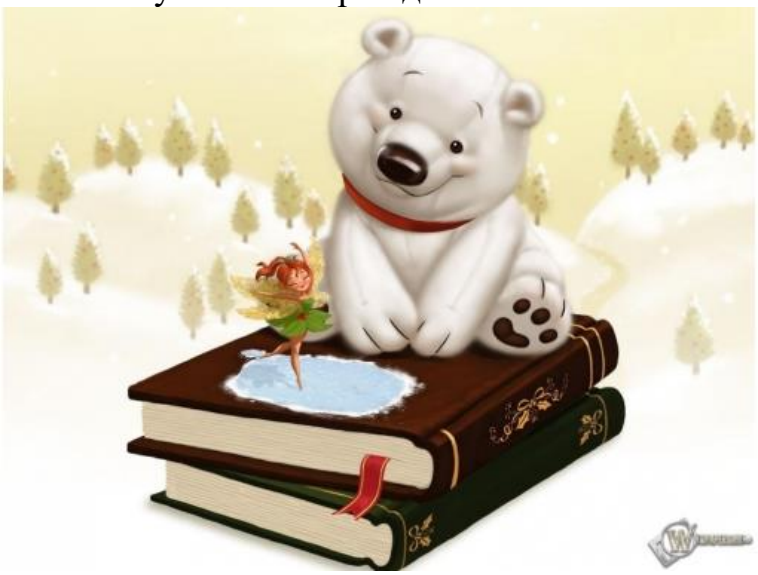


Рисунок 3 – Файл mini_7.jpg

3. Інші файли організовані аналогічним чином

Таблиці:…..
Списки:……
….

Контрольні запитання:

1. Що таке HTML?
2. Яким чином забезпечується форматування тексту в HTML-документі?
3. Яка мета розробки HTML5?
4. Який синтаксис HTML?
5. Структура HTML-документа?
6. Яким чином здійснюється взаємодія між HTML-документом і сервером?
7. Які текстові редактори для роботи з HTML Ви знаєте?
8. Що означає DOCTYPE у HTML-документі?
9. Що таке заголовок HTML-документа?
10. Що таке тіло HTML-документа?
11. Для чого служить текст між тегами <title>?
12. Яким тегом забезпечуються посилання на інші документи HTML і його атрибути?
13. Яким тегом визначаються параграфи?
14. Яким тегом визначаються заголовки?
15. Яким тегом визначається найбільший заголовок?
16. Яким тегом визначається найменший заголовок?
17. Що визначається тегом <img>?
18. Перелічити HTML - елементи форматування тексту.
19. Які теги списків ви знаєте?
20. Що визначає тег <table>?
21. Яким тегом визначається заголовок таблиці?
22. Для чого використовують тег ?
23. Для чого використовують тег ?
24. Що визначають <header>, <nav>, <section>, і <footer> елементи?

Відповіді на контрольні запитання:

1. HTML - це……
2. …….
3. …….
4. ……
   ……

24…….

Висновок: Виконуючи …..

Дякую за увагу!