Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Кафедра кібербезпеки та програмного забезпечення

Інтернет-технології та ресурси

МЕТОДИЧНІ ПОРАДИ

до лабораторних робіт з розділу “Основи web-програмування: мова JavaScript”

Для студентів напряму підготовки

029 "Інформаційна, бібліотечна та архівна справа"

Кропивницький – 2019

Марченко К.М. Інтернет-технології та ресурси : методичні поради до лабораторних робіт з розділу “Основи web-програмування: мова JavaScript”/ К.М. Марченко – Кропивницький: ЦНТУ, 2019. – 68с.

Для студентів напряму підготовки 029 "Інформаційна, бібліотечна та архівна справа" при вивченні навчальної дисципліни “Інтернет-технології та ресурси”. Визначено тематику лабораторних робіт, подані приклади створення web-сценаріїв за допомогою мови програмування JavaScript, приведені завдання та питання для самоконтролю.

Автор-укладач:

Марченко Костянтин Миколайович - канд. техн. наук, доцент кафедри кібербезпеки та програмного забезпечення

Затверджено на засіданні
кафедри КбПЗ, протокол №1
від 31.08.2018 р.

.

Зміст

Вступ

Верстка веб-сторінок - це створення структури html-коду, яка розміщує елементи веб-сторінки (зображення, текст і т.д.) у вікні браузера згідно з розробленим макетом таким чином, щоб елементи дизайну виглядали аналогічно до макета.

Спочатку дизайнер готує макети веб-сторінок у графічному редакторі (наприклад, Adobe Illustrator, Adobe Photoshop), затверджує їх у замовника і передає верстальнику для формування HTML-коду.

Верстальник отримує завдання у вигляді набору рисунок, кожен з яких відповідає макету окремої сторінки зі своїм дизайном. Необхідно проаналізувати рисунок і вирішити, як його перетворити в веб-сторінку. Для зручності відбувається логічне розбиття картинки на окремі блоки, з якими йде подальша робота.

Верстка веб-сторінок відрізняється від поліграфічної верстки тим, що необхідно враховувати різницю відображення елементів в різних браузерах, обмеження HTML і CSS, різницю в розмірах робочого простору пристроїв.

Хоча роботу верстальника не видно, саме вона забезпечує правильність відображення веб-сторінок і швидкість їх завантаження.

Всю величезну кількість існуючих сайтів можна розбити на 2 основні групи: статичні сайти і динамічні сайти.

Статичним прийнято називати сайт, що складається з незмінних HTML-сторінок. Статичні HTML-сторінки створюються вручну, після чого при кожному зверненні до сайту представляються користувачеві в незмінному вигляді. Щоб оновити інформацію на подібних сторінках, необхідно вручну внести зміни безпосередньо в HTML-код сторінки.

Динамічні сайти можуть «підлаштовуватися» під своїх відвідувачів, реагуючи на їхні дії. Для цього використовуються технології серверних, клієнтських скриптів, за допомогою яких і створюються сценарії поведінки сайту при певних діях користувачів.

JavaScript - це відносно проста об'єктно-орієнтована мова, призначена для створення невеликих клієнтських і серверних додатків для Internet. Програми, написані мовою JavaScript (скрипти), включаються до складу HTML-документів.

Прикладами програм на JavaScript можуть служити програми, які перевіряють введені користувачем дані або виконують певні дії при відкритті або закритті документа. Такі програми можуть реагувати на дії користувача - натискання кнопок "миші", введення даних в екранній формі або переміщення "миші" по сторінці. Більш того, JavaScript-програми можуть керувати самим браузером, атрибутами документу і навіть генерувати веб-сторінки.

Лабораторна робота №1

Основні положення мови програмування JavaScript

Програма (сценарій) на мові JavaScript є послідовність операторів з "крапкою з комою" (;) між ними. Якщо кожен оператор розміщується на одному рядку, то роздільник можна не писати. Один оператор може розташовуватися на декількох рядках.

У програмах на JavaScript можна використовувати коментарі. Для того щоб задати коментар, розташований на одному рядку, досить перед його текстом поставити дві косі риси (//). Якщо ж пояснювальний текст займає кілька рядків, то його слід укладати між символами /* і * /. В JavaScript малі та великі літери алфавіту вважаються різними символами. Будь-яка мова програмування оперує з постійними і змінними величинами. В JavaScript це літерали і змінні.

Визначення: Найпростіші дані, з якими може оперувати програма, називаються літералами.

Літерали не можуть змінюватися. Літерали цілого типу можуть бути задані в десятковому (по підставі 10), шестнадцатеричном (по підставі 16) або вісімковому (по підставі 8) виставі. Шістнадцятиричні числа включають цифри 0-9 і букви a, b, c, d, e, f. Шістнадцятиричні числа записуються з символами 0x перед числом, наприклад, 0x25, 0xa1,0xff. Запис речового литерала відрізняється від запису дійсного числа в математиці тим, що замість коми, що відокремлює цілу частину від дробової, вказується точка, наприклад, 123.34, -22.56. Крім того, для запису дійсних чисел можна використовувати так звану експонентну форму.

Крім цілих і речових значень в мові JavaScript можуть зустрічатися так звані логічні значення. Існують тільки два логічних значення: істина і брехня. Перше видається літералом true, друге - false. У деяких реалізаціях JavaScript може бути використана одиниця як true, і нуль як false.

Строковий літерал представляється послідовністю символів, укладеної в одинарні або подвійні лапки. Прикладом строкового литерала може бути рядок "результат" або 'результат'.

Визначення: Елемент, який використовується для зберігання даних, називається змінною.

Тип змінної залежить від збережених у ній даних, при зміні типу даних змінюється тип змінної. Визначити змінну можна за допомогою оператора var, наприклад: var test1.

В даному випадку тип змінної test1 не визначений і стане відомий тільки після присвоєння змінної деякого значення. Оператор

var можна використовувати і для ініціалізації змінної, наприклад, конструкцією var test2 = 276 визначається змінна test2 і їй присвоюється значення 276.

Значення змінної змінюється в результаті виконання оператора присвоювання. Оператор присвоювання може бути використаний в будь-якому місці програми і здатний змінити не тільки значення, але і тип змінної. Оператор присвоювання виглядає так: a = b, де a - змінна, якій ми хочемо задати деяке значення; b - вираз, що визначає нове значення змінної.

Нехай в сценарії описані наступні змінні

```
var n = 3725
 var x = 2.75
 var p = true
 var s = "Виконання завершено"
```

n і x мають тип number, тип змінної p - логічний, змінна s має тип string. В JavaScript визначено тип function для всіх стандартних функцій і функцій, визначених користувачем. Об'єкти JavaScript мають тип даних object. Змінні типу object часто називають просто об'єктами, вони можуть зберігати об'єкти.

Визначення: Вирази будуються з літералів, змінних, знаків операцій, дужок. Залежно від типу обчисленого значення виразу можна розділити на арифметичні, логічні та рядкові.

Операції відношення застосовні до операндам будь-якого типу. Результат операції - логічне значення true, якщо порівняння вірно, і false - в протилежному випадку.

Пріоритет операцій визначає порядок, в якому виконуються операції в вираженні.

Сценарії, написані на мові JavaScript, можуть розташовуватися безпосередньо в HTML-документі між тегами <script> і </script>.

Одним з параметрів тега <script> є language, який визначає використовуваний мову сценаріїв. Для мови JavaScript значення параметра дорівнює "JavaScript". Якщо застосовується мова сценаріїв VBScript, то значення параметра має бути рівним "VBScript". У разі використання мови JavaScript параметр language можна опускати, т. К. Ця мова вибирається браузером за замовчуванням.

Зазвичай браузери, які не підтримують будь-які теги HTML, ці теги просто ігнорують. Спроба браузера проаналізувати вміст не підтримуваних тегів може привести до невірного відображення сторінки. Щоб уникнути такої ситуації, рекомендується поміщати оператори мови JavaScript в теги коментаря <!- ... ->. Для правильної роботи інтерпретатора перед закриваючим тегом коментаря -> слід

поставити символи //.

Отже, для розміщення сценарію в HTML-документі слід написати наступне:

<Script language = "JavaScript">

</Script>

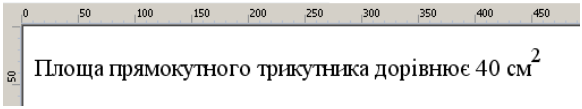
Документ може містити декілька тегів <script>. Всі вони послідовно обробляються інтерпретатором JavaScript. У наступному прикладі в розділ <body> (в тіло) HTML-документа вставлені оператори мови JavaScript.

Приклад 1. Обчислення площі трикутника

Необхідно написати сценарій, який визначає площу прямокутного трикутника за заданими катетам. Сценарій розмістимо в розділі <body> HTML-документа

```
<body>
    <script>
      var a = 8; h = 10 /* Ініціалізувалися дві змінні */
        document.write ("Площа прямокутного трикутника дорівнює ",
          a * h / 2, " см<sup>2</sup>")
          /* Для формування виведення використовується метод write
          об'єкта document */
    </script>
</body>
```

Результат:



Завдання:

1. Перевірити приклад з лабораторної роботи.
2. Скласти сценарій, в якому обчислюється площа кола по заданому радіусу.
3. Скласти сценарій, який обчислює гіпотенузу по заданих катетам.

