Міністерство освіти та науки України
Кіровоградський національний технічний університет
Механіко-технологічний факультет
Кафедра програмування та захисту інформації

Методичні вказівки

до виконання лабораторних робіт з дисципліни «Web-програмування»
для студентів денної та заочної форми навчання
спеціальностей 123 «Комп'ютерна інженерія» та 125 «Кібербезпека»

Розробили:
доцент Мелешко Є.В.,
викладач Константинова Л.В.

Кропивницький 2016

Web-програмування : метод. вказ. до викон. лаб. робіт студ. ден. та заоч. форми навч. спец. 123 "Комп'ютерна інженерія" та 125 "Кібербезпека" / М-во освіти і науки України, Кіровоградський нац. техн. ун-т, каф. програмування та захисту інформації; [укл. Є. В. Мелешко, Л. В. Константинова]. – Кропивницький : КНТУ, 2016. – 81 с.

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

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

Затверджено на засіданні кафедри програмування та захисту інформації протокол №2 від 31.08.2016

Схвалено на засіданні методичного семінару кафедри програмування та захисту інформації протокол № 1 від 29.08.2016 року

Вступ

Методичні вказівки до лабораторних робіт з дисципліни «Web-програмування» для студентів спеціальностей 123 «Комп'ютерна інженерія» та 125 «Кібербезпека» містять в собі інструкції виконання, завдання, теоретичний матеріал до лабораторних робіт і призначена для студентів ВУЗів, що вивчають комп'ютерні науки та можуть бути корисними для школярів, вчителів та просто особистостей, що цікавляться Web-програмуванням.

В методичних вказівках 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 рекомендованої літератури.

Дані методичні вказівки починають знайомити з основними засобами HyperText Markup Language (Html), **C**ascading **S**tyle **S**heets (CSS), JavaScript, 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, 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

Дані методичні вказівки містять у собі 7 лабораторних робіт за наступними темами:

Теми лабораторних робіт	Години
1. **Основні засоби html**	2
2. **Вивчення й практичне застосування стилів CSS**	2
3. **Застосування JavaScript при створенні Web-Сторінок**	2
4. **Застосування JavaScript при створенні Web form**	2
5. **Підтримка інформаційних мультимедіиа потоків**	2
6. **Вивчення й практичне застосування форм HTML та елементів керування**	2
7. **Застосування рядкових функцій в PHP**	2
Всього:	14

Форма підсумкового контролю: залік.

Максимальну кількість балів студент може одержати у випадку відвідування всіх лекцій, лабораторних занять, виконання всіх завдать з лабораторних робіт, що

демонструється викладачу і підтверджується звітом, виконання і захисту самостійної роботи у встановлений термін, проходження контролю.

При виконанні і 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 питання, 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 питання. Приклад оформлення звіту з лабораторної роботи дивись в Додатку 1.

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

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

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.

Лабораторна робота №1

Тема: Основні засоби html.

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.

Теоретичні відомості

HTML (англ. HyperText Markup Language) – стандартна мова розмітки гіпертекстових документів (веб-сторінок) в Інтернеті.

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

Ми будемо розглядати **HTML5**, дана версія була опублікована в 2014 році.

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 і стандартів для мови HTML займається організація W3C (англ. World Wide Web Consortium – консорціум Всесвітньої павутини). Переглянути інформацію про html5 на їхньому веб-сайті можна за наступним посиланням:

https://www.w3.org/TR/html5/

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 Notepad (або ін. 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.

Крім Notepad 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 по базі 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.

Приклад 3. Приклад структури сайту (стрілки ілюструють один з можливих варіантів посилань сторінок одна на одну):

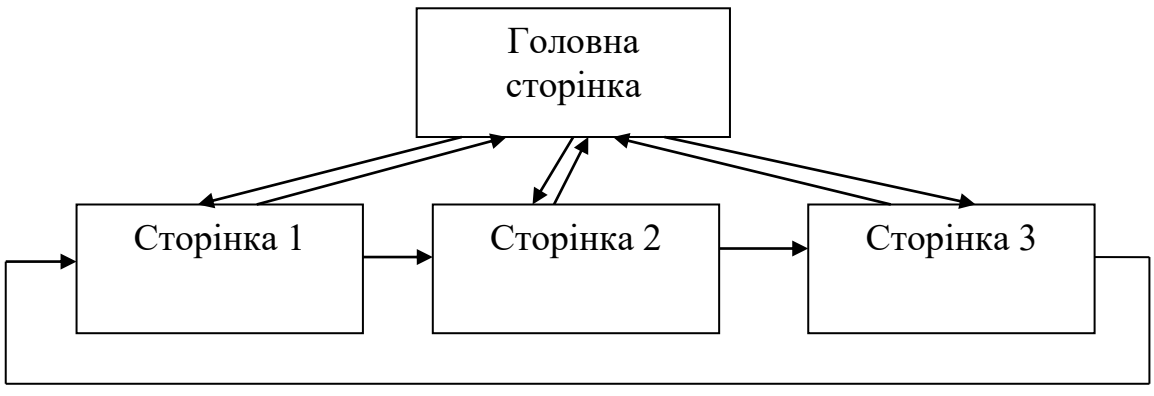


Рисунок 1

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 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">

Таблиця 1 – 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 4. 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 2 – 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

Тег	Опис
	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

Вибір кольору шрифта html - сторінки

Найм. коліру	Код
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 в браузері:
червоний цвет Синій цвет	Червоний колір Синій колір

Таблиця 3 – Html-теги списків

Тег	Опис
	Ненумерований список
	Нумерований список
	Елемент списку

Приклад 5. Приклад ненумерованого списку.

```
<!DOCTYPE html>
<html>
<body>
<h2>Unordered List with Default Bullets</h2>
<ul>
 <li> Яблука </li>
 <li> Банани </li>
 <li> Лимони </li>
 <li> Мандарини </li>
</ul>
</body>
</html>
```

