МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ

В. В. Кузьменко, Г. Г. Швачич, Н. С. Романова В. В. Байрак

КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ В ДОКУМЕНТОЗНАВСТВІ

РОЗДІЛ "КОМП'ЮТЕРНІ МЕРЕЖІ"

ДОВІДКОВЕ КЕРІВНИЦТВО ПО СТВОРЕННЮ WEB-СТОРІНОК НА ОСНОВІ МОВИ ПРОГРАМУВАННЯ JAVASCRIPT З ВПРАВАМИ

Затверджено на засіданні Вченої ради академії
як навчальний посібник

Дніпропетровськ НМетАУ 2006

УДК 004(075.8)
Кузьменко В. В., Швачич Г. Г. Романова Н. С. Байрак В. В. Комп’ютерні технології в документознавстві. Розділ “Комп’ютерні мережі”: Навчальний посібник. – Дніпропетровськ: НМетАУ, 2006. – 44с.

Викладені основні поняття мови програмування JavaScript, які надають можливість створення динамічних та інтерактивних Web-сторінок і Web-документів.

Призначений для студентів спеціальності 6.020100 – документознавство та інформаційна діяльність.

Іл. 15. Табл.1 Бібліогр.: 5 найм.

Відповідальний за випуск Г. Г. Швачич, канд. техн. наук, проф.

Рецензенти: Д. Г. Зеленцов, канд. техн. наук, доц. (Український Державний хіміко-технологічний університет)
Т. М. Пашова канд. техн. наук, доц. (Дніпропетровський Державний аграрний університет)

ВСТУП ЗАПУСК JAVASCRIPT

JavaScript – це мова програмування для створення сценаріїв (скриптів) інтерактивних Web-сторінок, яка розроблена фірмою Netscape. Щоб запускати сценарій, написаний мовою JavaScript, необхідний браузер, здатний працювати з JavaScript. Такими браузерами є Netscape Navigator (починаючи з версії 2.0), а також Microsoft Internet Explorer (MSIE – починаючи з версії 3.0). З тих часів, як ці браузери були широко розповсюджені, стало можливим працювати зі сценаріями (скриптами) створення інтерактивних Web-сторінок, які написані мовою програмування JavaScript.

Всі сценарії спочатку повинні бути набрані в програмі «Блокнот», збережені як файли з розширенням **.htm** на логічній частині жорсткого диска у своїй папці, потім відкриті при використанні Microsoft Internet Explorer.

ТЕМА 1 РОЗМІЩЕННЯ JAVASCRIPT НА HTML-СТОРІНЦІ

Керуючі теги (tags), або прапори, які використані для створення наступного сценарію Web-сторінки:

1. **<Html>** – показує, що документ створено на HTML-сторінці;
2. **<Body>** – головна частина (тіло) сторінки. Усе, що розташоване між тегами <body> та </body>, складає тіло документа;
3. **
** – перехід на наступну строку.

Програмний код мови JavaScript розміщається безпосередньо на HTML-сторінці.

Розглянемо приклад:

```
<html>
<body>
<br>
Це звичайний HTML документ.
<br>
<script language="JavaScript">
document.write ("А це JavaScript!")
</script>
```


Знову документ HTML.

</body>

</html>

Відмінністю від мови гіпертекстової розмітки HTML є код JavaScript: **<script language="JavaScript">**

… … … … … … … … … … … …

</script>

Щоб побачити, як працює наведений сценарій (скрипт) (рис.1) створення інтерактивних Web-сторінок, необхідно:

1) записати даний приклад у програмі «Блокнот»;
2) зберегти файл з розширенням **.htm** на логічній частині жорсткого диска у своїй папці;
3) відкрити файл, використовуючи Microsoft Internet Explorer.

Вікно відкриття HTML файлу наведено на рис.1.

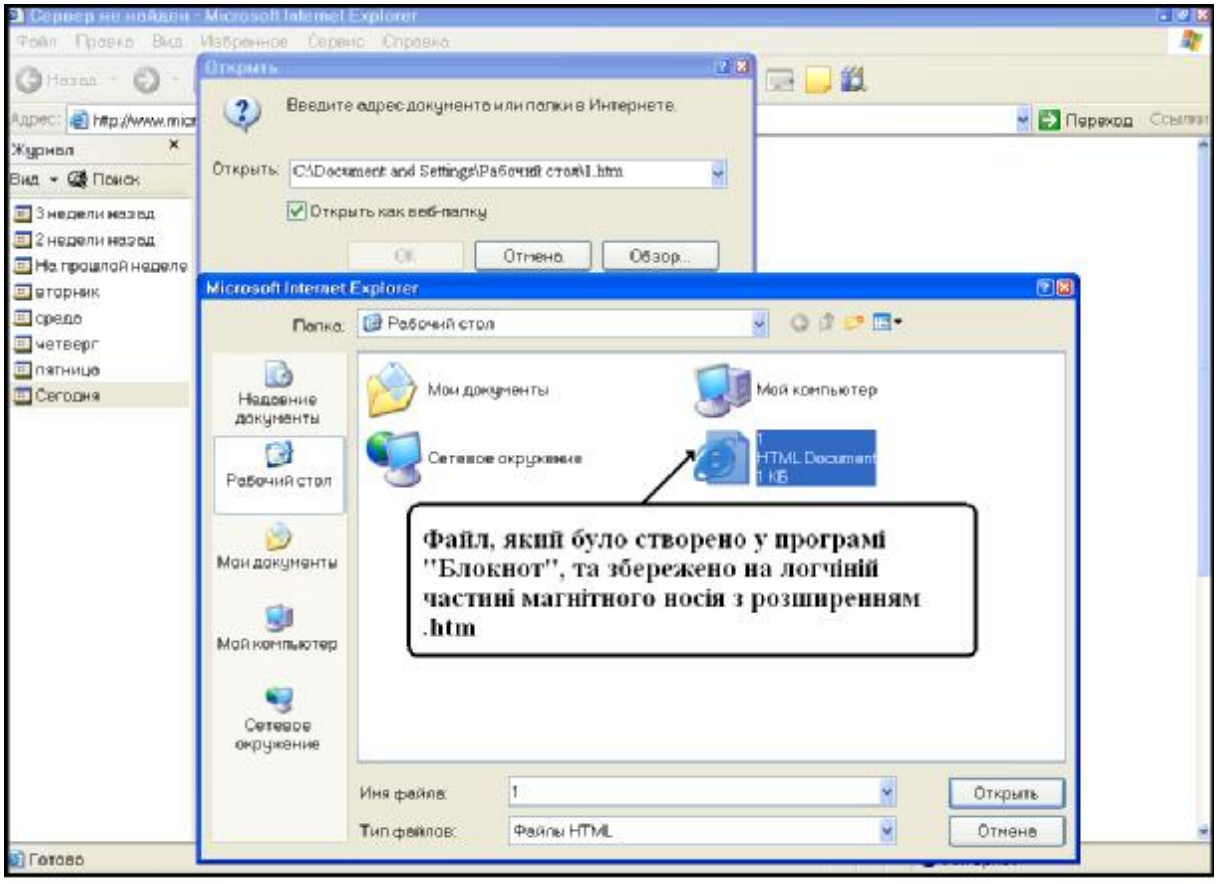

Рис.1

Результатом виконання цього файлу будуть три рядки:

Це звичайний HTML документ.
А це JavaScript!
Знову документ HTML.

Висновок

У прикладі продемонстрований тег (прапор) мови програмування JavaScript ознаки **<script>**. Усе, що знаходиться між тегами **<script>** й **</script>**, інтерпретується як код мови JavaScript. Інструкція **document.write()** – одна з найбільш важливих команд, що використовуються при програмуванні мовою JavaScript. Команда **document.write()** використовується, якщо необхідно зробити запис в поточному документі. В даному випадку програма на JavaScript в HTML-документі пише фразу **"А це JavaScript!".**

ТЕМА 2 ПОДІЇ JAVASCRIPT

Події й оброблювачі подій є важливою частиною програмування мовою JavaScript. Події ініціюються тими або іншими операціями користувача. Наприклад, якщо натиснути на кнопку «миші», відбувається подія **"Click",** якщо ж вказівник «миші» перетинає яку-небудь виноску гіпертексту, відбувається подія **"MouseOver"**.

Існує кілька різних типів подій, які можуть бути виконані за допомогою спеціальних програм їх обробки. Так, у результаті натискання на кнопку «миші» може створюватися вікно, що випадає. Це означає, що створення вікна повинно бути реакцією на подію **"Click".**

Програма – оброблювач подій, котра використана в даному випадку, називається **onClick.**

Нові керуючі теги, які використані для створення наступного сценарію Web-сторінки:

1) **<Form>** – тег, який надає можливість створити форму вводу даних;
2) **<Input>** – тег, який створює поле вводу даних;
3) **<Type>** – тег об’яви підрозділу локальних типів даних.

Наведений нижче програмний код JavaScript становить простий приклад програми обробки події **onClick:**

<form>

<input type = "button" value = "Click me" onClick = "alert ('Привіт!')">

</form>

У результаті виконання програмного коду створена форма із кнопкою

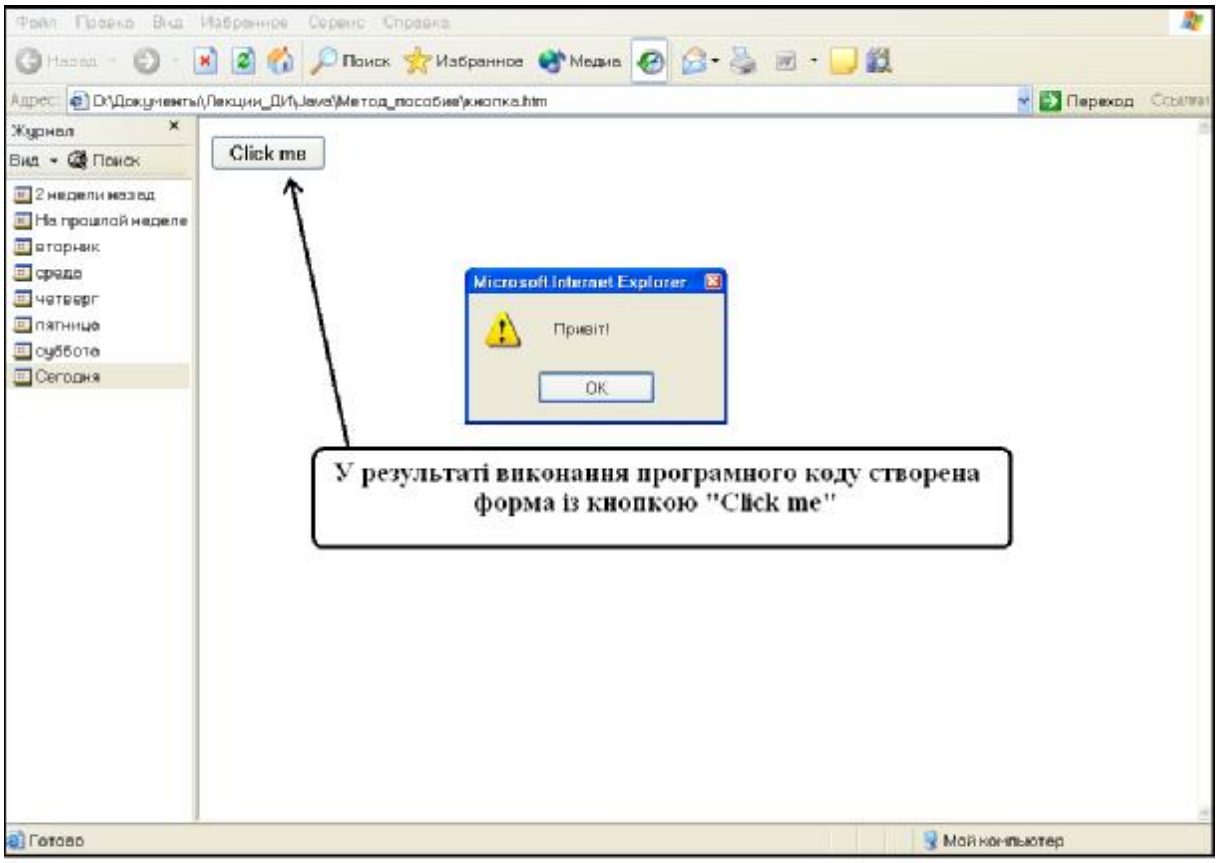

Рис.2

Висновок

Наведений приклад має два нових атрибути:

1. Атрибут **onClick = "alert ('Привіт!')"** у тэгу <input> обумовлює подію, що відбувається, при натисканні на кнопку «миші» ("button").