Контрольні питання:

1. Яке призначення мови програмування JavaScript?
2. Якими позначками відокремлюються сценарії JavaScript у коді веб-сторінки?
3. У яких розділах веб-сторінки можуть бути розміщені сценарії

JavaScript?

4. Якими символами відокремлюються оператори сценарії JavaScript?

5. Якими символами позначаються коментарі до сценаріїв JavaScript?

6. Що називається літералами?

7. Що називається змінною? Як надати значення змінній?

8. Які типи змінних передбачені у мові JavaScript?

9. Як впровадити сценарій JavaScript у код веб-сторінки?

10. Як вивести результати роботи сценарію у вікно браузера?

Лабораторна робота №2

Функція і обробка події

Основним елементом мови JavaScript є функція. Опис функції має вигляд

function F (V) {S},

де F - ідентифікатор функції, що задає ім'я, по якому можна звертатися до функції; V - список параметрів функції, поділюваних запитом; S - тіло функції, в ньому задаються дії, які потрібно виконати, щоб отримати результат. Необов'язковий оператор return визначає повернення функцією значення. Зазвичай все визначення та функції задаються в розділі <head> документа. Це забезпечує інтерпретацію і збереження в пам'яті всіх функцій при завантаженні документа в браузер.

Приклад 1. Знаходження площі трикутника.

У попередніх прикладах користувачеві не надавалася можливість вводити значення, і в залежності від них отримувати результат. Інтерактивні документи можна створювати, використовуючи форми. Припустимо, що ми хочемо створити форму, в якій поля Основа і Висота служать для введення відповідних значень. Крім того, в формі створимо кнопку Обчислити. При натисканні мишею по цій кнопці ми хочемо отримати значення площі трикутника. Дія користувача (наприклад, клацання кнопкою миші) викликає подію. Події в основному пов'язані з діями, здійсненими користувачем з елементами форм HTML. Зазвичай перехоплення і обробка події задається в параметрах елементів форм. Ім'я параметра обробки події починається з приставки on, за якою слідує ім'я самої події. Наприклад, параметр обробки події click буде виглядати як onclick.

При інтерпретації HTML-сторінки браузером створюються об'єкти JavaScript. Взаємозв'язок об'єктів між собою являє ієрархічну

структуру. На самому верхньому рівні ієрархії знаходиться об'єкт windows, що представляє вікно браузера.

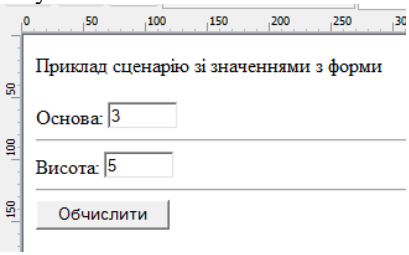
```
<HTML>
<HEAD>
    <Title> Обробка значень з форми </title>
    <Script language = "JavaScript">
        function care (a, h)
            {
            var s = (a * h) / 2;
            document.write ( "Площа прямокутного трикутника дорівнює", s);
            return s
            }
    </Script>
</HEAD>
<BODY>
    <P> Приклад сценарію зі значеннями з форми </P>
    <FORM name = "form1">
        Основа: <input type = "text" size = 5 name = "st1"> <hr>
        Висота: <input type = "text" size = 5 name = "st2"> <hr>
        <Input type = "button" value = Обчислити
        onClick = "care (document.form1.st1.value, document.form1.st2.value)">
    </FORM>
</BODY>
</HTML>
```

Об'єкт windows є предком або батьком усіх інших об'єктів. Кожна сторінка крім об'єкта windows має об'єкт document. Властивості об'єкта document визначаються вмістом самого документа: колір фону, колір шрифту і т.д. Для отримання значення основи трикутника, введеного в першому полі форми, повинна бути виконана конструкція

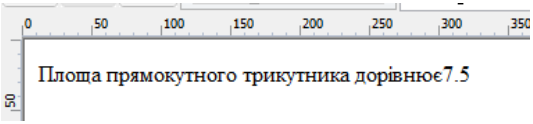
document.forml.stl.value

тобто, кажучи українською мовою (при цьому читаємо з кінця), використовуємо дані value з поля введення з ім'ям st1, яке знаходиться на формі form1 об'єкта document.

Результат



Після натиску кнопки «Обчислити» отримуємо

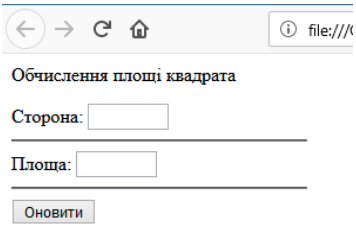


Приклад 2. Обчислення площі квадрата (Реакція на подію Change).

Напишемо сценарій, що визначає площу квадрата по заданій стороні. Площа повинна обчислюватися в той момент, коли змінилося значення його боку. Нехай форма містить два текстових поля: одне для довжини сторони квадрата, інше для обчисленої площі. Кнопка Оновити очищає поля форми. Площа квадрата обчислюється при виникненні події change, яке відбувається в той момент, коли значення елемента форми з ім'ям num1 змінилося, і елемент втратив фокус. HTML-код представлений в прикладі 2.

```
<HTML>
<HEAD>
    <Title> Обробка події Change - зміна значення елемента </ title>
    <Script>
        function srec (obj)
            {Obj.res.value = obj.num1.value * obj.num1.value}
    </Script>
</HEAD>
<BODY>
    <P> Обчислення площі квадрата </P>
    <FORM name = "form1">
        Сторона: <input type = "text" size = 7 name = "num1"
        onChange = "srec (form1)">
        <hr align="left" width="250">
        Площа: <input type = "text" size = 7 name = "res">
        <hr align="left" width="250">
        <Input type = "reset" value = Оновити>
    </FORM>
</BODY>
</HTML>
```

Результат:



Подія Focus виникає в момент, коли користувач переходить до елементу форми за допомогою клавіші <Tab> або клацання миші. Подія "втрата фокусу" Blur відбувається в той момент, коли елемент форми втрачає фокус. Подія select викликається вибором частини або всього тексту в текстовому полі. Наприклад, клацнувши двічі мишею по полю, ми виділимо поле, настане подія select, обробка якого призведе до обчислення необхідного значення.

У мові JavaScript визначені деякі стандартні об'єкти і функції, користуватися якими можна без попереднього опису. Одним із стандартних об'єктів є об'єкт Math. У властивостях згаданого об'єкта зберігаються основні математичні константи, а його методи можна використовувати для виклику основних математичних функцій.

Деякі методи об'єкта Math.

Math.abs() - абсолютне значення числа.
Math.acos() - арккосинус.
Math.asin() - арксинус.
Math.atan() - арктангенс.
Math.ceil() - повертає найбільше найближче ціле число.
Math.cos() - косинус.
Math.cbrt() - кубічний корінь.
Math.exp() - e^x.
Math.floor() - повертає найменше найближче ціле число.
Math.log() - натуральний логарифм.
Math.log10() - логарифм за основою 10
Math.log2() - логарифм за основою 2.
Math.max() - максимальне число.
Math.pow() - взводить число в степінь.
Math.random() - випадкове число від 0 до 1.
Math.sin() - синус
Math.sqrt() - квадратний корінь.
Math.tan() - тангенс.
Math.trunc() - повертає ціле число.
Наприклад, вираз y = log x запишеться y = Math.log (x).

Завдання:

1. Перевірити приклади з лабораторної роботи.
2. На площині задані координати трьох точок. Напишіть сценарій, який обчислює площу трикутника (використовувати подія Focus).
3. Напишіть сценарій, який для точки, заданої координатами на площині, визначає відстань до початку координат (використовувати подія Select).
4. Напишіть сценарій, який обмінює місцями значення двох введених змінних (використовувати подія Blur).

Контрольні питання:

1. Яке призначення мають функції JavaScript?
2. Як додається у скріпт функція?
3. Що називається тілом функції?
4. У якому розділі веб-сторінки слід розміщувати функції?
5. Що у програмуванні називають подіями?
6. Як здійснюється обмін даними між формою та функцією?
7. Який об'єкт є головним у ієрархії об'єктів?
8. Що являють собоб події з іменами Click, Change, Focus, Select?
9. Яке призначення об'єкту Math?
10. Як обчислити степень числа x та присвоїти результат числу y?