Таблиця 4 – Html-теги таблиці

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

Приклад 6. 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>
```

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

Рисунок 2 – Результат роботи коду з прикладу 6

Таблиця 5 – 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.

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:

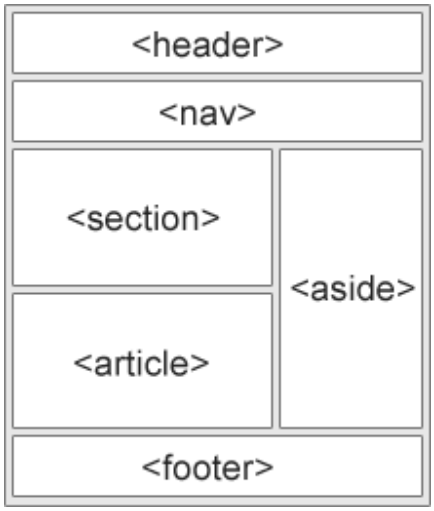


Рисунок 3

Таблиця 6 – Елементи розмітки веб-сторінок в HTML5

Тег	**Опис**
Header	Заголовок документа або розділу
Nav	Контейнер для навігаційних посилань
Section	Розділ в документі
Article	Незалежна самодостатня стаття
Aside	Бічна панель
Footer	Нижній колонтитул документа або розділу

Таблиця 7 – 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

Завдання:

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

Обов'язково використати:

- теги форматування тексту (розмір, стиль, колір, фон),
- теги для групування елементів,
- створення списків,
- створення таблиць,
- посилань,
- додавання малюнків,
- елементи розмітки веб-сторінок <header>, <nav>, <section>, і <footer>.

Не використовувати CSS.

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

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?

Лабораторна робота №2

Тема: Вивчення й практичне застосування стилів CSS.
Мета: Познайомитися із стилями CSS, синтаксисом, основними елементами.

Знати: Структуру HTML документа, основні підходи стилів CSS.
Вміти: Компонувати текстову інформацію при створенні гіпертекстових документів з використанням простих текстових редакторів.

Теоретичні відомості

Стилі CSS

Керовані WEB-дизайнером стилі з'явилися в HTML відносно недавно й відразу наблизили HTML до сучасних засобів підготовки публікацій.

Стиль - це можливість, один раз описавши правила форматування ділянки документа, використовувати потім ці правила простим посиланням на назву стилю.

Стилі дозволяють легко вирішити проблему "фірмового" оформлення сайтів. Е можливість не форматувати тисячі об'єктів вручну, а просто описати стиль і посилатися на нього там, де потрібно. Тоді Ви змогли б, змінивши форматування стилю в одному єдиному місці, домогтися зміни форматування відповідних об'єктів у всьому сайті.

Насправді стилі як такі були присутні в HTML із самого початку. Наприклад, теги **<H1>**, **<H2>**, **<BLOCKQUOTE>**, і т.д. є ні що інше, як стилі. Однак WEB-дизайнер не міг управляти ними. Наприклад, не можливо було вказати, що всі заголовки рівня **<H1>** повинні бути в 20 пунктів висотою й синім кольором. З CSS така можливість з'явилася.

Крім того, стилі дають цілий ряд нових, раніше не доступних, засобів форматування документа. І, нарешті, стилі - річ динамічна! Це означає, що зовнішній вигляд документа можна змінювати із програм-сценаріїв прямо на льоту *(наприклад, у відповідь на те, що користувач завів покажчик миші на картинку або на посилання)*.

На жаль, існує деякий розбрід у реалізації всіх цих можливостей у найбільш популярних браузерах. Фірма **Netscape Communications**, наприклад, до останнього часу випливала своєму унікальному стандарту **JSSS** (**J**ava**S**cript **S**tyle **S**heets), що не підтримувався ніякими іншими клієнтами. Компанія **Microsoft** просувала стандарт **CSS** (**C**ascading **S**tyle **S**heets), який підтримувався WWW консорціумом. Нарешті точка зору консорціуму взяла гору й компанію *Netscape Communications* оголосила про підтримку **CSS**, починаючи з **Netscape Navigator 4.04.** Проте реалізація **CSS** у різних браузерах трохи різна. Якоюсь мірою в цьому може допомогти таблиця сумісності, складена людьми, які просто взяли, і все зрівняли. Однак, кращий спосіб не помилитися, це **перевіряти**, як виглядають сторінки в **різних** браузерах перед тим, як "випустити їх у світло".

Отже, перелічимо по пунктах, що ми можемо робити, використовуючи стилі:

- 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, і інші ефекти;
- управляти форматуванням сторінок під час друку HTML документа;
- створювати різні ефекти для тексту й зображень (тіні, димку, і т.д.).

Стилі: **Втавка в документ**

Існує три способи вставити стиль в HTML документ.

- посиланням з HTML документ на спеціальний 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 вид;
- визначити всі стилі, що використовуються на початку документа, а потім використовувати. При такому підході Ви можете змінювати стилі на початку документа (один раз) і Ваші зміни відіб'ються у всіх місцях документа, де ці стилі використовуються;
- 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.

Звичайно при розробці сайту доводиться використовувати всі три підходи. При цьому, у випадку конфлікту, найвищий пріоритет у властивостей, вставлених безпосередньо в теги, потім - у блоків стилів, вбудованих у сторінку, і найнижчий - у блоків, на які є посилання.

Як резюме: якщо потрібно однаково відформатувати дві й більше ділянки тексту, то Вам належить визначити стиль. Двічі писати однакове форматування - ***дуже дурний тон.***

Стилі: **Синтаксис**

Синтаксис визначення стилю дуже простий. Пишеться ім'я стилю, потім у фігурних дужках перераховуються властивості, розділені символом ";". Перед закриваючою фігурною дужкою ";" не ставиться. Властивостей може бути дуже багато, пізніше ми розглянемо деякі з них.

Кілька описів стилів утворять блок стилів. Наприклад, визначимо блок, що задає червоний колір і кегль в 36 пунктів для 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
```

}

Стилі: **Вставка в документ**

Стилі можна використовувати в документі трьома способами: визначаючи посилання на них, вставляючи в початок документа або вставляючи властивості стилю прямо в теги.

Вставка за допомогою посилання

Для того, щоб зробити в документі посилання на окремий файл стилів, варто записати:

<LINK REL=STYLESHEET TYPE="text/css" HREF="URL">

Де URL - адреса файлу з визначенням стилів (стандартне розширення - CSS).

Наприклад,

<LINK REL=STYLESHEET TYPE="text/css" HREF="style2.css">

у файлі **style2.css** утримується наведене вище визначення **<H1>** і **<H2>**.

Вставка безпосередньо в документ

Для того, щоб вставити блок стилів прямо в документ, потрібно укласти його в тег **<STYLE> ... </STYLE>**. Наприклад, це може виглядати так:

```
<STYLE TYPE="text/css">
<! ---i
H1 {
   font-size: 36pt;
   color: red
}
H2 {
   font-size: 18pt;
   color: blue
}
---i>
</STYLE>
<H1>Заголовок рівня 1</H1>
Звичайний текст<BR>
Звичайний текст<BR>
<H2>Заголовок рівня 2</H2>
Звичайний текст<BR>
Звичайний текст<BR>
<H2>Ще заголовок рівня 2</H2>
Звичайний текст<BR>
Звичайний текст<BR>
<H1>Ще заголовок рівня 1</H1>
Звичайний текст<BR>
Звичайний текст<BR>
<H2>Знов заголовок рівня 2</H2>
Звичайний текст<BR>
Звичайний текст<BR>
```

Результат повинен бути таким же, як і минулого разу.

Важливе зауваження:

Хоча, тут цього не зроблено, існують ***дуже поважні причини*** *завжди вставляти* ***<STYLE> ... </STYLE>*** *у* ***заголовну частину документа****. Тобто між тегами* ***<HEAD>*** *і* ***</HEAD>****. Намагайтеся завжди робити саме так.*

Вставка властивостей у теги

Властивості стилів можна вставляти прямо у відповідні теги. Розглянемо такий приклад:

```
<P>Цей абзац відформатований за замовчуванням. Цей абзац
відформатований за замовчуванням. Цей абзац відформатований
<P STYLE=" margin-left: 2cm; margin-right: 2cm">
Цей абзац має відступи праворуч і ліворуч по 2 сантиметри.
Цей абзац має відступи праворуч і ліворуч по 2 сантиметри.
<P STYLE=" font-size: 1in; color: red; font-style: italic">
Цей абзац написаний червоними похилими буквами висотою в один дюйм.
```

Елементи оформлення можна вставляти практично в будь-який тег. Наприклад, у тег **<DIV STYLE="...">** вони будуть ставитися до описуваної секції, у тег **<UL STYLE="...">** - до списку, у тег **<LI STYLE="...">** - до елемента списку й т.д.

Існує новий спеціальний тег **<SPAN>**, він у всім аналогічний **<DIV>** з тої лише різницею, що ніяк не впливає на розбивку документа на рядки. У такий спосіб його можна використовувати для завдання стильових особливостей окремих слів і навіть букв.

Класи

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

Давайте визначимо, що всі теги **<P>** у документі будуть задавати абзаци, написані шрифтом розміром в 20 пунктів блакитного кольору. Потім визначимо, що в нас ще будуть спеціальні абзаци, такі ж, але зрушені ліворуч на пів дюйма. І, нарешті, у нас будуть абзаци ще й третього виду - такі ж, як перші, але написані похилими буквами. Рішення цього завдання може виглядати, наприклад, так:

```
<STYLE TYPE="text/css">
<! ---i
P { font-size: 20pt; color: blue}
P.m { margin-left: 0.5in}
P.i { font-style: italic}
---i>
</STYLE>
<P>
Це звичайний абзац.
<P CLASS=m>
Це зрушений абзац.
<P CLASS=i>
Це похилий абзац.
Подивіться, як це виглядає.
```

Зрозуміло, так можна оформляти не тільки спеціальні абзаци, але й спеціальні заголовки, розділи документа й т.д.

Посилання

Особливий випадок представляють посилання. Взагалі можна визначити стиль для тега **<A>**, і посилання будуть виглядати так, як ми хочемо. Однак, існують ще й посилання, на яких ми вже побували, і активні посилання. Для них спеціально визначені класи: A:link - посилання, A:visited - посилання, на якій уже побували й A:active - посилання активні в цей момент. Наприклад, зробимо так, щоб посилання на нашій сторінці були висотою в 20 пунктів, перекреслені. Ті посилання на яких ми не були - зелені. Ті, на яких були - червоні.