Оператор **value** визначає зміст події (значення даних, які передаються). Якщо має місце подія **Click,** повинен виконатися виклик **alert('Привіт!'**). У цьому випадку не використовується тэг <script>.

2. У команді **document.write()** використані подвійні лапки, а в конструкції **alert()** – одинарні.

У більшості випадків можливо використати обидва типи лапок. Але якщо написати **onClick = "alert ('Привіт!')"**, то код скрипта не буде виконаний, оскільки стає неясно, до якої із частин конструкції має відношення функція обробки подій **onClick**, а до якій не має. У наведеному програмному коді необхідно перемежовувати обидва типи лапок. При цьому не має значення, у якому порядку вони використані – спершу подвійні, а потім одинарні, або навпаки. Можна точно так само написати й **onClick = 'alert ("Привіт!")'**.

ТЕМА 3 ФУНКЦІЇ JAVASCRIPT

У більшості програм мовою JavaScript користуємося функціями. **Функції у програмуванні мовою JavaScript визначають спосіб спільного зв'язку декількох команд**. Створимо скрипт, що друкує один і той же текст три рази підряд.

Розглянемо приклад:

```
<html>
<script language="JavaScript">
document.write("Ласкаво просимо на мою сторінку!<br>");
document.write("Це JavaScript!<br>");

document.write("Ласкаво просимо на мою сторінку!<br>");
document.write("Це JavaScript!<br>");

document.write("Ласкаво просимо на мою сторінку!<br>");
document.write("Це JavaScript!<br>");
</script>
</html>
```

У результаті виконання коду скрипта отримаємо текст

Ласкаво просимо на мою сторінку!

Це JavaScript!

який надрукований три рази.

Якщо подивимося на вихідний код скрипта, то побачимо, що для одержання необхідного результату певна частина його коду була повторена три рази підряд. Це не ефективно. Можемо вирішити те ж завдання іншим способом, користуючись тегом **myFunction()**.

Розглянемо приклад:

```
<html>
<script language="JavaScript">
 function myFunction(){
      document.write("Ласкаво просимо на мою сторінку!<br>");
      document.write("Це JavaScript!<br>");
}

myFunction();
myFunction();
myFunction();
</script>
</html>
```

У скрипті визначена функція, що складається з наступних рядків:

```
function myFunction(){
document.write("Ласкаво просимо на мою сторінку!<br>");
document.write("Це JavaScript!<br>");
}
```

Всі команди скрипта, які перебувають усередині фігурних дужок – {} – належать функції **myFunction**.

У наведеному прикладі є три виклики функції **myFunction().** Це означає, що зміст функції (команди, яка вказана у фігурних дужках) було виконано тричі.

Можлива передача змінних при виклику функції надає скриптам гнучкість у використанні.

ТЕМА 4 СУМІСНЕ ВИКОРИСТАННЯ ФУНКЦІЙ ТА ПРОЦЕДУР ОБРОБКИ ПОДІЙ

Нові керуючі теги, які використані для створення наступного сценарію Web-сторінки:

1. **<Head>** – інформація про сторінку;
2. **<Var>** – підрозділ об’яви локальних змінних;
3. **<Alert>** – підрозділ об’яви результату виконання події.

Функції можуть використатися разом із процедурами обробки подій.

Розглянемо приклад:

```
<html>
<head>
<script language="JavaScript">
function calculation(){
 var x=12;
 var y=5;
var result="Результат підсумовування 12+5"
alert(result);
var result=x+y;
alert(result);
}
</script>
</head>
<body>
<form>
<input type="button" value="Calculate" onClick="calculation()">
</form>
</body>
</html>
```

При натисканні на кнопку здійснюється подія, виклик функції **calculation().** В нашому випадку функція задана рядком **function calculation()** у першій частині програмного коду. Процедура її виклику рядком **<input type="button" value="Calculate" onClick="calculation()">** у другій частині програмного коду.

Функція виконує обчислення, користуючись змінними **x, y** та командою **result** (рис.3)**.** Змінні визначені за допомогою ключового слова **var**. Команда **alert(result)** виконує те ж саме, що й **alert(17)**. У результаті виконання програмного коду скрипта одержали вікно, що випадає, у якому, при натисканні на кнопку **Calculate** спочатку з'явиться кнопка з надписом **Результат підсумовування 12+5.** При натисканні на останню кнопку, з'явиться кнопка з надписом **17**.

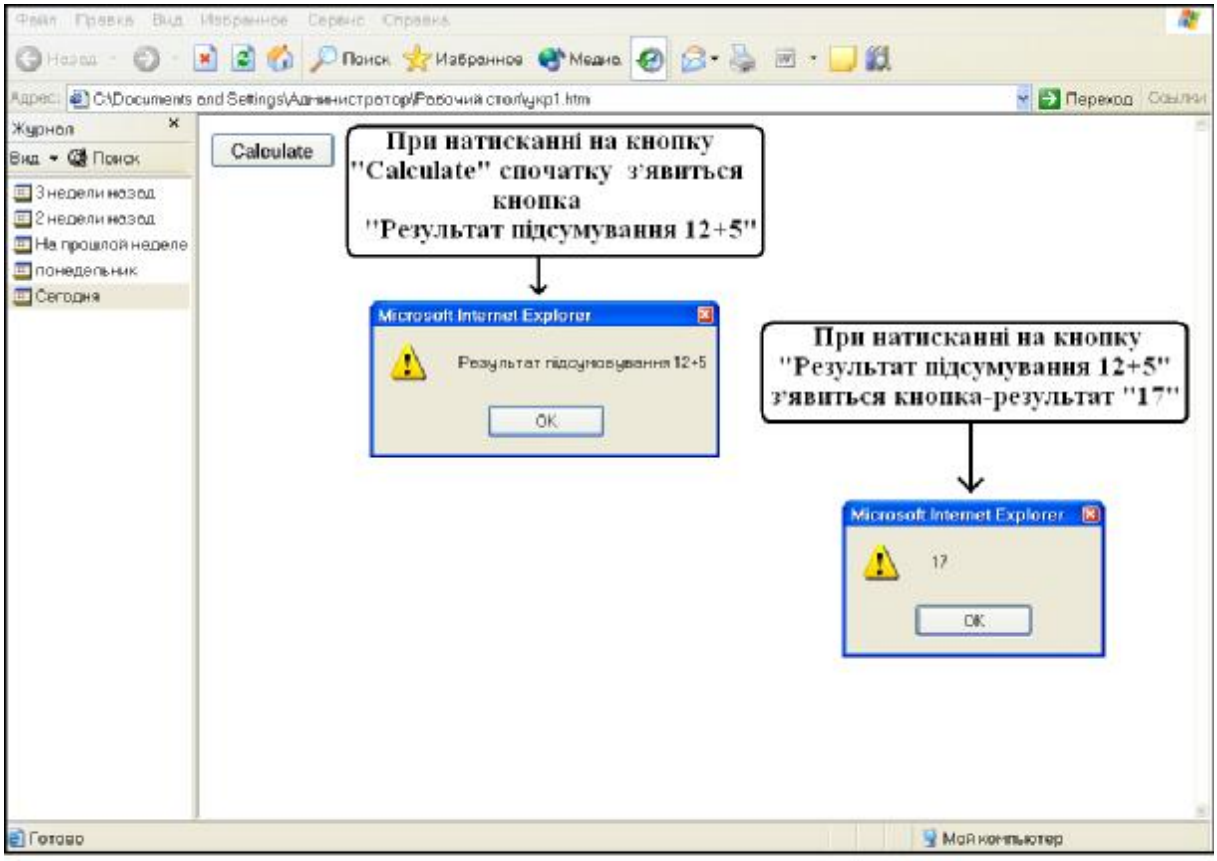

Рис.3

Висновок

Функція завжди приймає який-небудь аргумент. Змінні можуть бути використані для зберігання різних типів величин – чисел, рядків тексту й т. д. Рядок скрипта **var result= x + y;** повідомляє браузеру про те, що необхідно створити змінну **result** і помістити туди результат виконання арифметичної операції **x + y**.

ТЕМА 5 ІЄРАРХІЯ ОБ'ЄКТІВ В МОВІ ПРОГРАМУВАННЯ JAVASCRIPT

Нові керуючі теги, які використані для створення наступного сценарію Web-сторінки:

1. **<Bgcolor>** – номер або назва кольорів HTML-сторінки.
2. **<H1>** – розмір шрифту.
3. **<Align>** – вирівнювання об'єктів на HTML-сторінці.

Команда Align=center. вказує, що об'єкти повинні бути центровані на HTML-сторінці.

4. **<Title>** – заголовок елемента.
5. **<Checkbox>** – веріфікуєме поле вводу тексту.
6. <!- - початок коментаря. Зміст коментаря ігнорується браузером.
7. - - > закінчення коментаря.

У мові програмування JavaScript всі елементи web-сторінки вибудувані в ієрархічній структурі. Кожен елемент являє собою окремий об'єкт. Об'єкти мають певні властивості, а також методи роботи з ними. Мова програмування JavaScript має систему керування об'єктами web-сторінки. Система керування стає можливою лише при розумінні ієрархічної структури об'єктів, які розміщені на HTML-сторінці.