Лабораторна робота №3

Організація розгалужень в програмах

При складанні програми часто необхідне виконання різних дій в залежності від результатів перевірки деяких умов. Для організації розгалужень можна скористатися умовним оператором, який має вигляд:

if B {S1} else {S2}

де B - вираз логічного типу; S1 і S2 - оператори. Виконання умовного оператора здійснюється наступним чином. Обчислюється значення виразу B. Якщо воно істинне, то виконуються оператори S1, якщо помилково - оператори S2. Якщо послідовність операторів S1 або S2 складається лише з одного оператора, то фігурні дужки можна опустити. Можлива скорочена форма умовного оператора:

if B {S}

де B - вираз логічного типу; S - послідовність операторів. Виконання короткого умовного оператора здійснюється так: обчислюється значення виразу B, якщо воно істинне, то виконуються оператори S.

Приклад 1. Знаходження максимального значення

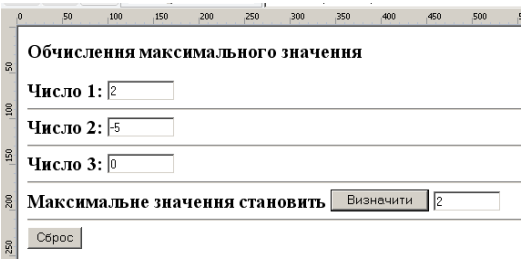
Для трьох заданих значень a, b, c необхідно написати сценарій, який визначає максимальне значення. Поступимо таким чином. Спочатку максимальним значенням m будемо вважати значення a, далі значення b порівняємо з максимальним. Якщо виявиться, що b більше m, то максимальним стає b. І, нарешті, значення c порівнюється з максимальним значенням з попередніх значень a і b. Якщо c більше m, то максимальним стає c. оператор присвоювання

obj.res.value = m

забезпечує запис обчисленого максимального значення у відповідне поле форми. Функція Number (s) перетворює об'єкт s, заданий як параметр, в число. Повністю сценарій може бути записаний так, як представлено в прикладі 1.

```
<HTML>
<HEAD>
    <TITLE> Обчислення максимального значення </TITLE>
    <Script language = "JavaScript">
        function maxval (obj)
            {
            var a = Number (obj.num1.value);
            var b = Number (obj.num2.value);
            var c = Number (obj.num3.value);
            var m = a
            if (b> m) m = b
            if (c> m) m = c
            obj.res.value = m}
    </Script>
</HEAD>
<BODY>
    <H4> Обчислення максимального значення </ H4>
    <FORM name = "form1">
        Число 1: <input type = "text" size = 8 name = "num1"> <hr>
        Число 2: <input type = "text" size = 8 name = "num2"> <hr>
        Число 3: <input type = "text" size = 8 name = "num3"> <hr>
        Максимальне значення становить
        <Input type = "button" value = Визначити onClick = "maxval (form1)">
        <Input type = "text" size = 8 name = "res"> <hr>
        <Input type = "reset">
    </FORM>
</BODY>
</HTML>
```

Результат роботи сценарію:



Вирішимо розглянуту задачу іншим способом. Згадаймо, що стандартний об'єкт Math має метод max, який визначає найбільше значення двох аргументів. Наведемо функцію maxval1, яка визначає максимальне значення з трьох заданих значень і використовує об'єкт Math.

```
function maxval1 (obj)
 {
 var a = Number (obj.num1.value);
 var b = Number (obj.num2.value);
 var з = Number (obj.num3.value);
 obj.res.value = Math.max (Math.max (a, b), c)
 }
```

Якби було потрібно визначити максимальне з чотирьох заданих значень a, b, c, d, то можна було б скористатися формулою

Math.max (Math.max (a, b), Math.max (c, d)).

Завдання

1. Перевірте приклад з лабораторної роботи.
2. Вводиться послідовність з п'яти чисел. Напишіть сценарій, в якому визначається число максимальних елементів.
3. Напишіть програму, яка визначає, чи можна побудувати трикутник із заданими довжинами сторін.
4. Точка на площині задається своїми координатами. Визначте, який з чвертей прямокутної системи координат належить задана точка.

Контрольні питання:

1. Яке призначення має розгалуження у комп’ютерних програмах?
2. Як організувати розгалуження у програмі?
3. Написання логічних операторів у мові JavaScript.
4. Що називається логічним виразом? Які значення повертають логічні вирази?

5. Яке написання має умовний оператор JavaScript?

6. Описати алгоритм знаходження максимального числа за допомогою умовного оператора.

7. Як створити поле для введення тексту на формі?

8. Як створити кнопку на формі?

9. Як обчислити максимальне значення за допомогою методу об’єкту Math?

Лабораторна робота №4

Методи в JavaScript

Під час інтерпретації HTML-документа браузером створюються об'єкти JavaScript. Властивості об'єктів визначаються параметрами тегів мови HTML. Структура документа відбивається в ієрархічній структурі об'єктів, відповідних HTML-тегами. Батьком всіх об'єктів є об'єкт windows, розташований на самому верхньому рівні ієрархії, він представляє вікно браузера і створюється при запуску браузера. Для того щоб відкрити нове вікно в сценарії JavaScript і відобразити в ньому новий документ, застосовується метод open, для закриття вікна можна скористатися методом close. Метод alert об'єкта windows відображає діалогове вікно з текстом, переданим методу в якості параметра. Даний метод використовується у випадках перевірки правильності даних, що вводяться за допомогою форми. Властивості об'єкта windows відносяться до всього вікна, в якому відображається документ.

Підлеглими об'єктами (або об'єктами нижнього рівня) є об'єкти document, history, location, frame. Властивості об'єкта history представляють адреси раніше завантажуються HTML-сторінок. Властивості об'єкта location пов'язані з URL-адресою відображуваного документа, об'єкта frame - зі спеціальним способом представлення даних.

Властивості об'єкта document визначаються вмістом самого документа: шрифт, колір фону, форми, зображення і т.д. Об'єкт document в залежності від свого вмісту може мати об'єкти, які є для нього підлеглими або дочірніми. Зокрема підлеглими для об'єкта document є об'єкти form, image, link, area і ін. Ієрархічна структура об'єктів представлена на рис. 1.

Для кожної сторінки створюється один об'єкт document, деякі його властивості відповідають параметрам тега <BODY>: bgColor, fgcolor, linkcolor, alinkcolor, vlinkColor. Методи write і writeln записують в документ текст, що задається параметром.

Якщо документ містить зображення, то доступ до об'єкта, який

визначає зображення, можна отримати за допомогою змінної, зазначеної в параметрі name тега <Img>. Об'єкт image має властивість images, яка містить посилання на всі зображення, розташовані в документі. Посилання перенумеровані, починаючи з нуля. Доступ до першого зображенню можна отримати за допомогою складеної конструкції document.images [0], до другого - document.images [1]. Якщо на сторінці п'ять зображень, то доступ до останнього зображенню можна отримати, скориставшись посиланням document.images [4].

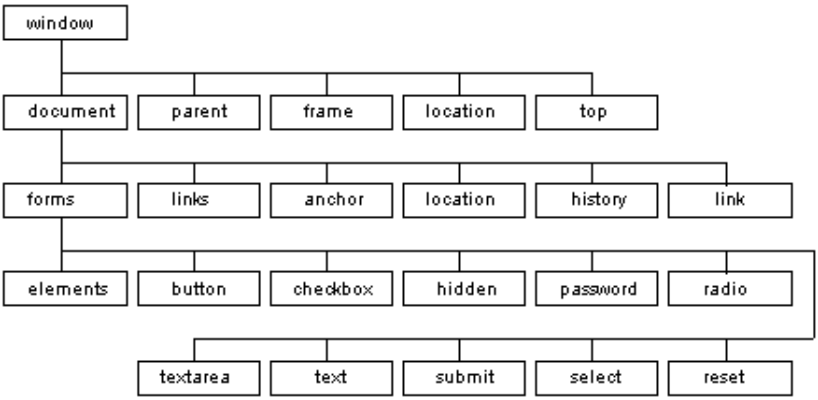


Рисунок 1 – Ієрархія об’єктів JavaScript

Розглянемо приклади, в яких використовуються різні властивості об'єктів.

Приклад 1. Перестановка зображень

Напишемо сценарій, який реалізує обмін малюнків в документі. Нехай в документі розташовано три зображення, пронумерованих від 1 до 3. У текстових полях вказуються номера малюнків, які необхідно поміняти місцями. Потрібно, щоб після натискання кнопки Обміняти зображення перемістилися на потрібні місця.