Рішення:

```
<STYLE TYPE="text/css">
<! ---i
A:link { font-size: 20pt; color: green; text-decoration: line-through }
A:visited { font-size: 20pt; color: red; text-decoration: line-through }
---i>
</STYLE>
<A HREF="fictive">Фіктивне посилання, на ньому ми ще не були</A>
<P>
<A HREF="http://www.botik.ru/~ks/ip/scripts/listany.cgi?file=lec2.html&code=win">На цьому посиланні ми вже були</A>
```

Спадкування

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

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

Таким же чином, розділ, вкладений в інший розділ, успадковує формати розділу що його охватує, але може щось перевизначити для себе.

Розглянемо приклад:

```
<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 всі сдвинуто на 1 сантиметр ліворуч від першого розділу.
    <DIV STYLE="color: blue; margin-left: 1cm">
    Це третій розділ усередині другого. Усе успадковано від другого,
    але букви блакитні й всі сдвинуто на 1 сантиметр ліворуч від другого розділу.
    </DIV>
    Це триває другий розділ (третій закінчився)
  </DIV>
Це триває перший розділ (другий закінчився)
</DIV>
```

Дивіться, 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.

У цьому й полягає зміст спадкування стильового оформлення в **CSS**. Наприклад, використання нового тега **<SPAN>**.

Багато з людей люблять виділяти товстим фломастером місця, що сподобалися їм, у газетах і книгах. При цьому текст на зафарбованій ділянці продовжує проступати, але фон змінює колір. Спробуємо використовувати такий прийом виділення головних думок. Оскільки ми виділяємо не в газеті, скористаємося своєю перевагою й, ще трохи зробимо товстішим шрифт у виділених ділянках. Рішення може бути таким:

```
<STYLE TYPE="text/css">
<!--
SPAN.NB {background: yellow;font-weight: 700}
-->
</STYLE>
<BR>
По некоторым оценкам, благодаря WWW может быть
поставлено под угрозу существование правительств.
С изобретением технологии, обеспечивающей
<SPAN CLASS="NB">истинную демократию</SPAN>,
в отличие от
<SPAN CLASS="NB">существующей представительской
демократии</SPAN>, весь процесс управления может
существенно, если не сказать коренным образом,
измениться.
<P ALIGN=RIGHT>Том Армстронг
```

Властивості

Дотепер ми користувалися властивостями стилів такими як font-**size**, **text-decoration**, **color** і ін. ніяк не визначаючи які взагалі властивості бувають. Насправді їх дуже багато.

На практиці варто обережно користуватися цією таблицею. Далеко не все працює скрізь, а дещо не працює просто ніде. Проте це - стандарт, і те, що не працює зараз, заробить у нових версіях браузерів.

Тепер, коли Ви розумієте, що таке стилі, як і навіщо їх використовувати, Ви можете просто брати з таблиці властивості й уписувати їх у свої визначення стилів. Особливих проблем *(крім несумісності браузерів)* виникати не повинно.

Володіння цією технікою дозволить Вам заощадити масу часу й сил при розробці більших сайтів.

CSS 2

Все, про що ми тут говорили, належить до **CSS 1**. У цей час існує чорновий варіант нового стандарту **CSS 2**. Там додане керування печаткою, фільтри для різних ефектів, а, головне - пряме керування розташуванням документа на екрані. На жаль, це поки не стало стандартом і реалізовано дуже по різному на різних браузерах.

Завдання:

На основі HTML сторінок лабораторної роботи №1, створити сторінку, на якій повинні бути: колір тла сторінки, атрибути шрифту, і заголовок сторінки, вибрати відповідно тематики сторінок використовуючи CSS.

Заголовок сторінки – «**Практичне застосування елементів CSS».**

Всі формати стилів робити за допомогою CSS.

Контрольні питання:

1) Що представляє собою CSS?
2) Як використовується CSS?
3) У яких випадках який спосіб застосування CSS кращий?
4) Який синтаксис визначення стилю?
5) В чому полягає ідея спадкування форматів у документі?
6) Що означає клас і як його визначити?

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
Основи JavaScript

Вставка в код сторінки

JavaScript являє собою повністю интерпретуєма мова опису сценаріїв із програмним кодом скрипта, що поставляється користувачеві у відкритому виді.

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 їсти 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 скриптов в окремих файлах. Цей підхід найбільш прийнятний при використанні скриптов досить великого обсягу, а також для зберігання службових процедур. Якщо ми зберігаємо наш скрипт у файлі з ім'ям filel.js, те 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 відповідному цьому випадку варіанта вмісту сторінки.

Використання скриптов 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++, досить розповсюдженим на Заході мовою програмування. 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 строкових вираженнях можуть використовуватися 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 і 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. Описана нами змінна name очевидно буде мати строковий тип. У деяких випадках ми можемо 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 коми наприкінці виражень JavaScript. Її застосування в багатьох випадках необов'язково, але все-таки крапку з коми краще ставити.

Зміна значень змінних відбувається за допомогою використання оператора присвоювання =.

Namel = "Вираження!";"Вираження!" може містити в собі інші змінні й оператори дій над ними. Більшість операторів JavaScript подібні до відповідних операторів мови C++:

(+) додавання, зчеплення рядків;
(-) вирахування;
(*) множення;
(/) розподіл;
(%) залишок від целочисленного розподілу;
(++) инкремент;
(--)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 вправо;

(<<) зрушення вліво.

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; вираження А буде обчислено.

(2=3) && (A) результат false; вираження А 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 якими виділяється пам'ять. При цьому розмірність і тип елементів масиву може мінятися в процесі виконання програми, тобто в мові JavaScript ви не зштовхнетеся з помилкою "Вихід за межі масиву". При звертанні до неіснуючих елементів (раніше не заданим), їхній уміст 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 строкових 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, як в інших мовах програмування:

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

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

JavaScript надає нам також кілька цікавих 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 прикмет наступне значення: "A -> B -> C"

Введення/виведення в JavaScript

Будь-яка мова програмування немислима без операторів виведення. JavaScript не є виключенням. Виведення даних на екран може відбуватися різними способами. При цьому оператори виводу оптимізовані для найбільш зручного їхнього 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, умістом 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 вікна оператора Alert досить зручно використовувати для контролю значень змінних на тому або іншому етапі виконання програми, тобто при налагодженні. Приведемо код HTML-Документа, що приводить до появи такого вікна.

Виведення на екран з використанням вікна оператора 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>
```

Як вправу рекомендуємо перевірити результат роботи перерахованих у попередньому пункті операторів JavaScript.

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 про, що є методом об'єкта document, що описує поточний документ, завантажений у дане вікно.

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

Іноді при виводі на екран засобами JavaScript виникають проблеми з кодуванням російського шрифту. Щоб ці проблеми не турбували вас, необхідно, щоб у настроюваннях браузера була обрана опція автоматичного визначення кодування документа.

Вивід у тіло документа засобами JavaScript ;

```
<HTML>
<ПОЗНАЧКА content="text/html; charset= windows-1251" http-equiv= Content-Type>
<BODY bgcolor= "#aa88aa" >
<H2> Це вивід засобами HTML <BR>
<SCRIPT>
document.write ("А це працює оператор Write ()");
document.writeln ("</H2> <H3> <FONT color=#fffffff> " +
"<center> Поміняємо 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 користувачеві буде виведене запрошення на введення, що втримується у виразі "Текст повідомлення". Після вводу введене 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, нажавши кнопку ОК, користувач введе це значення, а, натиснувши будь-яку іншу кнопку, може приступитися до вводу 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>
```

Ввод значень булевского 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 заданим питанням і двома кнопками . Залежно від натискання користувачем тої або іншої кнопки змінна b одержить або значення true (кнопка ОК), або false (кнопка Cancel, у русифікованої ОС Windows - 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. Розглянемо їхній синтаксис.

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

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

Вираження "умова" повинне бути булевского типу. Це може бути комбінація операторів відносини або результат дії оператора confirm. Оператор циклу з кінцевим числом повторень повторює певну послідовність дій задане число раз.

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

Тут необхідне використання целочисленної змінної, початковим значенням якої буде "вираження". Цикл буде повторюватися, поки буде щирим "умова". При цьому при кожній ітерації циклу над змінної-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. Синтаксис циклу наступний:

while ("умова") { Послідовність дій }

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

Для примусового виходу із 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 управляти як вікнами браузера, так і їхнім умістом. Команда document. clear очищає поточне вікно. Це може придатися при виводі даних у тіло документа. Команда window.close закриває поточне вікно браузера. Команда Window.open (ur 1, місце розташування, атрибути) відкриває Документ за адресою uri у вікні або фреймі, заданому у вираженні "місце розташування". Параметри вікна описані у вираженні "Атрибути". Дана команда широко використовується для відкриття супутньому основному вікну вікон з рекламою.

(*Приклади скрипт коду*):

Занесення у вибране:

```
<a href="#" onClick="window.external.addFavorite('http://www.ваш_сайт.ru/', '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.ru">Посилання 1</Option>
  <Option value="http://www.ваш_сайт.ru">Посилання 2</Option>
  <Option value="http://www.ваш_сайт.ru">Посилання 3</Option>
  <Option value="http://www.ваш_сайт.ru">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 стартової</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 баннера

```
<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) Визначити тип і версію браузера.

2) Скрипт утримуючий обчислення й виведення на екран основних математичних операцій (+,-,/,*). Для виводу різниці й суми скористатися методом write об'єкта document, для виводу ділення й множення - методом alert об'єкта window.

3) Використовуючи оброблювачі подій OnMouseOver і OnMouseOut забезпечте збільшення зображення при приміщенні на нього курсору й повернення від збільшеної копії до зменшеного послу зняття курсору.

4) Скрипт утримуюче поле типу "radio" (мінімально 5 варіантів вибору) . Забезпечити залежно від вибору поля вивід на екран одного з п'яти зображень. Забезпечте 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>).

5) Забезпечити статичне меню переходу на інші сторінки. Використовуючи оброблювачі подій, передбачити зміну кольору гіперпосилань при приміщенні на них курсору.

6) Реалізація меню переходу на основах елемента вводу select. В HTML-Документі оголосити форму з одного елемента select. Події onChange цього

елемента привласніть користувальницьку функцію, що здійснює перехід на запитувану користувачем сторінку.

7) Реалізація багатомовного інтерфейсу. 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 - RUS,UA).

8) Реалізувати скрипт, що випадковим чином виводить інформацію дня на екран (цитата, анекдот і.т.п.). Інформацію помістити в масив і винести в окремий файл скрипта (*.js).

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) Що представляють собою оброблювачі подій?
5) Що представляє собою ієрархія класів?

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 (JavaScript і форми HTML):

Об'єкт form

Як створити кілька форм? Справа в тому, що кожна форма буде мати своє власне ім'я (для цього використовується значення атрибута name елемента <form>). А виходить, одержати доступ до елементів форми буде нескладно. 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, що втримуються 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 усі імена форм записати в масив, тим самим організувавши незалежний доступ до кожної з них (formName[x]).

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 ().

Обробка помилок на формі за допомогою JavaScript

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

Почнемо з невеликого приклада. У лістингу закодована ціла сторінка, що дозволяє її відвідувачеві вводити дані. Скрипт здійснює перевірку правильності вводу поштового індексу.

Лістинг *Перевірка даних у формах за допомогою JavaScript*

```
<!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>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 введений 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 перевірить кожну позицію й установить, чи дійсно в ній перебуває цифра від 0 до 9.

Розширена функція zipCheck( ) може виглядати в такий спосіб:

```
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 й 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 із цифр!"); return(-l);

Якщо ж вираз if приймає значення «неправда», то команди, що втримуються 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 деяких випадках з подивом можна виявити, що зовсім і не потрібний доступ до каталогу CGI сервера. У наступному лістингу показаний приклад електронної форми і 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 другу форму. При натисканні на кнопку SEND IT! (Відправити!) вся накопичена інформація відправляється по електронній пошті.

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

Значення перемикачів зберігаються в спеціальних масивах, а отже, ми можемо 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;
      }
}
```

У свою чергу, всі перемикачі теж зберігаються в масиві об'єктів, імена яких визначаються за допомогою атрибута name, що вказується при їхньому створенні. У нашому прикладі name="where". З кожним екземпляром where можна використовувати властивість checked для того, щоб визначити, у якому положенні 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 положення перемикача. Коли він його знаходить, цикл for завершується (командою break), а зі значенням виробляються певні дії. У цьому випадку воно додається до рядка result_str.

Друге, що показано в цьому 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 якомусь масиві або використовувати цикл для їхнього перебору. До кожного прапорця потрібен індивідуальний підхід, у кожного з них особисто потрібно з'ясувати, чи є він 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;
```

Завдання:

На основі HTML сторінок лабораторних робіт №1, №2, №3 і на основі приклада 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 (не менш 4 полів вибору), Checkbox (не менш 5 полів вибору), Text Area (не менш 5 вводу), а також кнопку підтвердження введення даних і кнопку очищення полів введення даних. При натисканні на кнопку форми реалізувати проходження перевірки уведених 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 з пошти. При натисканні на кнопку форми, сформувати лист поштовому клієнтові операційної системи за адресою My_Support@mail.ru.

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 присутні на формах?
5)Як перевірити, чи є символи цифровими?

Лабораторна робота №5

Тема: Підтримка інформаційних мультимедіа потоків
Мета: Познайомитися зі сполученням веб-сторінок і мультимедійних елементів
Знати: Структуру HTML документа, основні елементи мови JavaScript.
Вміти: Компонувати текстову інформацію при створенні гіпертекстових документів з використанням простих текстових редакторів.

Теоретичні відомості:

Додавання гіперпосилань

Гипермедіа-посилання мало чим відрізняється від звичайного гіперпосилання. Але одну відмінність можна знайти: вона вказує не на документ, що відображається у вікні браузера, а на якийсь файл. Браузер повинен розпізнати його й завантажити відповідний допоміжний додаток. Останній і виводить мультимедійний файл у належному вигляді.

Гипермедіа-посилання виглядають зовсім так, як будь-які інші. Усередині якоря може втримуватися текст, активне зображення, посилання може являти собою й гарячою зоною карти посилань. Не слід забувати, що вводити потрібно URL не HTML-документа, а якогось файлу. Різниця невелика. От приклади посилання на звуковий файл у форматі MP3:

```
<a href="media/greeting.mpз">Поздоровлення</a>
```

Коли користувач клацає на посиланні, файл завантажується на його комп'ютер. Після цього браузер повинен запустити той додаток, що асоційовано з даним форматом. Якщо такого додатку немає або він не налаштований, можна просто зберегти файл на жорсткому диску для майбутнього використання.

Існує ряд форматів, які більшість браузерів можуть обробляти.

- Графічні, наприклад .GIF, JPEG і .PNG. Всі вони обговорювалися раніше.
- Звукові, зокрема .MIDI і .WAV. Звичайно вони використовуються для створення фонового озвучування сайтів.
- Звичайний текст із розширенням .TXT.

Коли користувач клацає на посиланні, тип файлу, зазначений у ньому, зіставляється з конфігурацією браузера. Якщо в налаштуваннях для якого-небудь

типу задано, що повинен запускатись допоміжний додаток або файл повинен бути негайно збережений на диску, то відповідна дія й виробляється. Якщо ж даному формату нічого не зіставлене, з'явиться діалогове вікно, у якому браузер буде запитувати, що робити з файлом далі.

Впровадження мультимедійних елементів

Іншим підходом до включення мультимедіа у свої сторінки є безпосереднє впровадження елементів. Це сильно нагадує додавання зображень за допомогою елемента <img />. У загальному випадку впровадження означає резервування місця на сторінці під включається элемент, що, з метою його подальшої обробки плагином. Останній відповідає за його відтворення.

Проблема полягає в тому, що весь механізм роботи із плагинами не є сумісним із жорстким XHTML. Застосовується створений компаніями Netscape і елемент, що одержав широке поширення, <embed>. Якщо він зустрічається на вашій сторінці, подбайте про те, щоб у її початку був зазначений DTD для перехідного XHTML.

У принципі, <embed> - це щось начебто <img />. Він складається з імен елемента, що включається, його URL і, при бажанні, розмірів резервованого вікна.

Наприклад: <embed name="Moviel" src="moviel.mov" width="240" height="120" pluginspage="http:// www.apple.com/quicktime/download/"> </embed>

Як видно, в <embed> є атрибут name, що дуже корисно при використанні JavaScript, а також атрибути, що задають висоту й ширину. Все це нагадує елемент <img>, щоправда, в <embed> повинен бути ще й закриваючий тег. Але ця подібність обманчиво хоча б тому, що кожний" плагин дозволяє задавати власні атрибути елемента <embed>, які не є стандартними для XHTML, але зате визначають різні режими програвання мультимедиа.

Тим часом є й цілком легальний з погляду XHTML елемент, яким дає можливість якось працювати із плагинами. Це контейнер <object>. Пізніше ви побачите, що він найчастіше використовується для Java-Апплетов, але допомагає реалізовувати й інші можливості впровадження. Наприклад, можна впроваджувати один HTML-Документ в іншій, вийде щось начебто внутрірядкового фрейму (елемент <i f rame> описаний у главі 12). Елемент <object> трохи складніше

описаного вище <embed>, однак деякі приклади його використання ви побачите вже в найближчих параграфах.

Internet Explorer 5.5 і наступні версії не мають підтримки елемента <embed>, зобов'язуючи веб-авторів використовувати <object>. Щоб впровадити мультимедійний елемент, тепер потрібно застосовувати елемент <object>, далі ми обговоримо його докладніше.

Впровадження QuickTime