Приклад HTML-сторінки:

```
<html>
<br>
<body bgcolor=#7fffff>
<h1 Align=center>
</h>
Моя домашня сторінка
</br>
<p>
<form name="myForm">
  Name:
<input type="text" name="name" value=""><br>
e-Mail:
```

```
<input type="text" name="email" value=""><br><br>
<p>
<input type="button" value="Clickme" onClick="alert('Закінчення вводу тексту')">
</form>
</body>
</html>
<!--ff0000 - red 00ff00 - green 0000ff - blue ffff00 Yellow 00ffff - cyan ff00ff - magenta
-->
```

Вид сторінки на екрані

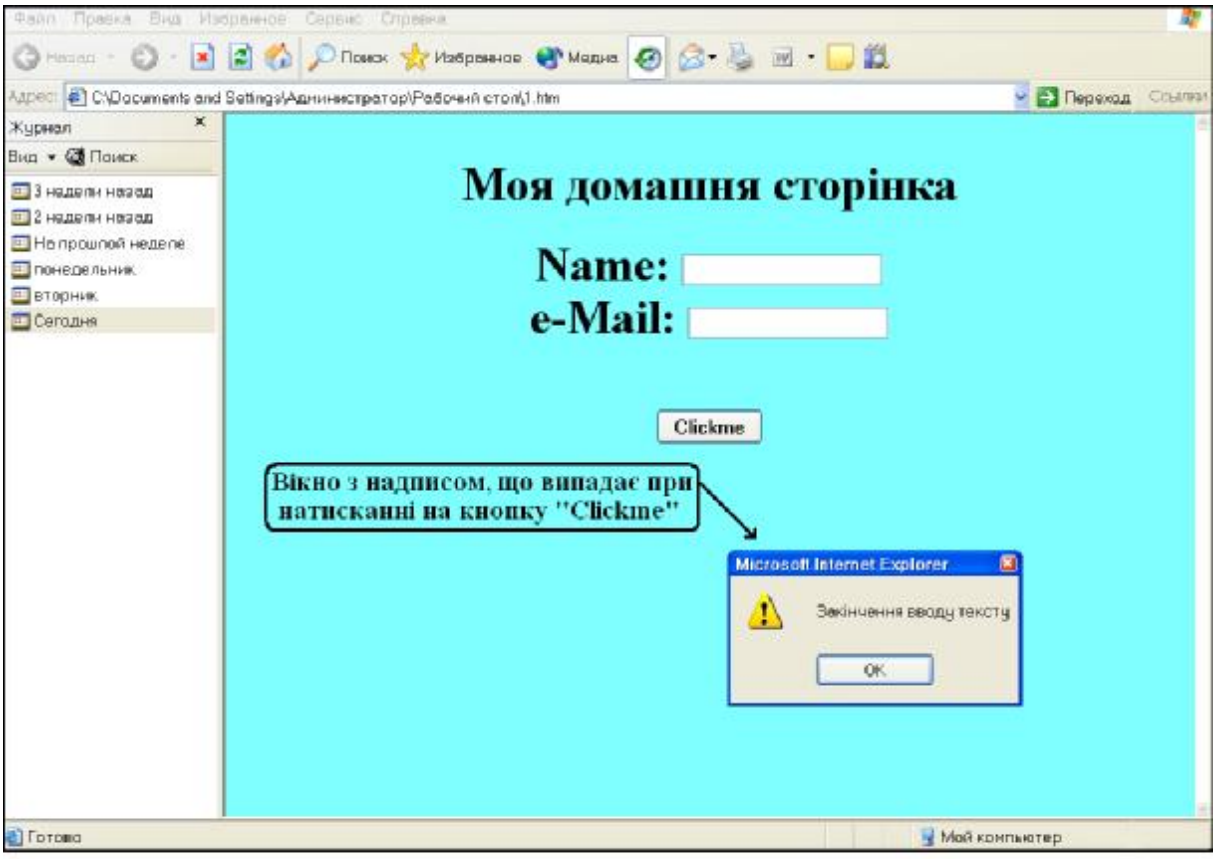

Рис.4

Ієрархія об'єктів HTML-сторінки:

1. Вікно браузера – об'єкт window. Він є головною сутністю, що містить у собі три дочірні сутності.
2. Напис "Моя домашня сторінка" – об'єкт HTML-сторінки. Він є дочірньою сутністю, котра підлегла головній сутності.

3. Форма із двома полями для вводу тексту **Name** й **e-Mail** – об'єкт HTML-сторінки. Форма є дочірньою сутністю, котра підлегла головній сутності.
4. Кнопка "Clickme", при натисканні якої виводиться повідомлення 'Закінчення вводу даних' – об'єкт HTML-сторінки. Кнопка є дочірньою сутністю, котра підлегла головній сутності.

Останній рядок програмного коду – **<!- -ff0000 - red 00ff00 - green 0000ff - blue ffff00 Yellow 00ffff - cyan ff00ff – magenta - ->** – коментар. У ньому представлене кодування наступних кольорів:

- ff0000 - red (червоний);
- 00ff00 - green (зелений);
- 0000ff - blue (блакитний);
- ffff00 yellow (жовтий);
- 00ffff - cyan (синій);
- ff00ff - magenta (бузковий).

З погляду мови програмування JavaScript вікно браузера – це об'єкт window. Він містить елементи оформлення. Усередині вікна розміщається документ HTML. Сконструйована сторінка, відповідно, є об'єктом **document**. До властивостей об'єкта document відносяться, наприклад, кольори web-сторінки. Усі без винятку об'єкти HTML є властивостями об'єкта document.

Ієрархія об'єктів, створювана HTML-сторінкою

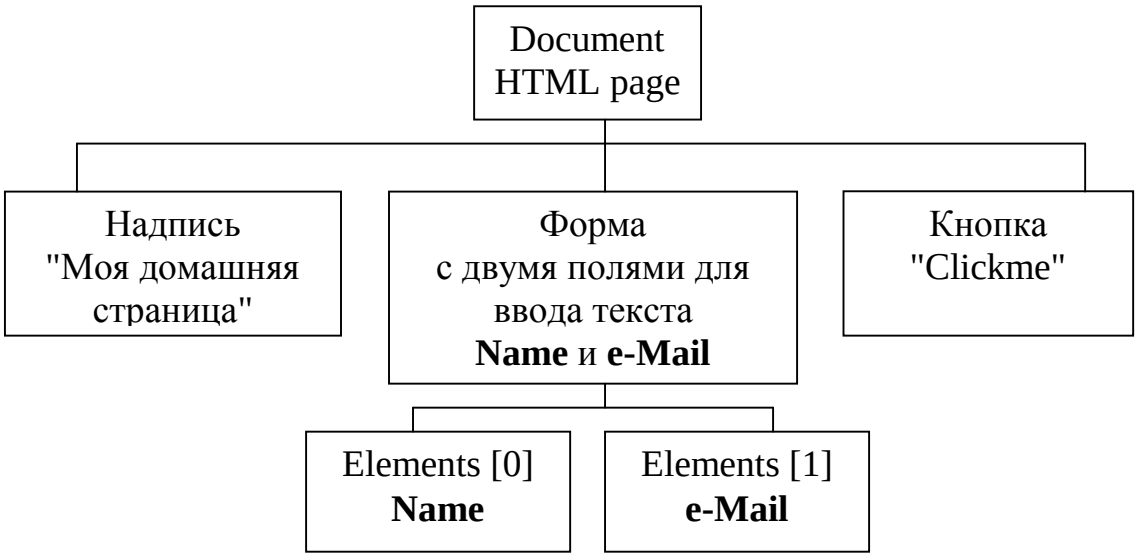

Рис.5

5.1 Імена об'єктів HTML-сторінки

Для керування об'єктами HTML-сторінки необхідно мати інформацію про їхню ієрархічну структуру. Кожен об'єкт ієрархічної структури має своє власне ім'я. Якщо хочемо звернутися до будь-якого об'єкта на сконструйованій вище HTML-сторінці, то варто почати обіг із самої вершини, з материнської сутності. Головний об'єкт структури називається **document.** Перший об'єкт на сторінці представлений як об'єкт **images[0].** Це означає, що ми можемо одержувати доступ до цього об'єкта, записавши в програмному коді JavaScript **document.images[0].**

Якщо, наприклад, хочемо знати, який текст ввів читач у перший елемент форми, то спершу варто з'ясувати, як одержати доступ до цього об'єкта. Починаємо з вершини створеної ієрархії об'єктів. Поступово простежуємо шлях до об'єкта з ім'ям **elements[0].** Послідовно записуємо назви всіх об'єктів, які минемо. Доступ до першого поля для вводу тексту можна одержати, записавши: **document.forms[0].elements[0].**

Елемент, що відповідає полю для вводу тексту, має властивість **value** (визначає дані, котрі слід відправляти). Шукане значення можна прочитати, написавши мовою JavaScript рядок:

name= document forms[0].elements[0].value;

Отриманий рядок заноситься в змінну **name**. Тепер можемо працювати безпосередньо із цією змінною. Наприклад, можемо створити вікно, що випадає, скориставшись **командою alert("Hi " + name).** У результаті, якщо ввести в це поле деякий вираз, то по команді **alert("Hi " + name)** буде відкрите вікно із заданим виразом.

Якщо маємо справу з великими сторінками, то процедура адресації різних об'єктів по номеру може бути досить заплутаною. Наприклад, прийдеться вирішувати, як варто звернутися до об'єкта **document.forms[3]. elements[17]** або до документа **Document.forms[2]. elements[18]?** Щоб уникнути подібної проблеми можна присвоювати об'єктам унікальні імена.

Приклад:

<form name="myForm"

Name:

**<input type="text" name="name" value="">
**

Цей запис означає, що об'єкт **forms[0]** одержав унікальне ім'я – **myForm**. Точно так само замість **elements[0]** можемо писати **name** (останнє було зазначено в атрибуті name тега **<input>**).

Таким чином, замість

name= document.forms[0].elements[0].value;

можемо записати

name= document.myForms.name.value;

Присвоювання об’єктам HTML-сторінки унікальних імен значно спрощує програмування на JavaScript, особливо у випадку з великими web-сторінками, що містять багато об'єктів. При написанні імен необхідно стежити за положенням регістра – тобто, можливо написати **tyform** замість **myForm).**

В мові програмування JavaScript багато властивостей об'єктів доступні не тільки для читання. Але, маємо можливість записувати в них нові значення. Наприклад, за допомогою JavaScript можемо записати в першому полі новий рядок замість введеного "Поле вводу тексту" у програмному коді, який представлено нижче.