Спочатку перевіримо правильність налаштувань номера зображень, якщо це не так, то видамо повідомлення. Змінна z служить для запам'ятовування адреси першого графічного зображення. Доступ до зображення з номером r1 проводиться за допомогою конструкції document.images [r1-1]. Для того щоб на місце зображення з номером r1 розмістити зображення з номером r2, потрібно виконати оператор присвоювання:

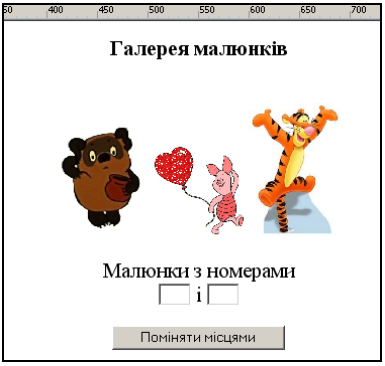
document.images [r1-1] .src = document.images [r2-1] .src

І, нарешті, на місце зображення з номером r2 міститься зображення, яке раніше було на місці з номером r1, і адреса якого

запам'ятали в змінної z: document.images [r2-1] .src = z

```
<HTML>
<HEAD>
  <TITLE> Перестановка зображень </TITLE>
  <Script>
    function chan (obj)
        {
        Var r1 = Number (obj.a1.value)
        var r2 = Number (obj.a2.value)
        if (
            (r1 <1) || (r1> 3) || (r2 <1) || (r2> 3)
            )
            alert ( "Невірно задані номери малюнків!")
        else
            {
             Var z = document.images [r1-1] .src
             document.images [r1-1] .src = document.images [r2-1] .src;
             document.images [r2-1] .src = z
            }
         }
  </Script>
</HEAD>
<BODY>
  <CENTER>
  <H4> Галерея малюнків </H4> <br>
  <IMG src = "1.gif" width = "90" name = "pic1">
  <IMG src = "2.gif" width = "90" name = "pic2">
  <IMG src = "3.jpg" width = "90" name = "pic3"> <br>
  <FORM name = form1>
     Малюнки з номерами <br>
     <Input type = "text" name = "a1" size = 1> і
     <Input type = "text" name = "a2" size = 1> <P>
     <Input type = "button" value = "Поміняти місцями" onClick = "chan (form1)">
  </FORM>
  </CENTER>
</BODY>
</HTML>
```

Результат:



Приклад 2. Просте вертикальне меню

Напишемо сценарій, який реалізує вертикальне графічне меню. При наведенні курсора миші на пункт меню змінюється колірна палітра, відповідна виділеному пункту меню.

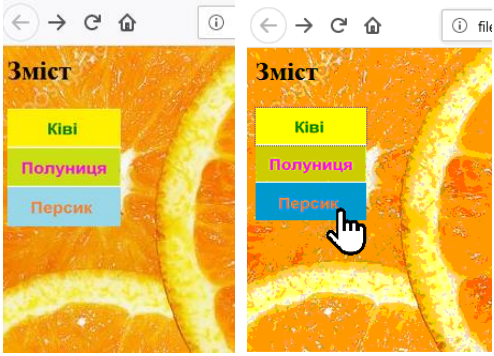
Кожному пункту меню відповідає два зображення: перше зображення, коли пункт меню не обраний, друге - при обраному пункті меню, колірна палітра малюнка змінена. Графічні зображення, відповідні ситуації, коли пункти меню не вибрані, зберігаються в файлах з іменами pch1.gif, pch2.gif. Відповідні їм графічні зображення зі зміненою палітрою зберігаються в файлах з іменами wpch1.gif, wpch2.gif.

Функція img має два параметри. Перший параметр задає вибір пункту меню, другий параметр - n - визначає номер пункту меню. Від цього параметра залежить, яке зображення в документі потрібно змінити document.images [n-1] .src (вставити на цьому місці малюнок "wpch" + n + ". Gif" або pch "+ n +". Gif). Файл формується динамічно і є конкатенацію (злиття) рядків, один з складових - значення другого параметра. Документ зі сценарієм, який реалізує вертикальне графічне меню, представлений у прикладі 2.

```
<HTML>
<HEAD>
  <TITLE>Простое вертикальное менк</TITLE>
  <script language="JavaScript">
    function img(n, action)
        {
        if (action)
            {
              document.images[n-1].src="wpch"+n+".gif"
            }
        else
            {
            document.images[n-1].src="pch"+n+".gif"
            }
      }
  </script>
</HEAD>
<BODY background="fon1.jpg">
  <H2><FONT color="#0000ff">Зміст</FONT></H2>
  <A href="tch1.html" onmouseover="img(1,1)" onmouseout="img(1,0)">
  <IMG src="pch1.gif" alt="лайм" border="0" width="103" height="35"></A><br>
  <A href="tch2.html" onmouseover="img(2,1)" onmouseout="img(2,0)">
  <IMG src="pch2.gif" alt="клубника" border="0" width="103" height="35"></A><br>
  <A href="tch3.html" onmouseover="img(3,1)" onmouseout="img(3,0)">
  <IMG src="pch3.gif" alt="персики" border="0" width="103" height="35"></A><br>
</BODY>
</HTML>
```

При попаданні курсора миші в область зображення виникає подія Mouseover, параметр обробки події onMouseOver отримує значення img (p1, true).

Результат:



Завдання

1. Перевірити приклади лабораторної роботи.
2. Написати сценарій вибору з трьох зображень одного, який вставляється нижче цих трьох.
3. Написати сценарій картинки з "ефектом наближення", тобто збільшення розмірів як реакція на потрапляння курсора миші в поле малюнка (використовувати властивості width і height).
4. Написати сценарій графічного горизонтального меню з з'являється стрілкою над пунктом, у якого знаходиться курсор.

Контрольні питання:

1. Який об'єкт є батьком всіх об'єктів JavaScript?
2. Який метод дозволяє відкрити нове вікно в сценарії JavaScript і відобразити в ньому новий документ?
3. Який метод закриває вікно документа?
4. Який метод об'єкта windows відображає діалогове вікно з текстом?
5. Які об'єкти є підлеглими до об'єкта windows?
6. Які засоби JavaScript дозволяють отримати доступ до зображень на веб-сторінці?
7. Що називається конкатенацією?

Лабораторна робота №5

Перемикачі

Дані зручно представляти за допомогою елемента управління "перемикач" (або "радіокнопка") в тому випадку, коли з декількох варіантів може бути обраний лише один.

Приклад 1. Обчислення площі фігури.

Необхідно вибрати форму фігури і визначити її площу.

Нехай для вибору фігури задана наступна форма:

```
<FORM name = "form1">
Введіть значення
<Input type = "text" name = "data" size = 10> <hr>
Вкажіть форму: <br>
<Input type = "radio" name = "fv" value = 1> квадрат <br>
<Input type = "radio" name = "fv" value = 2> Круг <br>
<Input type = "radio" name = "fv" value = 3> трикутник <hr>
<Input type = "reset" value = "Скасувати"> <hr>
Площа: <input type = "text" name = "res" size = 10>
</FORM>
```

У цій формі шість елементів. Перший елемент служить для введення рядка тексту. Наступні три елементи утворюють групу і є перемикачами. П'ятий елемент створює кнопку скидання, натискання якої скасовує всі зроблені зміни. Шостий елемент є елементом для введення рядка.

Так як об'єкт forms має властивість-масив elements, в якому містяться посилання на елементи форми в порядку їх перерахування в тезі <FORM>, то отримати доступ до першого елементу форми можна або за допомогою значення параметра name цього елемента (document.form1.data) , або використовуючи об'єктну модель JavaScript (document.forms[0].elements[0]). Другий елемент аналізованої форми можна отримати, якщо скористатися конструкцією document.forms[0]. elements[1]. Це елемент-перемикач, визначений у складі групи елементів. У розглянутому прикладі група елементів складається з трьох перемикачів. В одну групу входять елементи з однаковим значенням параметра name. Доступ до наступних елементів групи може бути здійснений так: document.forms[0].elements[2], document. forms[0].elements[3]. Обов'язковий параметр value повинен мати унікальне значення для кожного елемента групи. Користувач може вибрати тільки один варіант.

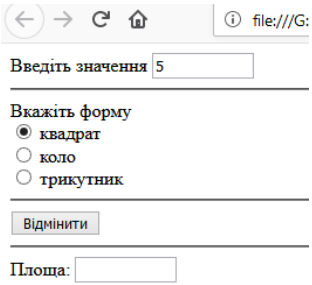
Напишемо сценарій, в якому в залежності від довжини сторони або радіусу і форми обраної фігури обчислюється її площа. Для

простоти будемо вважати, що фігура може мати або форму квадрата (задається його сторона), або форму кола (задається радіус), або форму рівностороннього трикутника (задається його сторона).

Площа розглянутих фігур обчислюється за формулою $k \cdot a^2$, де k - коефіцієнт, що залежить від форми обраної фігури; a - задане користувачем значення. Обчислення будуть простіші, якщо коефіцієнт k вказати в якості значення параметра value відповідного перемикача. Клацання на елементі "перемикач" відповідає події click, обробка якої полягає у виклику функції test. Функція має єдиний параметр, значення параметра - value перемикача, яке служить для обчислення площі фігури.

```
<HTML>
<HEAD>
  <TITLE> Дані з форми типу "перемикач". Подія Click </TITLE>
  <Script language = "JavaScript">
    function test (k)
      {
      Var a = form1.data.value
      if (a! = "")
        form1.res.value = k * Math.pow (a, 2)
      else alert ( "Введіть значення")
        }
  </Script>
</HEAD>
<BODY>
  <FORM name = "form1">
    Введіть значення
     <Input type = "text" name = "data" size = 10> <Hr>
     Вкажіть форму <br>
     <Input type = "radio" name = "fv" value = 1
    onClick = "test (form1.elements [1] .value)"> квадрат <br>
    <Input type = "radio" name = "fv" value = 3.14
     onClick = "test (form1.elements [2] .value)"> коло <br>
    <Input type = "radio" name = "fv" value = 0.42
     onClick = "test (form1.elements [3] .value)"> трикутник <hr>
    <Input type = "reset" value = "Відмінити"> <hr>
     Площа: <input type = "text" name = "res" size = 10>
  </FORM>
</BODY>
</HTML>
```

Результат:



Приклад 2. Вибір параметрів обтікання зображення текстом

Напишемо сценарій, який надає можливість користувачеві задавати значення параметрів, що визначають, до якого полю вікна (лівому або правому) притискається зображення, і, відповідно, з якого боку текст його обтікає.