```
<embed name="NewMovie" src="newmovie.mov" heigrt="320"  width="240">
</embed>
```

Фірма Apple пропонує при цьому використовувати ще кілька додаткових атрибутів. Вони стосуються режимів програвання й настроювань вікна, у якому з'являється впроваджений елемент. Деякі із цих атрибутів разом з рекомендуються значеннями, що, наведені в табл. 1

Запам’ятайте, що всі перераховані додаткові атрибути характерні тільки для впровадження за допомогою <embed> файлів QuickTime! Наприклад:

```
<embed src="moviel.mov" autoplay="true" height="320"  width="240"> </embed>
```

У цьому прикладі відео починає відтворюватися відразу ж після завантаження файлу на комп'ютер користувача. Елемент <embed> іноді використовується небагато по-іншому, роблячи процес перегляду відео двоступеневим. Розглянемо більш складний приклад:

```
<embed src="post_movie.mov" autoplay="true" controller="false"
href="full_movie.mov">!!!</embed>
```

Таблиця 1. Атрибути елемента <embed> при впровадженні QuickTime

(Повний список ***-http://www.apple.com/quicktime/authoring/embed.html****)*

Атрибут (команда)	**Значення, що рекомендується**	**Призначення**
Autoplay	true	Автоматично запускає відео
controller	false	Ховає панель керування
loop	true	Зациклює відео
playeveryframe	true	Змушує відмовитися від пропуску частини кадрів, навіть якщо швидкість уповільнена (обмежена технічними

		можливостями машини)
Volume	0-256	Установлює рівень гучності. 256 — максимальна гучність
Hidden	true	Приховує відеоряд (корисно для звукових файлів у форматі QuickTime)
href	url	Створює вікно, у якому програється відео, як гіперпосилання

Споконвічно відеофайл, що завантажується (post_movie.mov) мається на увазі лише демонстраційний, тобто у вікні невеликого розміру із зображенням низької якості, а може бути, і взагалі складається з одного кадру. Якщо користувач бажає переглянути відео цілком і з нормальною якістю, він може скористатись вікном впровадженого відео як гіперпосиланням і перейти по ньому на інший файл (full_movie.mov).

Оскільки в даному прикладі controller="false", то користувач не має можливості контролювати процес відтворення за допомогою панелі керування - вона схована. Після завантаження кліп почне програватися автоматично.

Apple відзначає, що деякі браузери, серед яких Internet Explorer версії 5.5 і наступних версій, не має підтримки елемента <embed>, зате працює з Active, технологією програмного впровадження елементів, розробленою Microsoft. Для таких браузерів варто застосовувати <object> поряд з <embed>. Фірма Apple рекомендує оформляти елемент <object> у такий спосіб:

```
<0bject classid="clsid:02BF25D 5-8C17-4B 23-BC80-D3488ABDDC6B" width="320"
heiglit="240" codebase="http://www.apple.com/qtacti vex/qtplugin.cab"><param
name="src" value="moviel.mov" /><param name="autoplay" value="true" />
 <param name="controller" value="false" />
 <embed src="movie.mov" width="320" height="240"
autoplay="true" controller="false" pi
uginspage="http://www.apple.com/quicktime/download/">
 </embed> </object>
```

По-перше, елемент <object> не тільки виведе кліп QuickTime у тих браузерах, у яких це можливо, але й автоматично скачає всі необхідні відсутні компоненти Active. Елемент <embed>, розташований всередині <object>, придасться тим браузерам, які ніколи нічого не чули про Active, однак проти QuickTime нічого не мають.

Важливе зауваження про відео у форматі QuickTime може бути як потоковим, так і таким, що завантажується. Тобто відтворення може починатися як у міру скачування частин файлу, так і після закінчення прийому всього файлу. Потокове відео може бути як трансляцією яких-небудь подій, так і звичайним записом.

Якщо ви раптом зважилися на застосування потокового відео, з кодом HTML вам майже нічого не потрібно робити. Але встає питання установки спеціального серверного програмного забезпечення для цих цілей.

Основна особливість реалізації включення потоку у форматі QuickTime у свою сторінку полягає в тому, що потрібно ставити посилання на особливий URL, що використовує відповідний протокол передачі реального часу (rtsp://).

Отже, для того, щоб визначити відеофайл, що зберігається на потоковому сервері QuickTime завантажува_ як, необхідно спочатку зберегти його як hinted, а потім використовувати всі ті ж елементи <embed> або <object>. Як видно з наведеного приклада, великої різниці немає, не вважаючи протоколу.

<embed src="Посилання на файл" width="240" height="180" pi ugi
nspage="http://www.apple.com/quicktime/download"></embed>
Більш докладно про можливості формату QuickTime можна довідатися за адресою “http://text.marsu.ru/osp/pcworld/2000/08/144.htm”

Формат Windows Media

Як і QuickTime, формат Windows Media може впроваджуватися як за допомогою технології Active, так і за допомогою плагинов фірми Netscape. Питання включення потокового відео й Windows Media зводяться до того самого.

Якщо ви все-таки хочете застосувати метод впровадження необхідно скористатись елементом <embed>:

<embed src="mymovie.avi" width="240" height="180" autoplay="-l"> </embed>

Для Windows Media існують наступні параметри елемента <embed>

Таблиця 2. *Параметри <embed> для відео у форматі Windows Media*

Параметр	Значення
showcontrols	Нуль — панель керування схована; будь-які інші значення — панель керування видна
Autosize	Нуль — відсутність автопідбору розміру відповідно до кадру
showstatusbar	Нуль — рядок стану схований
autostart	Нуль — не запускати відео автоматично після завантаження

З елементом <embed> можна використовувати атрибут pluginspage, що задає розміщення плагина для Windows Media, і src, що задає властиво розміщення файлу з відеозаписом. Можна до складу <embed> включити й параметр type, що дозволяє задавати очікуваний тип даних.

При застосуванні технології Active, варто згадати про <object>.

<object id="Player" type="application/ x-oleobject" classid="CLSID:6BF52A 52-394A-lld3-B 153-OOC04F79FAA6" standby=" Media Player..." width="320" height="240" codebase=
"http://activex.microsoft.com/activex/controls/mplayer/en/nsmp2inf.cab#Version=6.4.7.1112">
<param name-"autoStart" value="true" />
<param name="URL" value="http://www.fakecorp.com/movies/rnymovie4avi" />
<embed src="mymovie.avi" width="320" height="240" showstatusbar="-l" showcontrols="-l" showdisplay="l" pluginspage="http://www.microsoft.com/netshow/download/player.htm"> </embed> </object>

Усередині визначення <object> можна бачити атрибут codebase, що використовується для автоматичного завантаження й установки необхідних компонентів Active. Як і при роботі з файлами формату QuickTime, усередині <object> може розташовуватися елемент <embed>.

Формат RealMedia

Наступний формат, що ми розглянемо, зветься RealMedia. Він не дозволяє поширювати відео- або аудіодані, тобто зберігати їх на диску. Даний формат є винятково потоковим.

Потокове відео - тут

Такий підхід не є самим вдалим у змісті функціональності й гнучкості, оскільки в цьому випадку потрібно, щоб веб-сервер розпізнавав файли у форматі RealMedia, у той же час для відтворення відео запускається зовнішня програма - плейер RealMedia.

Більш правильним рішенням є використання сервера RealMedia. При його наявності ви одержуєте можливість як зв'язувати, так і впроваджувати мультимедіа, так ще й декількома способами.

<embed src="http://realserver.fakecorp.com:8080/ ramgen/realmovie.rm?embed" width="320" height="240" type="audio/ pn-realaudio-plugin" controls="ControlPanel" console="one" autostart="true"></embed>

Підключення анімації у форматі Flash

Флеш-презентації дотепер часто називають відеопрезентаціями, хоча вони, у принципі, відрізняються й від QuickTime, і від Windows Media. Замість лінійного потоку відео - або аудіоданих, у технології флеш використовується інтерактивна анімація.

Приклад впровадження флеш-анімацій:

```
<object classid="clsid:D27CDB6 E-AE6D-llcf-96B 8-444553540000"
 width="100" height="100" codebase="http://active.macromedia.com/flash5/
cabs/swflash.cab#version=5.0.0,0"> <param name="movie" value="flashmovie.swf" />
 <param name="play" value="true" /> <param name="loop" value="true" />
<param name="quality" value="high"/><embed src="flashmovie.swf" width="100"
height="100"
 play="true" loop="false" quality="high"
pluginspage="http://www.macromedia.com/Shockwave/
 download/index.cgi?pl_prod_version=shockwaveflash"> </embed>
 </object>
```

Якщо ви уважно читали параграфи, що стосуються QuickTime, ви повинні були помітити, що робота із флеш не дуже сильно відрізняється від роботи з іншими форматами.

Різниця при роботі із флеш форматом відрізняється від інших форматів лише тим, що в цьому випадку є довгий список можливих атрибутів. Вони служать для налаштування специфічних параметрів.

Завдання:

На основі HTML сторінок лабораторних робіт №1, №2, №3, №4 і відштовхуючись від теоретичних основ, створити HTML сторінку з підтримкою інформаційних потоків, а саме:

1) Забезпечити підтримку формату “AVI”;
2) Забезпечити підтримку формату “MP3”;
3) Забезпечити підтримку формату “QuickTime Movie”;
4) Забезпечити підтримку формату “QuikTime Panorama”;

5) Забезпечити підтримку формату “Flash”.

Після кожного підключеного мультимедійного файлу, описати в короткому вигляді його переваги й недоліки. Контекст мультимедійних файлів вибрати відповідно тематики сторінок попередніх лабораторних робіт.

Примітка

- ✓ *При необхідності інсталяція QuikTime перебуває на сервері в папці Install.*
- ✓ *Приклади робіт у форматі QuikTime - http://www.apple.ru/software/examples/index.htm.*
- ✓ *Приклади файлів інших форматів можна одержати через будь-яку пошукову систему.*

Контрольні питання:

1) Які мультимедіа формати Ви знаєте?
2) Яка анімація використовується у технології флеш?
3) Як здійснюється впровадження QuickTime?
4) Що представляє собою формат “AVI”?
5) Як забезпечити підтримку формату “MP3”?

Лабораторна робота №6 Робота з формами

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

Мета: Навчитися передавати дані серверу з допомогою форм.

Знати: Методи передачі даних. Елементи керування форм.

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

Теоретичні відомості:

Способи передачі параметрів сценарію

Як Ви вже зрозуміли, найпоширенішими методами передачі даних між браузером і сценарієм є GET і POST. Однак вручну задавати рядок параметрів для сценаріїв і до того ж URL-Кодувати їх, досить утомливо. Давайте подивимося, що пропонує нам для полегшення життя HTML.

Ми будемо розглядати метод GET для передачі запитів серверу.

Навіть програмістові утомливо набирати параметри в URL вручну. Різні ?, &, %... На щастя, існують зручні можливості мови HTML, які, звичайно, підтримуються браузерами.

Отже, нехай у нас 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>
```

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

Передача параметрів через адресний рядок браузера

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

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

Зверніть увагу, що ми розділяємо параметри символом **&**, а також використовуємо знак рівності **=**. Зараз з'ясуємо, що це означає.

Використання HTML-Форм

Як тепер нам зробити, щоб користувач міг у зручній формі ввести своє ім'я й вік? Очевидно, нам необхідно інтерактивне вікно, у яке здійснюється введення наших параметрів, при чому через браузер.

Отже, нам 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 усередині тегів <*input*> у формі й розділені символом & у рядку параметрів. Помітьте, що в атрибуті action тега <*form*> ми задали відносний шлях, тобто сценарій hello.php буде шукатися браузером у тому ж самому каталозі, що й файл form.html.

Як ми знаємо, всі перекодування й перетворення, які потрібні для URL-кодування даних, здійснюються браузером автоматично.

Використання форм дозволяє в принципі 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.

Залишилося тепер тільки визначитися, як ми можемо 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.

Мабуть, немає іншої такої мови, як PHP, яка б настільки полегшила вам завдання обробки й розбору зовнішніх змінних, тобто змінних, які надійшли з HTML-форм (із браузера користувача). Справа в тому, що в мову PHP вбудовані всі необхідні можливості, так що вам не прийдеться навіть і замислюватися над особливостями протоколу HTTP і міркувати, як же відбувається відправлення й прийом POST-форм або навіть завантаження файлів. Розробники PHP все передбачили.

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

```
"Привіт, <ім'я>! Вам <вік> років!"
```

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

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

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

А от і HTML-документ send.html, за допомогою якого ми параметри name і age передамо 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. От кілька прикладів:

```
<?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 їхній атрибут NAME.

Кожний елемент форми може мати початкове й кінцеве значення, які є символьними рядками. Початкові значення елементів не міняються, завдяки чому

може здійснюватися скидання значень, зазначених користувачем. Результатом цієї дії буде установка всіх керуючих елементів форми у свої первісні, що використовуються за замовчуванням значення.

В HTML 4.01 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. Це значить, що якщо один з них ставиться в положення "вкл", то всі інші автоматично - у положення "викл". Саме це і є перевагою їхнього 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.

Як бачите, 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. Початковий і кінцевий теги FORM задають границі форми, тому їхня вказівка є обов'язковою.

Приведемо атрибути тегу FORM:

- action- єдиний обов'язковий атрибут. У якості його значення вказується URL-Адреса запитуваного скрипта, що буде обробляти дані, що втримуються 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 дана 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 усередині форми поля введення рядка тексту, ім'я файлу, пароля й.т.ін.

Зверніть увагу на особливість INPUT - у нього немає кінцевого (завершального) тегу. Атрибути й особливості використання INPUT залежать від способу його використання. Розглянемо ці способи.

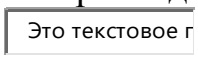
» Однорядкові поля введення

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.

От приклад однорядкового поля введення:



» Поля введення пароля

Звичайно, ім'я користувача можна ввести за допомогою звичайного текстового поля. А от пароль не повинен відображатися на екрані при його введенні. У цьому нам допоможе поле введення пароля:

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

Принцип роботи даного тегу точно такий же, як і текстового. 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 таких полів, наприклад, можуть передаватися параметри налаштування.

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

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

» Незалежні перемикачі

Дуже часто користувачеві, що заповнює форму в браузері, необхідно дати можливість указати свої налаштування за допомогою вибору певних значень. При цьому приводяться самі ці значення, 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

В HTML є й такий перемикач, що залежить від інших перемикачів, він розглядається далі.

» 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=Відправити>
</form>
```

Дана форма буде виглядати так:

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

Як тільки користувач натисне кнопку "Відправити", 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 такій:

```
<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, установлює для всіх елементів форми значення за замовчуванням. Бажано, щоб на формі була така кнопка, особливо, якщо це більша 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. Клич на цьому малюнку дає те ж саме, що й натискання на кнопку submit. Однак, крім цього, сценарію будуть передані координати місця кличу на малюнку. Координати будуть передані у форматі ім'я.x=коор_X, y=коор_Y. Синтаксис кнопки відправлення з малюнком:

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

Багатострокові текстові поля. Тег TEXTAREA

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

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

Кілька значень щодо використання атрибутів: 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 *клавіші* 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. От приклад богаторядкового текстового поля з використанням атрибуту wrap=Virtual:

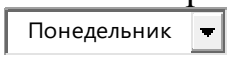
Списки вибору. Тег SELECT

» Списки з єдиним вибором

Досить часто існує необхідність представити які-небудь 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>Середовище</option>
    <option value=4>Четвер</option>
    <option value=5>П'ятниця</option>
    <option value=6>Субота</option>
    <option value=7>Неділя</option>
</select>
```

А от його практична реалізація:



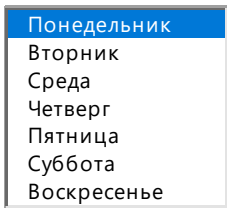
Атрибут name визначає ім'я параметра, що буде переданий скрипту. Якщо атрибут size дорівнює 1, то список буде "оснащений" смугою прокручування. Значення, обране в списку за замовчуванням, можна вказати за допомогою атрибута

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]>
```

Завдання:

1. Скласти програму, що приймає анкетні дані користувача. Намалювати HTML форму.
2. Всі поля обов'язкові для заповнення. Реалізувати перевірку на відсутність незаповнених полів за допомогою JavaScript.
3. 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 й для чого вони використовуються?
4. Як отримати достп до даних введених з форми на стороні сервера?

Лабораторна робота №7. Робота з рядками в PHP

Тема: Застосування рядкових функцій 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.

В PHP 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'
 ?>
```

А тепер докладно розберемо синтаксис типу даних **string**.

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:

```
<?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:

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 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 ви можете використовувати 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 містите 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. Приведемо приклади:

```
<?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.

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 багатострочными 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 вираженнями 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 адресатові..."
;
```

```
// 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 по його індексу:

```
$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. От 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 нам доводиться переводити якісь рядки, скажемо, у верхній регістр, тобто робити всі прописні букви в рядку заголовними. У принципі, для цієї мети можна було б скористатися функцією strtr(), розглянутої вище, але вона все-таки буде працювати не так швидко, як нам 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, "АБВГДЕЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ", "абвгдеежзийклмнопрстуфхцчшщъыьэюя");

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 локали (локальних настроювань)

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

Настроювання локали сильно залежать від операційної системи.

Для установки локали 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

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 Ci. 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, вибачите...

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 якихось 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 ($pass_b) - верифицируємий пароль
    echo "<b> Хеш-
   Код верифицируємого пароля '$pass_b':</b><b style=\"color:green\">".md5($pass
   _b)."</b><br>";
    // Порівнюємо хеш-коди MD5 вихідного й верифікованого пароля
    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. Скласти HTML форму, яка буде передавати текст PHP сценарію, обробляти його й виводити на екран.
2. Уведений 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.

Контрольні питання:

1. Які типи визначення рядків є?
2. Як вписати в рядок значення змінної?
3. Типи функцій для роботи з рядками
4. Що таке регулярний вираз?
5. Яким чином, можна видалити прогалини з тексту?
6. Яким чином можна виконати в PHP конкатенацію рядків?
7. Які є оператори порівняння рядків в PHP?

Список скорочень і спеціальних термінів

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)

SEO розшифровується як Search Engine Optimization - це оптимізація сайту для подальшого просування сайту в рейтингу пошукових систем умовно можна розділити на три частини:

- В роботі всередині сайту. У неї входять виправлення можливих помилок, додавання і зміна контенту, HTML-коду сторінок сайту, перелінковка і так далі. Так звана внутрішня оптимізація.

- Це розкрутка сайту самостійно. На цьому кроці необхідно вивести ресурс на перші позиції за допомогою груп заходів, що виконуються поза сайтом (на інших сайтах, в каталогах статей, форумах, закладках і інших майданчиках), завдання яких наростити необхідну кількість посилань і просунути сайт по цільових запитах, а так само збільшити його авторитетність. Це називається просуванням сайту або зовнішньою оптимізацією.

- В підтримці досягнутих позицій і поліпшення отриманих результатів. Спостереження за своїми результатами і показниками конкурентів, зміна ключових слів, текстів для посилань, змісту сайту, коректування майданчиків.

CMS - це англійська абревіатура, яка розшифровується як Content Management System. Дослівний переклад - Система керування контентом. CMS часто називають словами «движок», «адмінка» або просто: «на чому побудований сайт».

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

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.

Якщо опустити технічні деталі, то при виборі CMS потрібно врахувати три головні чинники:

-Популярні. 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.

Для створення сайтів використовують: HTML, CSS, JavaScript (VBScript), FLASH, PHP (Perl), рел. БД, наприклад, MySQL. Клієнтське середовище (браузер) - передній край роботи програми- тут відображаються HTML сторінки у вікні і обслуговуються історії сеансів HTML-сторінок, що відображаються в браузері протягом сесії. Об'єкти середовища за допомогою клієнтської мови - JavaScript- повинні мати можливість маніпулювати сторінками, вікнами, історією. При роботі з сервером необхідні мови типу PHP (Perl), рел. БД, наприклад, MySQL.

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. За допомогою CSS-можна розширити можливості по оформленню Web-сторінок.

CSS - (Cascading Style Sheets) - Каскадні 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.

HTML - код формує текст логічно, тобто структури Web- сторінки- розташування, порядок проходження абзаців, графічних зображень рядків і осередків у таблиці.

Таблиці стилів CSS формують тексти фізично, задають уявлення Web - сторінки, яким шрифтом буде набрано звичайний текст абзаців, яким кольором виділити заголовки, рамки та ін.

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

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.

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.

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.

Рекомендована література

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. Колисниченко Д.Н. PHP 5/6 и MySQL6. Разработка Web- приложений-СПб.:БХВ-Петербург, 2009.-624с.
5. Дейв Крейн, Эрик Паскарелло, Даррен Джеймс. Ajax в действии. Издательство: Диалектика, 2006 г., 640 стр.
6. Ричард Вагнер, Аллен Вайк. JavaScript. Энциклопедия пользователя. Издательство: ДиаСофт, 2001 г., 464 стр.
7. Квинт И. HTML, XHTML и CSS на 100%. Издательство:Питер, 2010, 384с.
8. Дунаев В. В. Web-программирование для всех. Издательство: БХВ_Петербург, 2008, 560с.
9. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. Питер, 2015, 688с.

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

Міністерство освіти і науки України
Кіровоградський національний технічний університет
Механіко-технологічний факультет
Кафедра програмування та захисту інформації

Лабораторна робота №_
з дисципліни «Web-програмування»
на тему:
«Тема лабораторної роботи»

Виконав: студент гр. КІ-__-__
Акуліч Д.
Перевірив:викладач
(Вказати ПІБ викладача)

Кропивницький 201_

Тема: Основні засоби html.

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 використати:

- теги форматування тексту (розмір, стиль, колір, фон),
- теги для групування елементів,
- створення списків,
- створення таблиць,
- посилань,
- додавання малюнків,
- елементи розмітки веб-сторінок <header>, <nav>, <section>, і <footer>.

Не використовувати CSS.

Варіант 1

Сайт – тлумачний словник з ілюстраціями

Хід роботи:

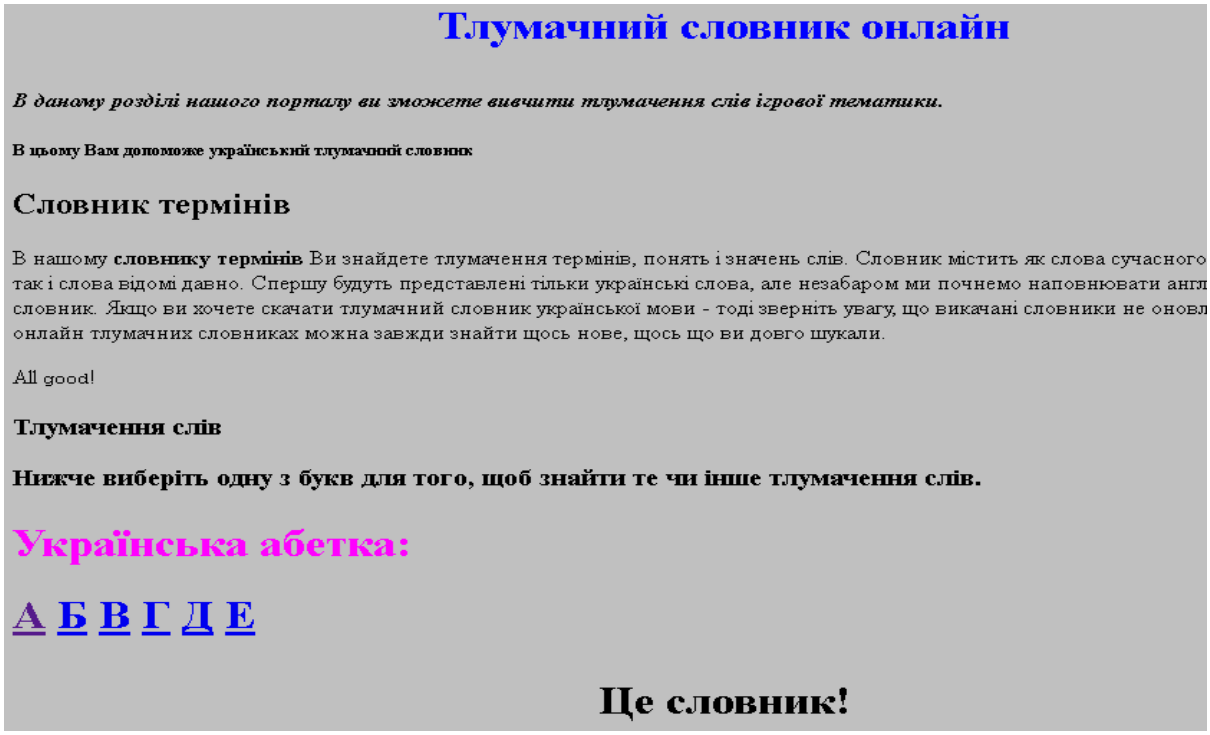
1 Лістинг лр1.html