Приклад коду на мові програмування JavaScript, що ілюструє таку можливість, запишемо як властивість onClick тега <input>:

```
<form name="myForm">
<input type="text" name="input" value="Поле уведення тексту">
<input type="button" value="Write" onClick="document myForm. input value="('Закінчення вводу тексту')">
```

Розглянемо приклад:

```
<html>
<head>
<title>Objects</title>
<script language="JavaScript">

function first() {
// функція створює вікно, котре випадає, у якому розміщається текст, введений у поле форми
```

```
alert("The value of the textelement is: " + document.myForm.myText.value);
}
function second() {
 // функція перевіряє стан перемикачів
var myString= "The checkbox is";
// перемикач включений чи ні?
if (document.myForm.myCheckbox.checked) myString+= "checked"
    else myString+= "not checked";
// вивід рядка на екран
 alert(myString);
}
</script>
</head>

<body bgcolor=#7fffff>
<h1 Align=center>
<form name="myForm">
<input type="text" name="myText" value="Поле вводу тексту">
<input type="button" name="button1" value="Button1" onClick="first()">
<br>
<input type="checkbox" name="myCheckbox" CHECKED>
<input type="button" name="button2" value="Button2" onClick="second()">
</form>
<p><br><br>
<script language="JavaScript">
document.write("The background color is:");
document.write(document.bgColor + "<br>");
document.write(document.myForm.button2.value);

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

Вид сторінки на екрані

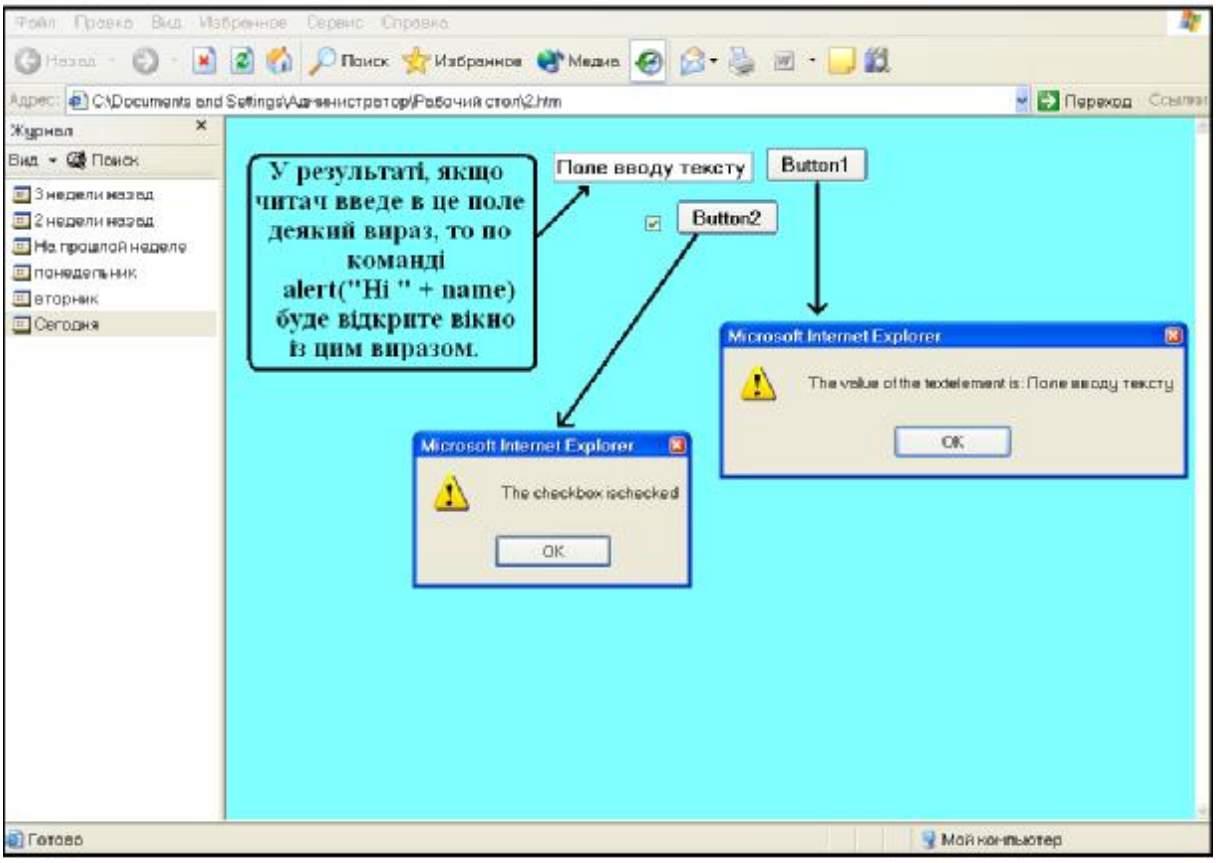

Рис.6

Висновок

Мова програмування JavaScript має досить гнучку систему керування об’єктами, які створюються за її використанням, в документах на HTML-сторінках. Для оволодіння системою керування об’єктами необхідно:

1) знання ієрархічної структури об’єктів HTML-сторінки;
2) знання імені об’єкта, до котрого йде звернення.

Будь-яка сконструйована HTML-сторінка є об'єктом **document** або материнською сутністю для об’єктів, котрі на ній розташовані. До властивостей об'єкта **document** відносяться усі без винятку об'єкти HTML-сторінки. Якщо хочемо звернутися до будь-якого об'єкта на сконструйованій HTML-сторінці, то варто починати обіг із самої вершини, з материнської сутності. Якщо маємо справу з великими сторінками, то процедура адресації об'єктів по номеру може бути досить складною. Щоб уникнути складностей, слід присвоювати об'єктам унікальні імена.

5.2 Об'єкт location

Керуючий тег, який використано для створення наступного сценарію Web-сторінки:

1. **<Href>** – базова адреса.

Крім об'єктів **window** й **document** в мові JavaScript є ще один об'єкт – **location.** У цьому об'єкті представлена адреса HTML-документа, що завантажується. Якщо завантажити сторінку **http://\\www.xyz.corn/page.html,** то значення **location.href** буде відповідати вказаній адресі.

У прикладі кнопка (рис.7) завантажує в поточне вікно нову Web-сторінку:

```
<form>
<input type=button value="Yahoo" onClick="location.href=
'http://www.yahoo.com';">
</form>
```

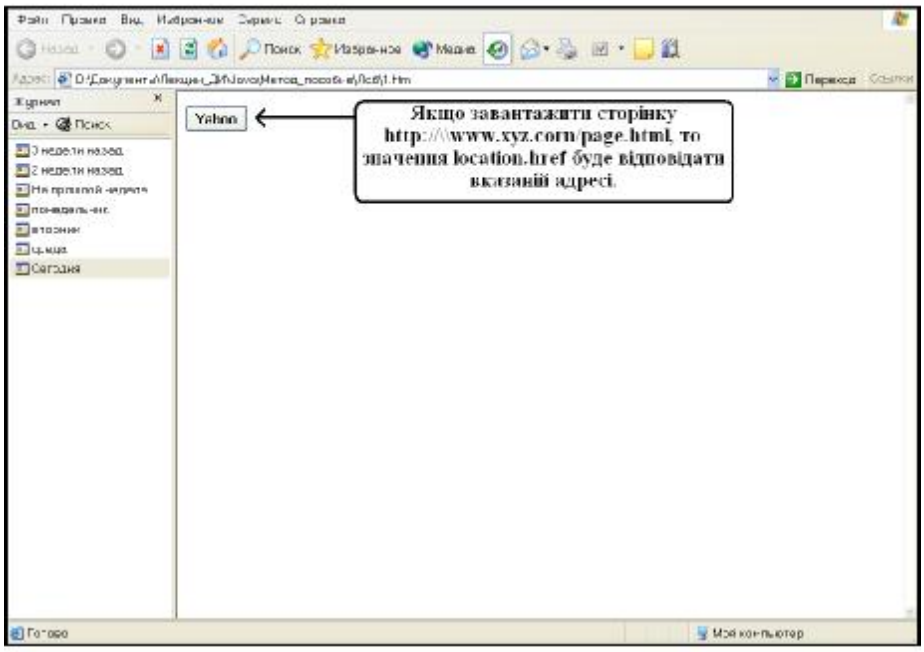

Рис.7

Висновок

Об’єкти **location** є підлеглими сутностями по відношенню до об’єктів **document.** Вони несуть в собі ім’я методу. Метод ніколи не унаслідується. Таким чином, об’єкти **location** не можуть мати власних дочірніх сутностей.

ТЕМА 6 ФРЕЙМИ В JAVASCRIPT

6.1 Створення фреймів

Фрейм – це виділене у вікні браузера поле у формі прямокутника. За допомогою HTML-коду вікно браузера розмічують на кілька окремих фреймів. Можна створити, наприклад, два фрейми. У перший фрейм завантажити «Свою домашню сторінку», у другий – інший документ.

Хоча створення фреймів є задачею мови гіпертекстової розмітки HTML, виникає необхідність описати основні положення цього процесу. Кожен із фреймів видає на екран зміст власного документа, котрий, в свою чергу, створюється за допомогою мови програмування JavaScript. Таким чином, фрейми слід розглядати і як об’єкти мови JavaScript.

Керуючі теги HTML-коду, які необхідні для розмітки web-сторінки:

1. **<Frameset> –** визначає набор певної кількості фреймів.
2. **<Frame> –** визначає сторінку серед набору фреймів.

Атрибут **Src** визначає адресу зовнішнього файлу сценарію.

Приклад HTML-коду (рис.8), якій надає можливість створити два фрейми:

```
<html>
<frameset rows="50%,50%">
<frame src="page1.htm" name="frame1">
<frame src="page2.htm" name="frame2">
</frameset>
</html>
```

У тегу **<frameset>** використана властивість **rows**. Це означає, що фрейми на web-сторінці розташовані один над одним. Верхній фрейм призначений для документа **page1.htm,** нижній – для документа **page2.htm.** Якщо необхідно розташувати фрейми з завантаженими у них документами поруч, то слід у тегу **<frameset>** використати властивість **cols** а не **rows**.

Запис **"50%,50%"** повідомляє, наскільки великими повинні бути вікна фреймів. Для одержання того ж результату можливо записати й **"50%, *"**. Якщо задаємо розмір фрейму в пікселах, то після числа не ставимо символ **%**.

Будь-якому фрейму можна присвоїти унікальне ім'я, скориставшись у тегу **<frame>** атрибутом **name.**

У результаті виконання HTML коду створені два фрейми

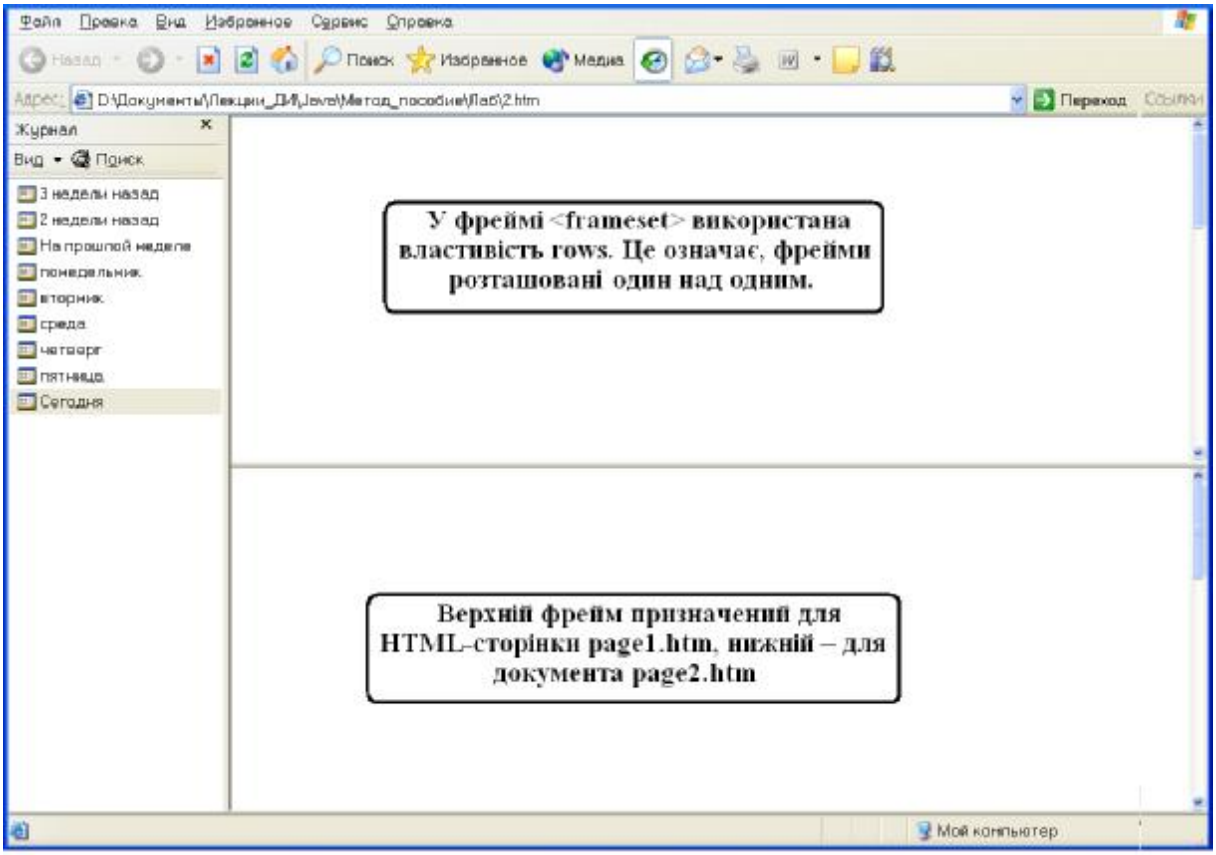

Рис.8

Для розмітки web-сторінок на необхідну кількість фреймів в HTML-коді використовують декілька вкладених тегів **<frameset>**.

Приклад HTML-коду (рис.9) для розмітки web-сторінки на п’ять фреймів:

```
<frameset cols="50%,50%">
<frameset rows="50%,50%">
<frame src="cell.htm">
<frame src="cell.htm">
</frameset>
<frameset rows="33%,33%,33%">
<frame src="cell.htm">
<frame src="cell.htm">
<frame src="cell.htm">
</frameset>
</frameset>
```

Створена структура фреймів буде мати наступній вигляд:

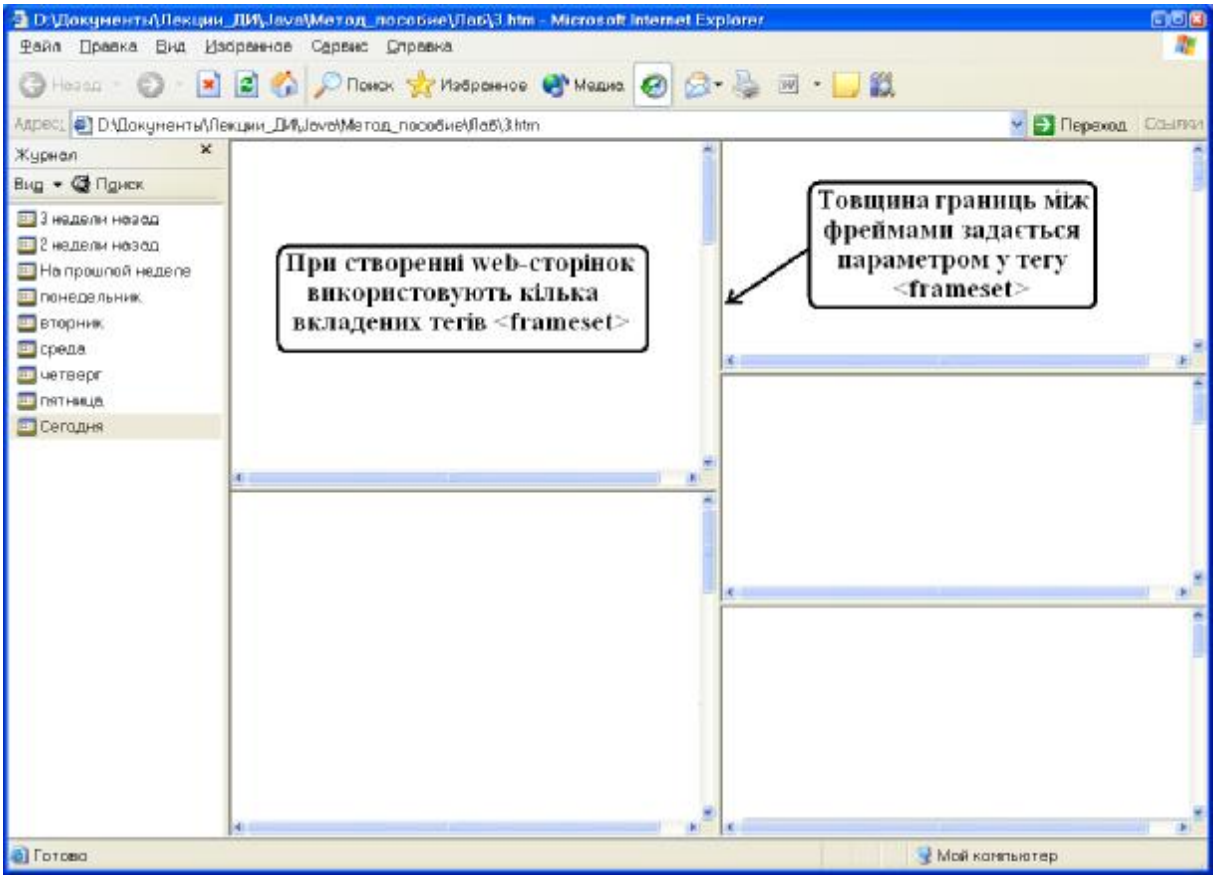

Рис.9

Можна задати товщину межі між фреймами, скориставшись у тегу **<frameset>** параметром **border.** Запис **border=0** означає, що між тегами є яка-небудь межа.

6.2 Фрейми як об’єкти мови програмування JavaScript

У мові програмування JavaScript всі елементи web-сторінки вибудувані в ієрархічній структурі, це відноситься й до фреймів. На рисунку 10 показана ієрархія об'єктів, створених за допомогою HTML-коду, який надає можливість розмітити web- сторінку на два фрейми.

У вершині ієрархії перебуває вікно браузера (browser window). Воно, як об'єкт, є родоначальником ієрархії (parent). Два фрейми – відповідно, його нащадки (children). Після присвоєння цим двом фреймам унікальних імен – **frame1** й **frame2**. виникає можливість обміну інформації між вікном браузера та зазначеними фреймами.

Ієрархія об'єктів web-сторінки

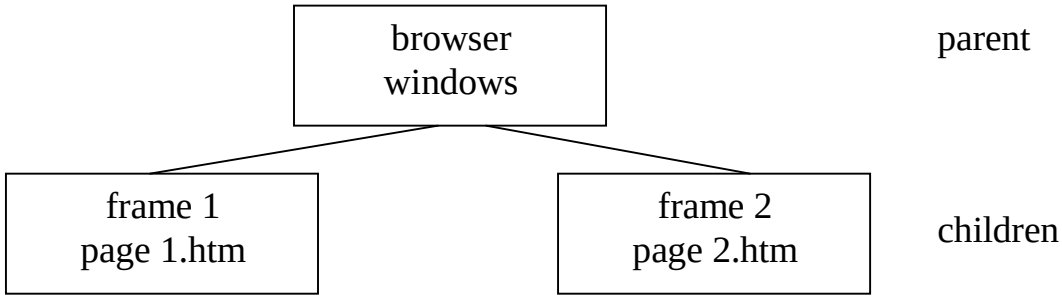

Рис.10

6.2.1 Умови навігації по сайту

Якщо знаємо унікальні імена фреймів, то за допомогою створеного на мові програмування JavaScript коду можна вирішити наступну задачу. Відвідувач сайту активує документ в першому фреймі. Документ повинен завантажуватися в інший фрейм.

Прикладом вирішення такої задачі може служити **складання меню** для користувача та **створення навігаційних панелей** в одному із фреймів, котрий завжди залишається незмінним.

Розглянемо три варіанти постановки задачі:

- Головне вікно/фрейм одержує доступ до фрейму-нащадка.
- Фрейм-нащадок одержує доступ до батьківського вікна/фрейму.
- Доступ від одного до іншого фрейму-нащадка.

Існує прямий взаємозв'язок між батьківським вікном і кожним із фреймів-нащадків (рис.11), яким відповідно надані імена **frame1** й **frame2**. Таким чином, якщо написати програмний код для батьківського вікна, тобто для сторінки, що створює ці фрейми, то можна звертатися й до фреймів, називаючи лише їх імена. Наприклад, можна написати:

frame2.document.write("Повідомлення від батьківського вікна.");

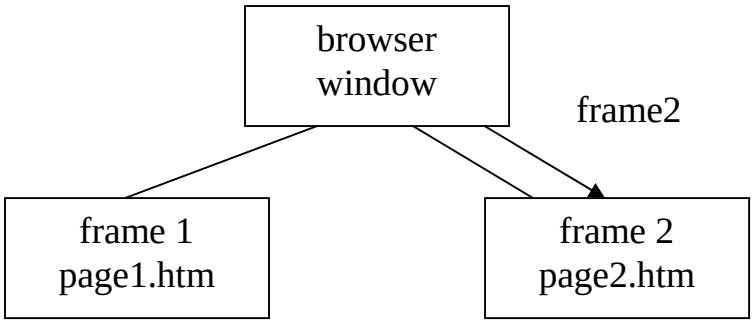

Рис.11

У деяких випадках, перебуваючи у фреймі, необхідно одержати доступ до батьківського вікна. Наприклад, якщо хочемо позбутися одного з фреймів. Видалення фрейму означає завантаження документа, що був у цьому фреймі, до батьківського вікна. Обумовимо доступ до батьківського – **parent –** вікна (або батьківському фрейму) із фреймів.

Щоб завантажити новий документ, необхідно внести в тег **location.href** нову **адресу URL**. Оскільки хочемо позбутися від одного з фреймів, то слід використати об'єкт **location,** призначений безпосередньо для батьківського вікна.

Нагадаємо, <Location> – тег визначення місцезнаходження елемента в ієрархічній структурі об’єкта **document**. Властивість **href,** в даному випадку, означає вершину ієрархічної структури елементів, або вихідний каталог, для котрого необхідно записати **адресу URL.**

Якщо в кожен із фреймів можна завантажити власну сторінку, то для кожного фрейму існує власний об'єкт LOCATION.

Завантажимо нову сторінку (рис.12) у батьківське вікно за допомогою команди: **parent. location.href= "http://...";**

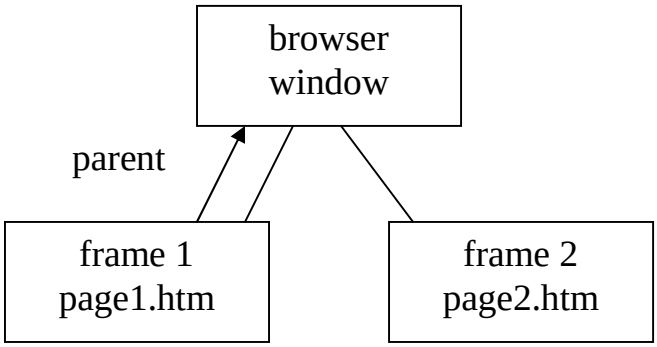

Рис.12

Часто доводиться вирішувати задачу забезпечення доступу від одного фрейму-нащадка до іншого такого ж фрейму-нащадка. Між цими двома фреймами немає прямого зв'язку. Ми не можемо викликати **frame2**, перебуваючи у **frame1.** Така задача вирішується через звертання до об'єкта **document.**

Щоб одержати доступ до об'єкта **document** (рис.13), необхідно в програмному коді JavaScript написати наступне:

parent.frame2.document.write("Привіт, це виклик з першого фрейму.");

Схема повідомлень між фреймами

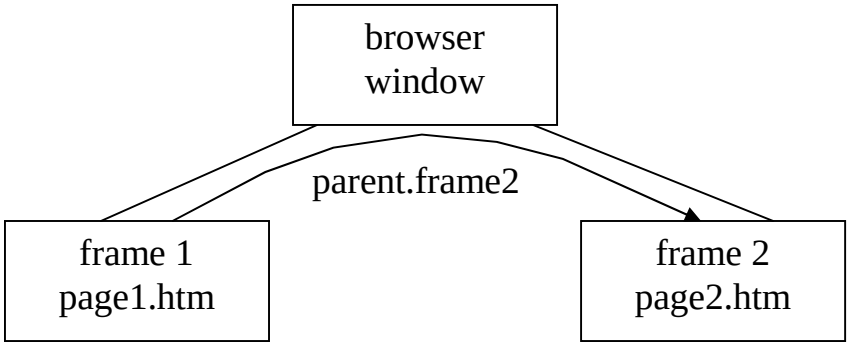

Рис.13

6.3 Навігаційні панелі та меню користувача

Для створення навігаційних панелей користуємося наступним HTML-кодом:

frames3.htm