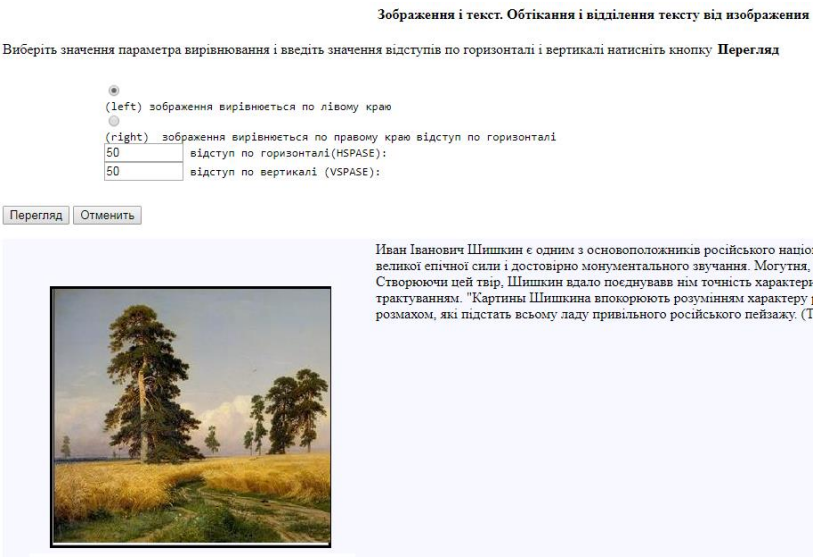
Якщо значення параметра align одно Left, то зображення притискається до лівого краю вікна перегляду браузера, а текст або інші елементи документа "обтікають" зображення з правого боку. Текст, що розміщується поруч із зображенням, може займати кілька рядків. За замовчуванням значення параметра align одно Left. При натисканні на кнопку Оновити для зображення і тексту будуть встановлені значення параметрів, прийнятих за замовчуванням.

```
<HTML>
<HEAD>
    <TITLE>Изображениеи текст. Обтекание</title>
    <script>
        function chpict(obj)
            {
            var h=obj.hsp.value
            var v=obj.vsp.value
            document.mypict.hspace=h
            document.mypict.vspace=v
            if ((obj.elements[0]).checked)
                document.mypict.align="Left"
            else
                document.mypict.align="Right"
            }
        function rset(obj)
            {
            document.mypict.align="Left"
            document.mypict.hspace=0
            document.mypict.vspace=0
            obj.hsp.value=0
            obj.vsp.value=0
            }
    </script>
</HEAD>
```

```
<BODY>
    <CENTER>
    <H4>Зображення і текст. Обтікання і відділення тексту від изображения</H4>
    </CENTER>
    <FORM name="form1">
        Виберіть значення параметра вирівнювання  і введіть значення відступів по
        горизонталі і вертикалі  натисніть кнопку <B>Просмотр</B>.<br>
        <PRE>
        <input type="radio" name="alg" checked value=ld>
        (left) зображення вирівнюється по лівому краю
        <input  type="radio"  name="alg"  value=rd>
        (right)  зображення вирівнюється по правому краю відступ по горизонталі
        <input type="text" name="hsp" size="8" value="0"> відступ по горизонталі(HSPASE):
        <input type="text" name="vsp" size="8" value="0"> відступ по вертикалі (VSPASE):
        </PRE>
        <input type="button" value="Перегляд" onclick="chpict(form1)">
        <input type="reset"value="Отменить" onclick="rset(form1)">
    </FORM>
    <TABLE bgcolor="F8F8FF">
        <TR>
            <TD>Иван Іванович Шишкин є одним з основоположників  російського
                національного пейзажу.
                <IMG src="pl.jpg" name="mypict" align="left" border="3" height="310">
                У полотні "Рожь" Шишкин створив образ великої епічної сили  і достовірно
                монументального звучання. Могутня, повна  богатирських сил природа, багат
                привільний край.  Створюючи цей твір, Шишкин вдало поєднував ним  точніс
                характеристики  форм  природи  з  широкою, узагальненою  їх трактуванням.
                "Картины Шишкина впокоряють розумінням  характеру рідної природи, своєю
                спокійною епічною силою  і широким розмахом, які підстать всьому ладу
                привільного російського пейзажу. (Т. Юрова)
            </TD>
        </TR>
    </TABLE>
</BODY>
</HTML>
```

Якщо зображення розглядається як елемент рядка, то значення параметрів вирівнювання задають розташування зображення щодо рядка тексту. Верхня межа зображення може бути вирівняна або за найвищим текстового елементу поточного рядка, або за найвищим елементу в рядку (наприклад, іншого зображенню). Базовою вважається нижня частина лінії тексту, яка проводиться без урахування нижній частині деяких символів. Середину зображення можна вирівняти або за базовою лінії, або по середині поточного рядка. Нижню частину зображення можна вирівняти по базовій лінії, або по нижній межі поточного рядка.

Результат:



Завдання

1. Перевірити приклади з лабораторної роботи.
2. Напишіть сценарій, який дозволяє продемонструвати зміни розмірів і положення на сторінці горизонтальної лінії.
3. Розробіть анкету, визначальну стать, вік, сімейний стан і т.п., людини.

Контрольні питання:

1. Яке призначення має перемикач радіокнопка?
2. Як створити радіокнопку у формі?
3. Як отримати доступ до певного елементу форми у сценарії?
4. Що означає конструкція document.forms[0].elements[1]?
5. Для чого використовується параметр value форми?

Лабораторна робота №6

Прапорці

Елемент управління "прапорець" використовується в разі, коли із запропонованих варіантів можна вибрати як один, так і кілька. Кожен варіант вибору задається прапорцем, який можна або встановити, або скинути. Прапорець визначається в тезі <input> значенням checkbox параметра type. Обов'язковою параметром є

параметр value, значення якого буде передано на обробку в разі вибору натисненням кнопки.

Приклад 1. Вибір характеристик видання

Припустимо, читачеві пропонується заповнити анкету, в якій потрібно вказати назву улюбленого видання і вибрати із запропонованого списку характеристики, які притаманні розглядався виданню.

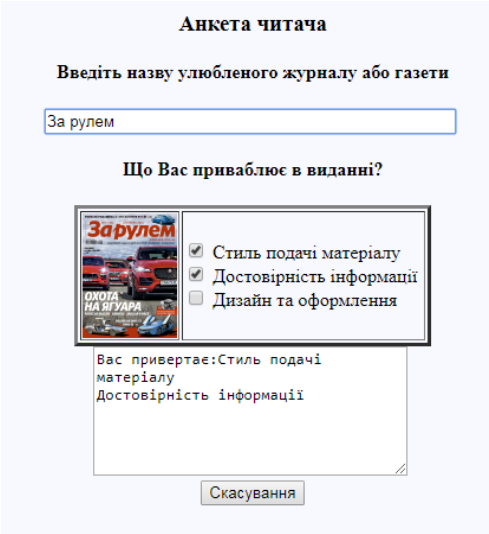
Для завдання параметрів видання можна скористатися прапорцем. Користувач встановлює прапорці для тих властивостей, якими, на його думку, має видання. Обробка анкети буде полягати в тому, що обрані властивості будуть відображені в поле введення багаторядкового тексту.