```
<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.

```
</p>
<div>All <tt>good</tt>!</div>
<h3>Тлумачення слів<h3>
Нижче виберіть одну з букв для того, щоб знайти те чи інше тлумачення слів.
<h1><font color="#FF00FF">Українська абетка:</font><h1>
<a href="А_лр1.html">А</a>
<a href="Б_лр1.html">Б</a>
<a href="В_лр1.html">В</a>
<a href="Г_лр1.html">Г</a>
<a href="Д_лр1.html">Д</a>
<a href="Е_лр1.html">Е</a>
……..
<p><center><img src="img/mini_7.jpg" alt="Це словник!"></center></p>
</body>
</html>
```



Мал. 1 - Головна сторінка сайту

2 Лістинг А_лр.html

```
<html>
<head>
<title>А_lr1 Прізвище студента </title>

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

<h1><p align=center><font color="blue">Тлумачний словник онлайн
</font></p></h1>
```

Слова на літеру А

В списку нижче <small>представлені слова, які починаються на літеру</small> <big>А.</big>

Виберіть одне із слів:

Азарт

<img src="img/tigr.jpg" alt="Ось який азарт!">

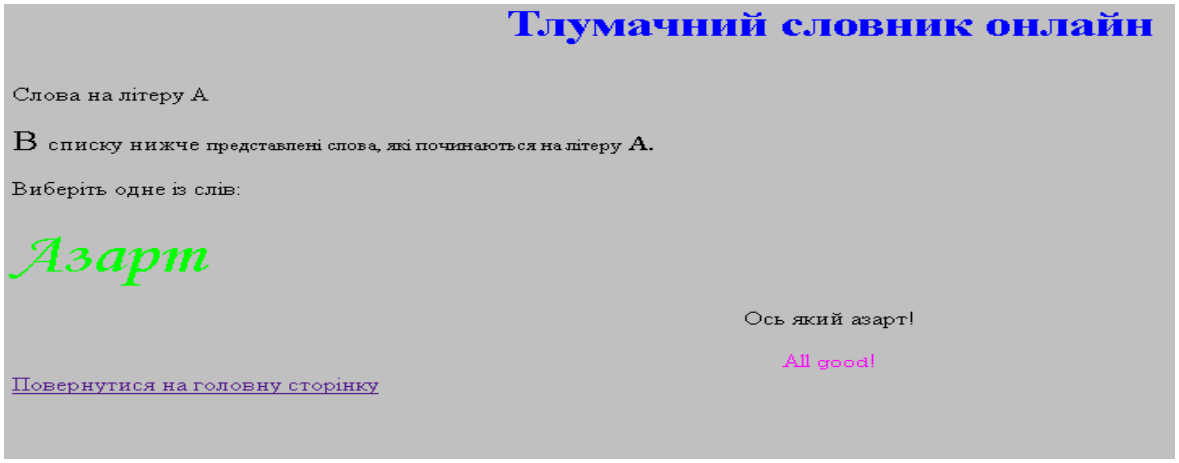
All <tt>good</tt>!

Повернутися на головну сторінку

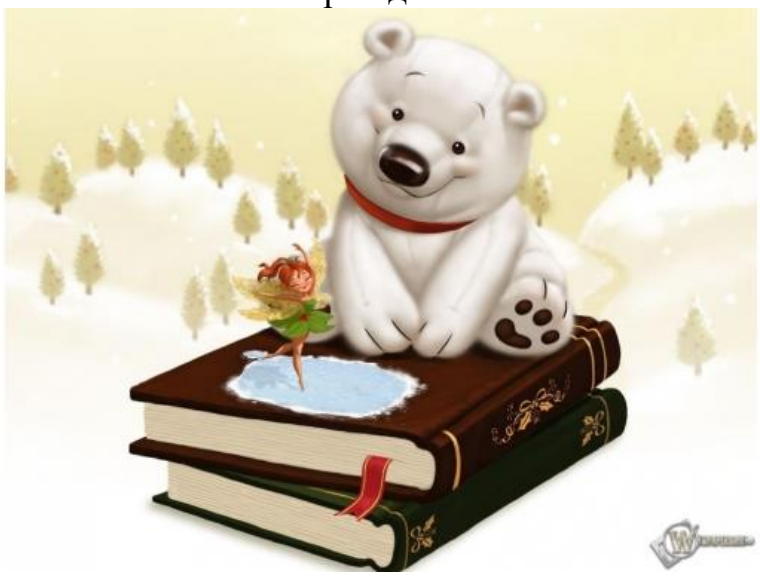
….

</body>

</html>



Мал.2 – Перехід за посилкою



Мал. 3 – Файл mini_7.jpg

3. **Інші файли організовані аналогічним чином**

Таблиці:…..

Списки:……

….

Контрольні питання:

25. Що таке HTML?
26. Яким чином забезпечується форматування тексту в HTML-документі?
27. Яка мета розробки HTML5?
28. Який синтаксис HTML?
29. Структура HTML-документа?
30. Яким чином здійснюється взаємодія між HTML-документом і сервером?
31. Які текстові редактори для роботи з HTML Ви знаєте?
32. Що означає DOCTYPE у HTML-документі?
33. Що таке заголовок HTML-документа?
34. Що таке тіло HTML-документа?
35. Для чого служить текст між тегами <title>?
36. Яким тегом забезпечуються посилання на інші документи HTML і його атрибути?
37. Яким тегом визначаються параграфи?
38. Яким тегом визначаються заголовки?
39. Яким тегом визначається найбільший заголовок?
40. Яким тегом визначається найменший заголовок?
41. Що визначається тегом <img>?
42. Перелічити HTML - елементи форматування тексту.
43. Які теги списків ви знаєте?
44. Що визначає тег <table>?
45. Яким тегом визначається заголовок таблиці?
46. Для чого використовують тег ?
47. Для чого використовують тег ?
48. Що визначають <header>, <nav>, <section>, і <footer> елементи?

Відповіді:

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