```
<html>
<frameset rows="80,20%">
<frame src="start.htm" name="main">
<frame src="menu.htm" name="menu">
</frameset>
</html>
```

У представленому HTML-коді запис **"start.htm"** – це та сторінка, яка буде завантажена в головний фрейм (main). Наступна web-сторінка буде завантажена у фрейм **"menu"**(рис.14).

menu.htm

```
<html>
<head>
<script language="JavaScript">
function load(url) {
 parent.main.location.href=url;
}
</script>
</head>
```

```
</body>
<a href="JavaScript load('first.htm')">first</a>
<a href="second.htm" target="main">second</a>
<a href="third.htm" target="_top">third</a>
```

Створення меню користувача

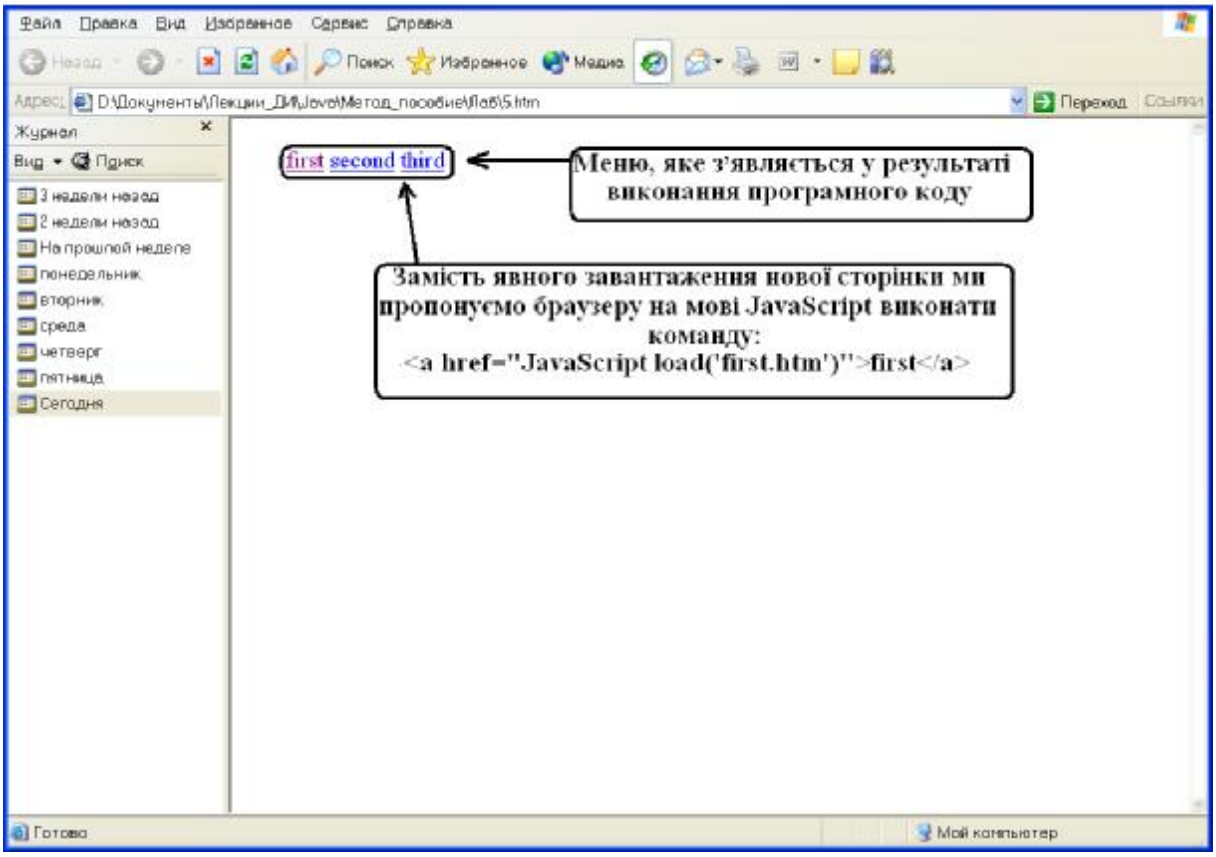

Рис.14

Розглянемо два способи завантаження нової сторінки у фрейм **main.** У першому способі для досягнення цілі скористаємося функцією **load().** Замість явного завантаження нової сторінки пропонуємо браузеру на мові JavaScript виконати команду:

first

У наведеному програмному коді використаний параметр **JavaScript** замість звичайного **href.** Усередині дужок стоїть **'first.htm'.** Цей запис передається як аргумент функції **load().** Сама ж функція **load()** записується таким чином:

```
function load(url) {
parent.main.location.href=url;
}
```

Параметр **url** означає, що рядок **'firstl.htm'** при виклику функції заноситься в змінну **url.** Нову змінну тепер можна використати при роботі усередині функцій **load().** Якщо необхідно позбутися від фреймів за допомогою функції **load(),** то слід написати: **parent.location.href=url.**

У другому способі скористаємося параметром **target,** який є конструкцією мови гіпертекстової розмітки HTML. Параметр **target** указує ім'я необхідного фрейму. Параметром **target** користуються, якщо завантажується нова сторінка в інший фрейм. Нехай, маємо три фрейми з іменами **frame1, frame2** й **frame3**. Припустимо, що відвідувач сайту активує посилання у **frame1.** Необхідно, щоб при цьому у два інших фрейми завантажувалися дві різні web-сторінки. При рішенні такого завдання скористаємося функцією:

```
function loadtwo() {
parent.frame1.location.href="first.htm";
parent.frame2.location.href="second.htm";
}
```

Якщо хочемо використати функцію багаторазово, то скористаємося можливістю передачі змінної як аргументу. Результат буде записаний таким чином:

```
function loadtwo(url1,url2) {
parent.frame1.location.href=url1;
parent.frame2.location.href=url2;
}
```

Передача змінної як аргументу робить функцію більше гнучкою. У результаті можна використати її багаторазово й у різних контекстах.

Висновок

Фрейми є об’єктами мови програмування JavaScript, по-перше, тому, що кожен з фреймів видає на екран зміст власного документа, котрий створюється за допомогою мови програмування JavaScript; по-друге, тому що фрейми є об’єктами в ієрархічній структурі web-сторінки.

Будь-якому фрейму можна присвоїти унікальне ім'я, скориставшись у тегу **<frame>** атрибутом **name.**

Оскільки в кожен із фреймів можна завантажити власну web-сторінку, то для кожного фрейму існує і власний об'єкт **location.**

ТЕМА 7 ВІКНА Й ДИНАМІЧНО СТВОРЮВАНІ ДОКУМЕНТИ В JAVASCRIPT

7.1 Створення вікон

Відкриття нових вікон у браузері обумовлено властивостями і методами мови програмування JavaScript. У нове вікно можна завантажити документи HTML або динамічно створені нові матеріали.

Приклад відкриття нового вікна й завантаження в нього web-сторінки

```
<html>
<head>
<script language="JavaScript">
function openWin() {
myWin=open("www.rambler.ru");
}
</script>
</head>
<body>
<form>
<input type="button" value="Відкрити нове вікно" onClick="openWin()">
</form>
</body>
</html>
```

У наведеному прикладі до нового вікна за допомогою методу **open()** записується сторінка "www.rambler.ru". По відношенню до властивостей нового вікна можна вказати, повинне воно чи ні мати рядок статусу, панель інструментів або меню. Для кожного вікна можна задати розмір.

Наприклад, відкриємо вікно розміром 400x300 пікселов.

```
<html>
<head>
<script language="JavaScript">
function openWin2() {
myWin=open("www.rambler.ru","displayWindow",
"width=400, height=300, status=no,toolbar=no,menubar=no");
}
```

```
</script>
</head>
<body>
<form>
<input type="button" value="Відкрити нове вікно" onClick="openWin2()">
</form>
</body>
</html>
```

Властивості вікна сформульовані в рядку "width=400, height=300, status=no,toolbar=no,menubar=no". **Не слід поміщати в цьому рядку символи пробілів.**

Таблиця 1 – **Список властивостей вікна, якими можна управляти**

Directories	Yes|No
Height	Кількість пікселов
Location	Yes|No$
Menubar	Yes|No
Resizable	Yes|No
Scrollbars	Yes|No
Status	Yes|No
Toolbar	Yes|No
Width	Кількість пікселов
AlwaysLowered	Yes|No
AlwaysRaised	Yes|no
Dependent	Yes]No
Hotkeys	Yes|No
InnerWidth	Кількість пікселов
InnerHeight	Кількість пікселов
OuterHeight	Кількість пікселов
Screen	Кількість пікселов
Screen	Кількість пікселов
Titlebar	Yes|no

7.2 Ім'я вікна

Відкриваючи вікно, у першій строчці, в дужках, запишемо аргументи: **myWin=open("www.rambler.ru","displayWindow".**

У другій визначемо розмір вікна:

"width=400, height=300, status=no,toolbar=no,menubar=no");

Аргумент **"displayWindow"-** це ім'я вікна. Якщо знаємо таке ім'я, то використавши запис:

<a href="www.rambler.ru" terget="displayWindow">, можна завантажити до вікна нову сторінку.

Якщо вікна з ім'ям **displayWindow** не існує, то буде створене нове вікно з ім'ям **MyWin.** Тепер тільки за допомогою змінної **MyWin** можемо одержати доступ до створеного вікна.

Область дії змінної **MyWin** – це лише той скрипт, у якому вона визначена. Ім'я вікна (у нашому випадку **"displayWindow"**) – унікальний ідентифікатор, яким можна користуватися із кожного вікна браузера.

7.3 Закриття вікон

Щоб закрити вікно, знадобиться метод **close()**.

Приклад JavaScript-коду для закриття вікна:

```
<html>
<script language="JavaScript">
function closeIt() {
close();
}
</script>
<center>
<form>
<input type=button value="Clickme" onClick="closeIt()">
</form>
</center>
</html>
```

Open() і close() – це методи об'єкта **window**. Варто писати не просто **open() і close()**, **а window.open()** і **window.close()**. У нашому випадку об'єкт window можна опустити. Немає необхідності писати префікс **window,** якщо хочемо викликати один з методів цього об'єкта.

7.4 Динамічне створення документів

Використовуючи мову програмування JavaScript, вирішимо завдання самостійного створення нових HTML-сторінок. Створимо простий HTML-документ та покажемо його у новому вікні.

Розглянемо приклад:

```
<html>
<head>
<script language="JavaScript">
function openWin3() {
myWin=open("", "displayWindow",
"width=500, height=400, status=yes, toolbar=yes, menubar=yes");
// відкрити об'єкт document для наступної печатки
myWin.document.open();
//генерувати новий документ
myWin.document.write("<html><head><title>On-the-fly");
myWin.document.write("</title></head><body>");
myWin.document.write("<center><font size=+3>");
myWin.document.write("This HTML-document has been created");
myWin.document.write("with the help of JavaScript!");
myWin.document.write("</font></center>");
myWin.document.write("</body></html>");
//закрити документ - (але не вікно!)
myWin.document.close();
}
</script>
</head>
<body>
```

<form>
<input type=button value="On-the-fly" onClick="openWin3()">
</form>
</body>
</html>

Розглянемо функцію **winOpen3()**. Перший аргумент цієї функції – порожній рядок (""); це значить, що не визначено конкретну адресу **URL**. Браузер не обробляє наявний документ, а додатково створює новий. Визначаємо змінну **myWin**. З її допомогою одержуємо доступ до нового вікна. У нашому випадку не можемо користуватися для цієї мети ім'ям вікна (**displayWindow**). Після того, як відкрили вікно, відкриваємо для запису об'єкт **document.** Робиться це за допомогою команди:

// відкрити об'єкт document для наступної печатки

myWin.document.open();

Команда не відкриває нового вікна – вона готує **document** до печатки (рис.15). Необхідно поставити перед **document.open()** приставку **myWin**, щоб одержати можливість робити запис в новому вікні. У наступних рядках скрипта за допомогою виклику **document.write()** формується текст нового документа:

//генерувати новий документ
myWin.document.write("<html><head><title>On-the-fly");
myWin.document.write("</title></head><body>");
myWin.document.write(" ");
myWin.document.write("This HTML-document has been created");
myWin.document.write("with the help of JavaScript!");
myWin.document.write(" ");
myWin.document.write("</body></html>");

За допомогою звичайних тегів HTML-коду записуємо зміст документа, котрий розміщується у вікні, що випадає при натиску користувача на кнопку, яка в нашому прикладі має назву **"On-the-fly".** По завершенні запису змісту документа необхідно закрити документ. Це робиться командою: **// закрити документ - (але не вікно!)**

myWin.document.close();

Якщо створені два фрейми з іменами **frame1** й **frame2**, можемо розміщати документи в тому або іншому фреймі. Для цього необхідно написати: **parent.frame2.document.open();**

parent.frame2.document.write("Here goes your HTML-code");

parent.frame2.document.close();

Документ HTML у вікні браузера

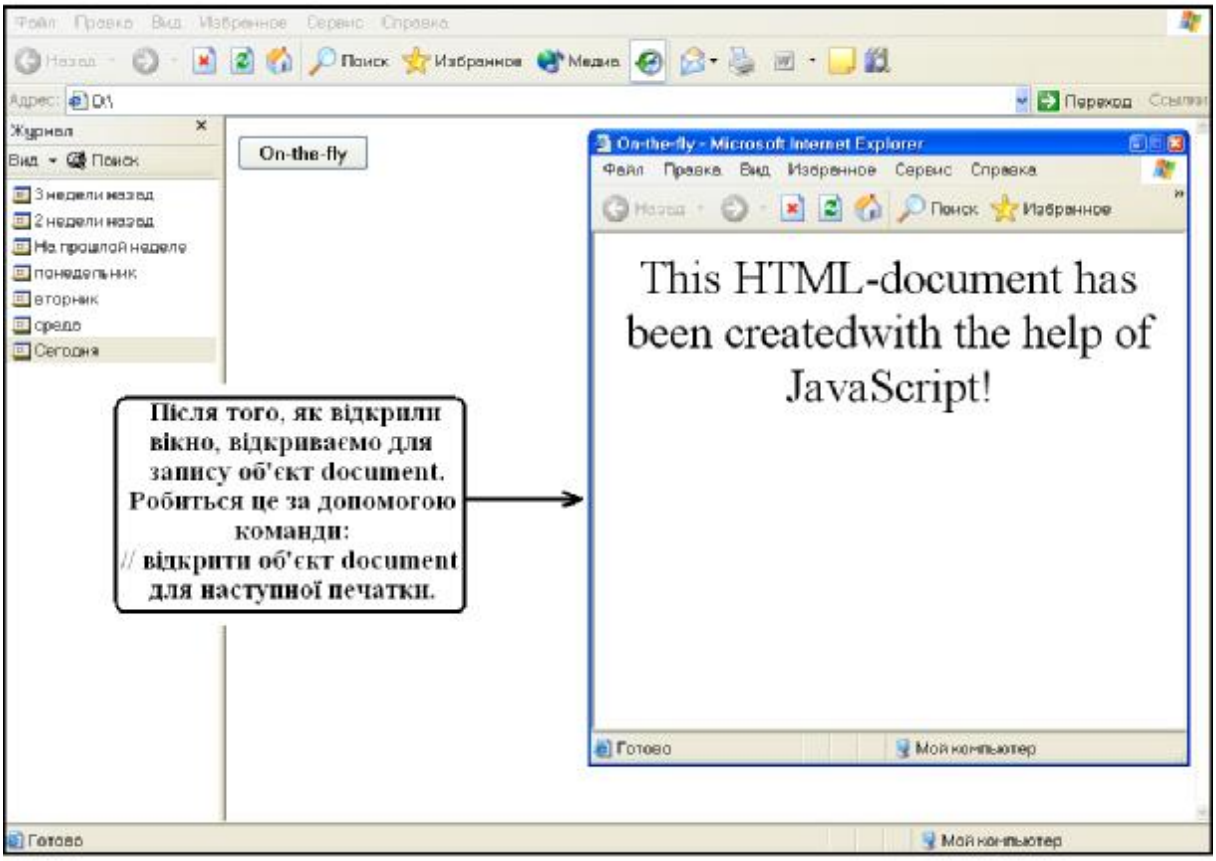

Рис.15

Висновок

Властивості і методи мови програмування JavaScript дозволяють створювати динамічні документи та загружати їх у вікна браузера. Найважливішими з розглянутих методів є: **open(),** який дозволяє відкрити вікно; **close()** – дозволяє закрити вікно. За допомогою функції **myWin** одержуємо доступ до нового вікна. Ім'я вікна – унікальний ідентифікатор, яким можна користуватися із кожного вікна браузера. Якщо створити декілька фреймів, то документ можна розмістити в тому чи іншому фреймі.

ТЕМА 8 РЯДОК СТАНУ Й ТАЙМЕРИ В JAVASCRIPT

8.1 Створення рядка стану об'єкта document в JavaScript

Рядок стану об'єкта **document** – це прямокутник нижньої частини вікна браузера. Функція **function statbar** мови програмування JavaScript дає можливість зробити запис тексту в рядок стану об'єкта **document**. Запис відбувається при виконанні події **onClick** – натисканні на створену в програмному коді JavaScript кнопку.

У наступному прикладі показане створення двох кнопок, які використані для запису тексту в рядок стану об'єкта document.

```
<html>
<head>
<script language="JavaScript">
function statbar(txt){
window.status=txt;
}
</script>
</head>
<body>
<form>
<input type="button" name="look" value="Писати!" onClick="statbar('Це вікно стану!');">
<input type="button" name="erase" value="Стерти!"onClick="statbar('вікно');">
</form>
</body>
</html>
```

При використанні коду мови програмування JavaScript створена форма із двома кнопками, які викликають функцію **statbar()**, котра задає зміст тексту в рядок стану об'єкта **document**. Перша подія (**onClick)** визначена як натискання клавіші "**Писати!"**. У результаті її виконання в рядку стану з’являється текст: "**Це вікно стану!"** Заданий текст, як змінна, пере-

дається функції **statbar()**. Функція **statbar()** у програмному коді записана таким чином:

```
function statbar(txt) {
window.status = txt;
}
```

У заголовку функції, **function statbar(txt),** у дужках визначена змінна **txt**. Це означає, що зміст тексту рядка стану поміщається в змінну **txt.** Передача функціям змінних – це прийом, що застосовується для надання їм гнучкості. Можна передати функції кілька різних аргументів. Для цього необхідно відокремити аргументи один від одного комами. Зміст змінної **txt** заноситься в рядок стану командою **window.status = txt**. Видалення тексту з рядка стану виконується як запис в **window.status** порожнього рядка.

8.2 Рядок стану й посилання

Механізм виводу тексту в рядок стану зручно використати при роботі з посиланнями. Замість того, щоб виводити на екран адресу (**URL**) даного посилання, можна в рядку стану об'єкта **document** коротко представити зміст наступної сторінки. Досить лише помістити покажчик маніпулятора «миші» над посиланням.

Приклад вихідного коду виглядає таким чином:

```
<a href="dontclck.html"
onMouseOver="window.status='don’t click me!';return true;"
onMouseOut="window.status"=";">link</a>
```

У представленому програмному коді процедури **onMouseOver** й **onMouseOut** використані для того, щоб виявити, коли покажчик «миші» проходить над посиланням. Браузер при цьому не виконує власний код обробки події **MouseOver**. У рядку стану показана лише адреса (**URL)** відповідного посилання.

Якщо не використати значення **true –** повернення**,** то відразу ж після того, як код буде виконаний, браузер перепише рядок стану. Текст, внесений у програмному коді, виявиться затертим, і користувач не зможе його побачити.

У деяких випадках необхідно друкувати символи лапок, наприклад, якщо хочемо вивести до вікна стану текст **Don't click me.** Оскільки цей ря-

док не переданий у процедуру обробки події **onMouseOver**, то для цієї мети використані одинарні лапки.

8.3 Таймери

За допомогою функції **Timeout** (або таймера) програмуємо комп'ютер на виконання подій, які відбуваються після закінчення заданого в програмному коді часу.

У наступному прикладі задана кнопка, яка відкриває вікно, що випадає, після закінчення трьох секунд.