При натисканні мишею на прапорці виникає подія click, обробка якої полягає у виклику функції set з одним параметром, що приймають значення параметра value прапорця. Для формування рядка результату служить глобальна змінна s; до наявного значення додається значення параметра функції і поміщається в текстове поле. Якщо натиснути на кнопку Скасування, то очистяться все поля форми. Однак слід подбати про те, щоб значення змінної s змінилося на початкове. Значення параметра реакції на подію click при натисканні на кнопці Скасування задається оператором присвоювання, що забезпечує початкові умови.

```
<HTML>
<HEAD>
 <TITLE> Анкета читача </TITLE>
 <Script>
   var s = "Вас привертає:"
   function set (vch)
       {
       s=s+vch+"\r\n"; document.form1.area.value=s
       }
 </Script>
</HEAD>
<BODY bgcolor = "F8F8FF">
 <CENTER>
 <H3 align = "center"> Анкета читача </H3>
 <FORM name = "form0">
   <H4> Введіть назву улюбленого журналу або газети </H4>
   <Input type = "text" name = "n" size = 45> <br>
 </FORM>
```

```
<FORM name = "form1">
  <H4> Що Вас приваблює в виданні? </H4>
  <TABLE border = 3 align = center>
      <TR>
          <TD>
              <img src = "1.jpg" align = "center" width=80>
          </TD>
          <TD> <input type = "checkbox" name = "m1"
              value = "Стиль подачі матеріалу"
              onClick = "set (form1.elements[0].value)">
              Стиль подачі матеріалу <br>
              <Input type = "checkbox" name = "m2"
              value = "Достовірність інформації"
              onClick = "set (form1.elements[1].value)">
              Достовірність інформації <br>
              <Input type = "checkbox" name = "m3"
              value = "Дизайн та оформлення"
              onClick = "set (form1.elements[2].value)">
              Дизайн та оформлення <br>
          </TD>
      </TR>
  </TABLE>
  <Textarea name = "area" cols = 35 rows = 7> </textarea> <br>
  <Input type="reset" value="Скасування" onclick="s ='Вас привертає:">
 </FORM>
</BODY>
</HTML>
```

Результат:



У розглянутих прикладах значення параметра name прапорців були різні, оскільки кожен прапорець існував незалежно від інших. Прапорці можна об'єднати в групу. Для цього слід всім прапорців привласнити одне і те ж значення параметра name.

Приклад 2. Використання прапорців в анкеті перекладача

В анкеті потрібно вказати ті мови, якими володіє перекладач. Припустимо, що за знання кожної мови призначається певна сума. Розмір винагороди визначається після заповнення анкети в залежності від тих мов, якими користувач володіє. За результатами заповненої перекладачем анкети напишіть сценарій визначення розміру винагороди.

Для завдання відомостей про те, чи володіє користувач певною мовою, зручно застосовувати прапорець. При натисканні мишею по кнопці Винагорода виконується функція grant(). Потрібно проаналізувати стан прапорців. Властивість checked повертає логічне значення, що представляє поточне значення окремого прапорця (true або false). Скористаємося тим, що кожен об'єкт form має властивість-масив elements, отримаємо доступ до кожного прапорця форми. Стан першого прапорця можна визначити за допомогою наступної конструкції:

(Document.forml.elements[0]). Checked

другого -

(Document. Form1.elements[1]). Checked

і т. д. В змінної до накопичується сума. Крок збільшення цієї змінної задається в якості значення параметра value. Після аналізу всіх прапорців отримана сума виводиться в документ.

```
<HTML>
<HEAD>
<TITLE> Дані, представлені прапорцем. Анкета перекладача
</TITLE>
<Script language = "JavaScript">
<! - //
function grant ()
{Var d = document
var k = 0;
if ((d.form1.elements [0]). checked)
k = k + Number (d.form1.elements [0] .value)
if ((d.form1.elements [1]). checked)
k = k + Number (d.form1.elements [1] .value)
if ((d.form1.elements [2]). checked)
k = k + Number (d.form1.elements [2] .value)
form1.ww.value = "Вам належить винагорода" + k + "у.е."
}
```

```
//->
</Script>
</HEAD>
<BODY>
<H3> Анкета для перекладачів </H3>
Вкажіть ті мови, якими Ви володієте досконало: <br>
<FORM name = "form1">
<Input type = "checkbox" name = "lan" value = 100> русскій <br>
<Input type = "checkbox" name = "lan" value = 200> англійська
<br>
<Input type = "checkbox" name = "lan" value = 300> французький
<br>
<Input type = "button" value = винагороди onClick = "grant ()">
<hr>
<Input type = "Text" size = 50 name = "ww" value = ""> <br>
<Input type = "reset" value = "Скасувати">
</FORM> <hr>
</BODY>
</HTML>
```

Завдання

Напишіть сценарій обробки анкети слухача курсів. Користувач може вибрати курс, його тривалість, мову, на якому він готовий працювати з викладачем, і форму звітності. Залежно від цих параметрів визначається вартість окремого курсу і вартість всього навчання.

Контрольні питання:

1. В яких випадках використовуються елементи управління "прапореці"?
2. Як визначити елемент «прапорець» у сценарії?
3. Яким чином сценарій обробляє натиск мишею на «прапорці»?
4. Як очистити поля форми за допомогою кнопки?
5. Як об’єднати прапорці у групу?
6. Як проаналізувати стан «прапорців»?

Лабораторна робота №7

Списки

Якщо елементів багато, то подання їх за допомогою прапорців або перемикачів збільшує розмір форми. В цьому випадку варіанти

вибору можуть бути представлені у вікні браузера більш компактно за допомогою тега <select>. Тег має кілька параметрів. Параметр name є обов'язковим. Для того щоб встановити число одночасно видимих елементів, слід задати параметр size = n. Коли n дорівнює 1, то відображається спадаюче меню або список вибору; при n > 1 виводиться список з n одночасно видимими значеннями. Якщо параметр size не заданий, то за замовчуванням приймається значення рівне 1. Вказівка параметра multiple означає, що з меню або списку можна вибрати кілька елементів. Елементи меню задаються всередині тега <select> за допомогою тега <option>. Загальний вигляд тега такий:

<Option selected value = рядок>

Параметр selected означає, що даний елемент списку вважається обраним за замовчуванням. Параметр value містить значення, яке передається, якщо даний елемент обраний зі списку або меню.

Приклад 1. Обробка анкети перекладача

Напишемо сценарій обробки анкети перекладача. Відомості про тих мовах, якими володіє перекладач, потрібно задати за допомогою списку, Вибрані мови слід поміщати в поле введення багаторядкового тексту.

Нагадаємо, що подія change відбувається в той момент, коли значення елемента форми text, select або textarea змінилося, і елемент втратив фокус. Будемо обробляти анкету перекладача наступним чином. Параметр обробки події помістимо в тег <select>. Як тільки обрано конкретну мову, тобто відбулася подія change, виконується функція gr:

```
function gr (obj, m)
 {Var r = 100 * (Number (((obj.elements [0]) [m]). Value) +1)
 s + = ((obj.elements [0]) [m]) .text + "\ r \ n"
 obj.restext.value = s
 sum + = r
 obj.res.value = r}
```

Перший параметр - ім'я форми, другий - значення параметра value вибраного пункту. Другий оператор забезпечує формування рядки всіх обраних користувачем мов. Третій оператор поміщає обчислене значення в текстове поле. В результаті виконання четвертого оператора присвоювання в змінної sum формується значення, яке, потім при натисканні кнопки Сума буде виведено в текстове поле. Останній оператор поміщає значення для вибраної мови в відповідне поле форми.

```
<HTML>
<HEAD>
```

```
<TITLE> Реакція на подію Change в тезі select </TITLE>
<Script language = "JavaScript">
<! - //
var s = ""
var sum = 0
function gr (obj, m)
{Var r = 100 * (Number (((obj.elements [0]) [m]). Value) +1)
s + = ((obj.elements [0]) [m]). text + "\ r \ n"
obj.restext.value = s
sum + = r
obj.res.value = r
}
//->
</Script>
</HEAD>
<BODY>
<FORM name = "form1">
<H3> Анкета перекладача </H3>
<TABLE border = 3 bgcolor = silver>
<TR> <TH> Вибрана мова </TH> <TH> Результат </TH>
</TR>
<TR>
<TD valign = top>
<Select name = "data" size = 3 onChange = "gr (form1,
form1.data.value)">
<Option value = 0> російська
<Option value = 1> англійська
<Option value = 2> французький
</Select> <P>
<Input type = "text" name = "res" size = 15>
</TD>
<TD> <TEXTAREA name = "restext" cols = 15 rows = 6>
</TEXTAREA> <P>
<Input type = "button" value = Сума onClick = "form1.resgr.value
= sum">
<Input type = "text" name = "resgr" size = 10>
</TD> </TR> </TABLE> <p>
<Input type = "reset" value = "Скасувати" onClick = "sum = 0; s =
"">
</FORM>
</BODY>
```

```
</HTML>
```

Приклад 2. Тест "Міста і пам'ятники"

Напишемо сценарій обробки тесту "Міста і пам'ятники". Назви міст і пам'ятників задаються за допомогою списків. Користувач вибирає в лівому переліку назву міста, а в правому - пам'ятник, розташований в цьому місті. Після натискання кнопки Результат в текстове поле виводиться кількість правильних відповідей.