```
<script language="JavaScript">
function timer(){
setTimeout("alert('Час минув!')",3000);
}
</script>
<form>
<input type="button" value="Timer"onClick="timer()">
</form>
```

SetTimeout() – це метод об'єкта window, що встановлює інтервал часу. Перший аргумент, заданий при виклику функції **function timer(),** – запис **("alert('Час минув!')".** Запис виводиться після закінчення зазначеного часу. У другому аргументі безпосередньо визначається час виконання події. При цьому, час необхідно вказувати в мілісекундах (3000 мілісекунд = 3 секунди).

Висновок

Рядок стану використовується для запису у ньому необхідного тексту. Це відбувається завдяки функції **function statbar** мови програмування JavaScript. У заголовку функції, у дужках, визначена змінна **txt**. Зміст тексту рядка стану поміщається в змінну **txt.** Механізм виводу тексту в рядок стану зручно використати при роботі з посиланнями.

За допомогою функції **Timeout** програмуємо комп'ютер на виконання подій, які відбуваються після закінчення заданого часу. **SetTimeout()** – метод об'єкта window, що встановлює інтервал часу.

ВПРАВИ ДО НАВЕДЕНИХ ТЕМ

Тема 1 Розміщення JavaScript на HTML-сторінці

Вправа 1

Використовуючи текст програмного коду, наведеного на сторінках 3-4, та теги **<Html>, <Body>,
,** виконайте наступні завдання:

1. Запишіть у програмі «Блокнот» програмний код, який виводить на печать в HTML- сторінці строки:

Це звичайний HTML документ.
А це JavaScript!
Знову документ HTML.

2. Збережіть файл спочатку як текстовий, з розширенням .txt, потім як файл HTML, з розширенням .htm.
3. Видолайте у програмі «Блокнот» з коду строки, які не пов’язані безпосередньо з мовою програмування JavaScript.
4. Новий файл збережіть з розширенням .htm.
5. Поясніть значення тегів: **<Html>, <Body>,
.**

Тема 2 Події JavaScript

Вправа 2

Використовуючи текст програмного коду, наведеного на сторінці 5, та теги **<Form>**, **<Input>, <Type>,** виконайте наступні завдання:

1. Запишіть у програмі «Блокнот» програмний код, який становить простий приклад програми обробки події **onClick.**
2. У програмному коді створіть кнопку, на котрій виводиться наступний надпис «Це подія JavaScript».
3. Збережіть файл спочатку як текстовий, з розширенням .txt, потім як файл HTML, з розширенням .htm.
4. Поясніть значення тегів: **<Form>**, **<Input>, <Type>.**
5. Який з тегів безпосередньо створює форму «кнопка».
6. Запишіть атрибути, котрі надають можливість оброблювати події.
7. З якою метою у команді document.write() використані подвійні лапки, а в конструкції alert() – одинарні.

Тема 3 Функції JavaScript

Вправа 3

Використовуючи програмні коди мови JavaScript, наведені на сторінках 7-8, та функцію **myFunction,** у програмі «Блокнот» запишіть код, який виведе на печать на HTML-сторінці надпис:

Ласкаво просимо на мою сторінку!
Це JavaScript!

Такий надпис, при використанні функції **myFunction,** повинен бути надрукований чотири рази.

1. Які команди є аргументами функції **myFunction()**
2. Яким чином у створеному програмному коді команди document.write() зв'язані воєдино й можуть бути виконані при виклику зазначеної функції.

Тема 4 Сумісне використання функцій та процедур обробки подій

Вправа 4

Використовуючи теги **<Head>, <Var>, <Alert>** та програмний код мови JavaScript, наведений на сторінці 9, за допомогою програми «Блокнот» створіть HTML-сторінку, на котрій:

1. Використана подія **onClick.**
2. При натисканні на кнопку здійснюється виклик функції **calculation().**
3. Процедура виклику події задана рядком:
   <input type="button" value="Calculate" onClick="calculation()">.
4. У процедурі **function calculation()** присутні змінні **X** та **Y,** яким відповідно надані значення: **x=20, y=50**.
5. У результаті виконання програмного коду повинна з’являтися кнопка з надписом **Calculate.**
6. При натисканні на кнопку **Calculate** спочатку з’явиться кнопка з надписом **Результат підсумовування 20+50.**
7. При натисканні на останню кнопку, з’явиться кнопка з надписом **70**.
8. Поясніть значення тегів **<Var>** та **<Alert>.**
9. Поясніть значення атрибуту **value.**
10. Яку функцію виконує команда **result.**
11. Для збереження яких типів величин можуть бути використані змінні.

Тема 5 Ієрархія об'єктів в мові програмування JavaScript

Вправа 5

Використовуючи теги **<Bgcolor> <H1> <Align> <Title> <Checkbox>** та програмний код мови JavaScript, наведений на сторінках 11-12, за допомогою програми «Блокнот» створіть HTML-сторінку, на котрій система керування об’єктами можлива лише при їх правильному розташуванні в ієрархічній структурі об'єкта window.

На створеній сторінці повинні бути відображені наступні об’єкти:

1. Вікно браузера – об'єкт window. Він є головною сутністю, що містить у собі три дочірні сутності.
2. Напис "Моя домашня сторінка" – об'єкт HTML-сторінки. Він є дочірньою сутністю, котра підлегла головній сутності.
3. Форма із двома полями для вводу тексту **Name** й **e-Mail** – об'єкт HTML-сторінки. Форма є дочірньою сутністю, котра підлегла головній сутності.
4. Кнопка "Clickme", при натисканні якої виводиться повідомлення 'Закінчення вводу даних', – об'єкт HTML-сторінки. Кнопка є дочірньою сутністю, котра підлегла головній сутності.
5. Вікно браузера повинно бути пофарбоване у бузковий колір.

Останній рядок (с.12) програмного коду – коментар, у котрому представлені деякі відомості про кодування кольорів.

6. Складіть схему ієрархічної структури об’єктів для створеного об’єкта window.

Вправа 6

Використовуючи основні положення про імена об’єктів документа window, викладені на сторінках 14-15 та програмний код мови JavaScript, представлений на сторінках 15-16, у програмі «Блокнот» створіть HTML-сторінку, на котрій містяться:

1. Checkbox (поле вводу та верифікації тексту).
2. Дві кнопки, котрі відповідно надають можливість: вносити текст у поле Checkbox, запам’ятовувати внесений у нього текст.
3. Поле об’єкту document повинне мати блакитний колір.
4. Звернення до об’єктів повинне бути розпочатим з материнської сутності.

Вправа 7

Використовуючи:

1. Керуючий тег **<Href>.**
2. Основні положення про імена об’єктів документа window викладені на сторінках 14-15.
3. Програмний код мови JavaScript, представлений на сторінці 18.

У програмі «Блокнот» створіть: об’єкт **location,** котрий надає можливість загрузити до фрейму сторінку сайту пошукової системи Yahoo.

Тема 6 Фрейми в JavaScript

Вправа 8

Використовуючи теги **<Frameset> <Frame>** та програмний код мови JavaScript, наведений на сторінці 19, за допомогою програми «Блокнот» створіть на HTML-сторінці:

1. Два фрейми, котрі володіють властивістю **rows.**
2. Верхній фрейм призначений для документа **page1.htm,** нижній – для документа **page2.htm.**
3. Вікна фреймів повинні мати розмір, визначений у програмному коді як **"30%,70%".**

Вправа 9

Використовуючи теги **<Frameset> <Frame>** та програмний код мови JavaScript, наведений на сторінці 20, за допомогою програми «Блокнот» створіть на HTML-сторінці:

1. П’ять фреймів, котрі володіють властивостями **frameset cols**="40%,60%" та **frameset rows** "33%,33%,33%".

Атрибут **Src** визначає адресу зовнішнього файлу сценарію.

Вправа 10

Використовуючи теги **<Frameset> <Frame>** та програмний код мови JavaScript, наведений на сторінці 24, за допомогою програми «Блокнот» створіть на HTML-сторінці:

1. Навігаційні панелі, котрі володіють властивістю **frameset rows="80,20%".**

Вправа 11

Використовуючи програмний код мови JavaScript, наведений на сторінці 24, за допомогою програми «Блокнот» створіть на HTML-сторінці:

1. **"Menu",** котре завантажує web-сторінку у фрейм.

Тема 7 Вікна й динамічно створювані документи в JavaScript

Вправа 12

Використовуючи програмний код мови JavaScript, наведений на сторінці 27, за допомогою програми «Блокнот» створіть:

1. Нове вікно браузера.
2. Завантажте в нього web-сторінку **www.rambler.ru**.
3. Відкрийте вікно розміром 400x300 пікселов.

Властивості вікна сформульовані на сторінці 27 у рядку **"width=400, height=300, status=no,toolbar=no,menubar=no"**.

Вправа 13

Використовуючи код мови JavaScript, наведений на сторінці 29, за допомогою програми «Блокнот» створіть:

4. Програму закриття вікна web-сторінки.

Щоб закрити вікно, знадобиться метод **close()**. Слід пам’ятати, що **"displayWindow"** – унікальний ідентифікатор, яким можна користуватися із кожного вікна браузера.

Вправа 14

Використовуючи програмний код мови JavaScript, наведений на сторінках 30-31, за допомогою програми «Блокнот» створіть:

1. Простий HTML-документ та покажіть його у новому вікні браузера.

Після того, як відкрили вікно за допомогою команди:

// відкрити об'єкт document для наступної печатки

myWin.document.open(); відкрийте для запису об'єкт **document.**

2. За допомогою виклику **document.write()** сформулюйте текст нового документа: **//генерувати новий документ.**
3. За допомогою звичайних тегів HTML-коду запишіть зміст документа, котрий при натиску на кнопку, яка має назву **"On-the-fly",** розміщується у вікні, що випадає.

Тема 8 Рядок стану й таймери в JavaScript

Вправа 15

Використовуючи програмний код мови JavaScript, наведений на сторінці 33, за допомогою програми «Блокнот» створіть:

1. Дві кнопки, які використовуються для запису тексту в рядок стану об'єкта document.
2. Функція **statbar()** повинна задавати зміст тексту в рядок стану об'єкта **document**.
3. Подія **onClick** повинна визначати натискання клавіші "**Писати!**".
4. У результаті виконання події (**onClick)** в рядку стану повинен з’явитися текст: "**Це вікно стану!".**

Вправа 16

Використовуючи програмний код мови JavaScript, наведений на сторінці 34, за допомогою програми «Блокнот» створіть:

1. Механізм виводу посиланнями рядка стану об'єкта **document.**
2. Посилання повинно коротко представити зміст наступної сторінки.

Вправа 17

Використовуючи програмний код мови JavaScript, наведений на сторінці 35, за допомогою програми «Блокнот» створіть:

1. Кнопку, яка відкриває вікно, що випадає, після закінчення п’яти секунд.
2. Вкажіть час відкриття вікна в мілісекундах.

Література

1. Белунцов В. Новейший самоучитель по разработке Web-страниц. – М.: ДЕСС КОМ, 2000. – 448.
2. Винтер Р., Винтер П. Microsoft Office 97. в 2-х т. Т.1 – СПб.: – BHV, 1997. – 453с.
3. Глушаков С. В., Сурядный А. С. Microsoft Office 2000: Учебный курс. – Харьков: Фолио, 2002. – 500с.
4. Джонс Э., Саттон Д. Библия пользователя Office 97. – К.: Диалектика, 1997. – 642с.
5. Краткий курс Microsoft Office 2000. – Ростов на Дону: Феникс, 2000. – 458с.

Зміст

Навчальне видання

Кузьменко В’ячеслав Віталійович
Швачич Геннадій Григорович
Романова Наталія Сергіївна
Байрак Віталій Васильович

Комп’ютерні технології в документознавстві
Розділ “Комп’ютерні мережі”

Довідкове керівництво по створенню Web-сторінок на основі мови програмування JavaScript з вправами
Навчальний посібник

Тем. план 2006, поз. 109

Підписано до друку 14. 02.06. Формат 60х84 1/16. Папір друк. Друк плоский. Облік.-вид. арк. 2,58. Умов.-друк. арк. 2,56. Тираж 100 пр. Замовлення №

Національна металургійна академія України
49600, Дніпропетровськ – 5, пр. Гагаріна, 4

Редакційно – видавничий відділ НМетАУ