У сценарії використовуються три глобальні змінні. Змінна q зберігає останнім вибране значення в лівій колонці; змінна a - вибране значення правого стовпчика; значення змінної sum містить число правильних відповідей. У двох списках для правильної пари "питання /відповідь" збігаються відповідні значення параметра value. Ці значення перевіряються після вибору елемента списку правого стовпчика. Результат тестування можна подивитися, якщо натиснути кнопку Результат.

```
<HTML>
<HEAD>
<TITLE> Міста і пам'ятники. </TITLE>
<Script>
<! -
var sum = 0
var q
var a
function eq ()
{Q = test.question.value
a = test.answer.value;
if (a == q) sum + = 1
}
function result ()
{Document.test.res.value = sum}
//->
</Script>
<BODY background = "fon3.gif">
<H3 align = center> Міста і пам'ятники </hз>
<FORM name = "test">
<TABLE border = 3 align = center cellpadding = 5
cellspacing = 6 bgcolor = silver>
<TR> <TH> Пам'ятник </TH> <TH> Знаходиться в місті </TH>
<TR> <TD>
<Select size = 7 name = "question"
onChange = "q = test.question.value">
```

```
<Option value = "mad"> Музей Прадо <br>
<Option value = "ber"> Рейхстаг <br>
<Option value = "mil"> Оперний театр Ла Скала <br>
<Option value = "ier"> Стіна Плачу <br>
<Option value = "mek"> Священний камінь Кааб <br>
<Option value = "spb"> Мідний Вершник <br>
<Option value = "mos"> Третьяковська галерея <br>
<Option value = "par"> Тріумфальна Арка <br>
<Option value = "new"> Статуя Свободи <br>
<Option value = "lon"> Тауер <br>
</Select>
</TD>
<TD>
<Select size = 7 name = "answer" onChange = "eq ()">
<Option value = "spb"> Санкт-петербург <br>
<Option value = "mos"> Москва <br>
<Option value = "mek"> Мекка <br>
<Option value = "ier"> Єрусалім <br>
<Option value = "mil"> Мілан <br>
<Option value = "par"> Паріж <br>
<Option value = "mad"> Мадрид <br>
<Option value = "lon"> Лондон <br>
<Option value = "new"> Нью-Йорк <br>
<Option value = "ber"> Берлін <br>
</Select>
</TD> </TR>
</TABLE>
<CENTER> <P>
<Input type = "button" value = "Результат" onclick = "result ()">
<br>
Кількість правильних відповідей
<Input type = "text" name = "res" size = "5"> <P>
<Input type = "reset" value = "Оновити" onclick = "sum = 0">
</FORM>
</BODY>
</HTML>
```

Завдання

1. Перевірити приклади з лабораторної роботи.

2. Напишіть сценарій, який дозволяє вибрати для таблиці і складових її осередків або колір фону, або фонове зображення, або і те

й інше. Передбачте можливість завдання свого кольору фону для кожного осередку. сс

3. Напишіть сценарій, який дозволяє порахувати вартість передбачуваної покупки. Здається список продуктів, ціна за одиницю товару та кількість примірників.

Контрольні питання:

1. Для чого призначений тег <select>?
2. В яких випадках на формах застосовується еоемент управління список?
3. Який параметр є обов'язковим для тегу <select>?
4. Як установити число одночасно видимих елементів у списку?
5. В яких випадках на формі відображається спадаюче меню?
6. Яку функцію виконує тег <option>?
7. Як забезпечити одночасне виділення декількох елементів спмску?
8. Що означає параметр selected?

Лабораторна робота №8

Фрейми

Вікно перегляду браузера можна розбити на кілька прямокутних областей, так званих фреймів. Області стикаються одна з однлю, в кожну з областей можна завантажити окремий HTML-документ і працювати з ним незалежно від документів, завантажених в інші області вікна або фрейму. Між фреймами можна організувати взаємодію, наприклад, вибір посилання в одному з фреймів дозволить змінити вміст інших кадрів. Фрейми часто використовуються у випадках, коли виникає необхідність завантажити документ в одну з областей при роботі в іншій області, або коли слід відобразити інформацію, яка повинна постійно перебувати на екрані.

Приклад 1. Проста фреймова структура

Створимо документ, який розбиває область екрану на дві частини. Ліва частина містить зміст розділів документа, який розташовується в правій частині. При виборі пункту змісту в лівій частині з'являється відповідний розділ документа в правій частині.

Розіб'ємо область екрану на два фрейми. Лівий фрейм займає 25% ширини всього вікна і буде містити зміст розділів документа, який завантажимо в правий фрейм. Нехай ім'я файлу, що містить зміст - contents.htm, а ім'я документа - ch.htm. Фреймова структура задає спосіб організації екрана і визначає, які документи повинні бути

спочатку завантажені у фрейми.

```
<HTML>
<HEAD>
<TITLE> Проста фреймова структура </TITLE>
</HEAD>
<Frameset cols = "25%, 75%">
<Frame src = contents0.htm name = left>
<Frame src = ch.htm name = right>
</Frameset>
</HTML>
```

Параметр cols тега <frameset> має вигляд cols = "список значень". У списку через кому перераховуються значення, які визначають розміри фреймів. Список повинен містити не менше двох значень. Значення можуть задаватися в відсотках, у точках, в відносних одиницях.

Тег <frame> визначає один фрейм. Він повинен розташовуватися усередині парного тега <frameset> і </frameset>. Число тегів <frame> має збігатися з кількістю тегів, визначених при описі фреймової структури. У розглянутому прикладі в тезі <frameset cols = "25%, 75%"> визначено два фрейми, тому в подальшому слід опис кожного з фреймів за допомогою тега <frame>.

Значення параметра src тега <frame> визначає адресу документа, який спочатку завантажується у фрейм. В даному випадку в лівий фрейм завантажується документ з ім'ям contents0.htm, а в правий фрейм - документ з ім'ям ch.htm. У тезі параметр name визначає ім'я фрейма, необхідне для вказівки, в якій фрейм завантажити документ. Якщо ім'я фрейма не ставити, то буде створений фрейм без імені, але послатися на нього з інших фреймів буде не можна.

Приклад 2. Фреймова структура з завантажуються документами

Створимо документ, ліва частина якого представляє зміст, а в праву частину завантажуються документи обраного пункту змісту. Документи, що відповідають пунктам змісту, зберігаються в різних файлах.

При вирішенні завдання екран і раніше розбивається на два фрейми. Лівий фрейм займає 30% ширини всього вікна і буде містити зміст документів, які можуть бути переглянуті користувачем при виборі відповідного пункту. Правий фрейм займає більшу частину вікна перегляду і призначений для відображення самих документів. При первинному завантаженні обидва фрейма ділять вікно перегляду по вертикалі в співвідношенні 30% і 70%. Дане співвідношення може змінюватися при перегляді. Кожен з фреймів має свою смугу

прокрутки, що забезпечує перегляд всього документа. При виборі посилання в лівому фреймі відповідний документ буде завантажений в правий фрейм. Така структура дозволяє одночасно бачити на екрані і зміст документів, і самі документи.

Нехай зміст документа містить шість пунктів і розташовується в файлі з ім'ям contents.htm. Потрібно, щоб файл, який містить зміст, завантажувався в лівий фрейм. Файли з іменами ch1.htm, ch2.htm, ..., ch6.htm містять документи, що відповідають пунктам змісту.

Фреймова структура мало відрізняється від тієї, яка була розглянута в попередньому прикладі (Приклад 2, а).

Приклад 2, а. Завдання фреймової структури

```
<HTML>
<HEAD>
<TITLE> Проста фреймова структура </TITLE>
</HEAD> <frameset cols = "30%, 70%">
<Frame src = contents.htm name = left>
<Frame src = empty.htm name = right>
</Frameset>
</HTML>
```

У правий фрейм спочатку завантажується файл з ім'ям empty.htm. Якщо відразу невідомо, який файл завантажувати у фрейм, то можна використовувати файл, що містить HTML-код (Приклад 2, б).

Приклад 2, б. Документ для первинного завантаження

```
<HTML>
<HEAD>
<TITLE> Порожній документ </TITLE>
</HEAD>
<BODY>
</BODY>
</HTML>
```

В лівий фрейм поміщається Зміст, яке містить посилання на документи, розташовані в різних файлах. Зміст може бути сформовано так, як зазначено в Прімері 2, ст.

Приклад 2, ст. Зміст, завантажувати в лівий фрейм

```
<HTML>
<HEAD>
<TITLE> Зміст </TITLE>
</HEAD>
<BODY background = "decor.gif" bgcolor = silver>
<Base target = right>
```

```
<H3> Зміст </h3>
<OL>
<LI> <A href="ch1.htm"> Основи мови HTML </A>
<LI> <A href="ch2.htm"> Графіка </A>
<LI> <A href="ch3.htm"> Ізображеніе-карта </A>
<LI> <A href="ch4.htm"> Списки </A>
<LI> <A href="ch5.htm"> Таблиці </A>
<LI> <A href="ch6.htm"> Фрейми </A>
</OL>
</BODY>
</HTML>
```

Приклад 3. Обмін вмістом фреймів

Створимо документ, який розбиває вікно перегляду за допомогою фреймів на три прямокутні області. У верхній частині перелічено два фрейми, нижня область складається з одного фрейма. У нижній області розташовується кнопка, яка обмінює вміст верхніх фреймів.

Область вікна перегляду спочатку розбивається на два фрейми по горизонталі. Параметр rows = "список значень" задає кількість і розміри фреймів по горизонталі. При описі тега <frameset rows = "*, 50"> визначається, що розмір нижнього фрейма 50 пікселів, інший простір відводиться під верхній фрейм. Верхній фрейм в свою чергу ділиться по вертикалі на два фрейми, розміри яких задані у відсотках до області перегляду. HTML-код документа, що створює описану фреймової структуру, виглядає так, як зазначено в Прикладі 3, а.

Приклад 3, а. Три фрейми з кнопкою для обміну вмісту фреймів

```
<HTML>
<HEAD>
<TITLE> Три фрейму з кнопкою для обміну </TITLE>
</HEAD>
<Frameset rows = "*, 50">
<Frameset cols = "55%, 45%">
<Frame src = lpict.htm name = left>
<Frame src = rtext.htm name = right>
</Frameset>
<Frame src = but1.htm name = butt>
</Frameset>
</HTML>
```

Для посилання на фрейми будемо, як і раніше, використовувати імена. Спочатку слід запам'ятати ім'я файлу, завантаженого в лівий фрейм. Для цього використовується змінна l. Потім в лівий фрейм

завантажується документ, розташований в правому фреймі. Це досягається виконанням оператора присвоювання

top.left.location = top.right.location

І, нарешті, в правий фрейм завантажується той документ, адреса якого запам'ятали в змінної l.

Приклад 3, б. Сценарій для нижнього фрейма

```
<HTML>
<HEAD>
<TITLE> Кнопка для зміни вмісту фреймів </TITLE>
<Script>
<! - //
function chframe ()
{Var l = top.left.location
top.left.location = top.right.location
top.right.location = l
}
//->
</Script>
</HEAD>
<BODY background = "decor.gif" bgcolor = silver>
<CENTER>
<FORM name = form1>
<Input type = "button" value = Обмін onclick = "chframe ()">
</FORM>
</CENTER>
</BODY>
</HTML>
```

При натисканні на кнопку Обмін вміст верхніх фреймів поміняється місцями.

Функція, що здійснює обмін вмістом фреймів, може бути описана наступним чином:

```
function chframe ()
{Var l = top.frames [0] .location
top.frames [0] .location = top.frames [1] .location
top.frames [1] .location = l
}
```

Браузер Microsoft Internet Explorer дозволяє використання так званих плаваючих фреймів, опис яких може бути розташований в тексті звичайного HTML-документа. Для визначення плаваючого фрейма використовується тег <iframe>. У зазначеному тезі можуть зустрічатися ті ж параметри, що і в тезі опису звичайних фреймів, крім

параметра noresize, оскільки розмір плаваючого фрейма не може бути змінений користувачем. Браузери, які не підтримують тег <iframe>, відображають записану між тегами <iframe> і </iframe> інформацію.

Для плаваючих фреймів за допомогою параметрів можна задавати розміри фрейма, горизонтальне вирівнювання, розмір відступу вмісту фрейма від кордонів. У наступному прикладі наведено опис плаваючого фрейма і задані параметри. Фрейм має висоту 320 пікселів, займає по ширині 60% вікна, розташований праворуч від тексту, смуги прокрутки будуть встановлені в разі, якщо документ не стане поміщатися у фрейм. Вміст фрейму належить відокремлювати від кордону по горизонталі і вертикалі на задане число пікселів.

<Iframe src = pictsh.htm name = "flframe" height = 320 width = 60% hspace = 50 vspace = 10 scrolling = auto align = right>

Завдання

1. Створіть документ, який розбиває вікно перегляду за допомогою фреймів на дві прямокутні області: верхню і нижню. У верхній області помістіть зміст у вигляді списку, при виборі пунктів якого відповідний розділ повинен з'являтися в нижній частині вікна.

2. Створіть документ, який розбиває вікно перегляду за допомогою фреймів на дві прямокутні області: ліву і праву. У лівій області помістіть зміст, при виборі пунктів якого відповідає пункту розділ повинен з'являтися в правій частині вікна. Зміст подайте за допомогою графічного вертикального меню.

Контрольні питання:

1. Для чого використовують фрейми на веб-сторінках?
2. Якими тегами створюється фреймова структура?
3. Як вивести потрібний документ у фреймі?
4. Як установити зв’язок між документами, розмішеними в окремих фреймах?
5. Як вказати заданий фрейм для відкриття документу за посиланням з іншого документу?
6. Яке призначення плаваючих фреймів? Яким тегом створюється поаваючий фрейм?
7. Як установити заданий розмір фрейма?
8. За допомогою якого параметра можна керувати прокруткою вміста фрейма?
9. Яким чином можна організувати обмін умістом фреймів?

Лабораторна робота №9

Повторювані обчислення - цикли

Для успішного вирішення широкого кола завдань потрібно багато разів повторити деяку послідовність дій, записану в програмі один раз. У тому випадку, коли число повторень послідовності дій нам невідомо, або число повторень залежить від деяких умов, можна скористатися оператором циклу виду:

while (B) {s}

де B - вираз логічного типу; s - оператори, звані тілом циклу. Оператори s в фігурних дужках виконуються до тих пір, поки умова B не стане хибним.

Приклад 1. Знаходження загального дільника

Напишемо програму, яка для двох заданих чисел визначає найбільший спільний дільник.

При вирішенні задачі скористаємося алгоритмом Евкліда. Якщо значення m дорівнює нулю, то найбільший спільний дільник чисел n і m дорівнює n:

НСД (n, 0) = n.

В інших випадках вірно наступне співвідношення:

НСД (n, m) = НСД (m, n% m).

У функції nod змінна p використовується для отримання залишку від ділення чисел n і m. Виконання циклу триває до тих пір, поки значення p не стане рівним нулю. Останнє обчислене значення m одно найбільшою загальною делителю.

```
<HTML>
<HEAD>
<TITLE> Найбільший спільний дільник двох чисел </TITLE>
<Script language = "JavaScript">
<! - //
function nod (obj)
{Var n = obj.num1.value
var m = obj.num2.value
var p = n% m
while (p! = 0)
{N = m
m = p
p = n% m
}
obj.res.value = m
}
```

```
//->
</Script>
</HEAD>
<BODY>
Найбільший спільний дільник двох заданих чисел
<FORM name = "form1">
Введіть число <input type = "text" name = "num1" size = "8">
<br>
Введіть число <input type = "text" name = "num2" size = "8">
<br>
<Input type = "button" value = "Обчислити" onClick = "nod
(form1)"> <br>
Найбільший спільний дільник <input type = "text" name = "res"
size = "8"> <hr>
<Input type = "reset" value = "Скасувати">
</FORM>
</BODY>
</HTML>
```

Якщо число повторень заздалегідь відомо, то можна скористатися наступним оператором циклу, який часто називають оператором циклу лічильного типу. Синтаксис цього оператора такий:

for (A; B; I) {S}

Вираз A служить для ініціалізації параметра циклу і обчислюється один раз на початку виконання циклу. Вираз B (умова продовження) управляє роботою циклу. Якщо значення виразу помилково, то виконання циклу завершується, якщо істинно, то виконується оператор S, що становить тіло циклу. Вираз I служить для зміни значення параметра циклу. Після виконання тіла циклу S обчислюється значення виразу I, потім знову обчислюється значення виразу B і т.д. Цикл може припинити свою роботу в результаті виконання оператора break в тілі циклу.

Приклад 2. Досконалі числа

Напишемо програму, що визначає, чи є задане число n досконалим.

Досконалим називається число n, яка дорівнює загальній кількості своїх подільників, крім самого числа. Наприклад, число 6 є досконалим, т. К. Вірно 6 = 1 + 2 + 3, де 1, 2, 3 - подільники числа 6. Число 28 також є досконалим, справедливо рівність 28 = 1 + 2 + 4 + 7 + 14.

```
<HTML>
<HEAD>
```

```
<TITLE> Ітераційні методи. Досконалі числа </TITLE>
<Script language = "JavaScript">
<! - //
function sumdel (n)
{Var s = 1;
for (var i = 2; i <= n /2; i ++)
{If (n% i == 0) s + = i}
 return s
}
function sov (obj)
{Var n = obj.numb.value;
var s = ""
if (n == sumdel (n)) s = "досконале"
else s = "не є досконалим"
return s
}
//->
</Script>
</HEAD>
<BODY>
<P> Ітераційні методи. Досконалі числа </P>
<FORM name = "form0">
Введіть натуральне число: <input type = "text" size = 8 name = "numb">
<Input type = "button" value = Виконати onClick = "this.form.res.value = sov (form0)"> <hr>
Дане число: <input type = "text" size = 24 name = "res"> <hr>
<Input type = "reset" value = Скасувати>
</FORM>
</BODY>
</HTML>
```

Зверніть увагу на значення параметра обробки події. В даному випадку це оператор присвоювання, в правій частині якого виклик функції sov.

Оператор for ... in використовується для аналізу властивостей об'єкта. Синтаксис оператора:

for (i in t) {s}

де i - змінна циклу; t - об'єкт; s - послідовність операторів.

В результаті виконання оператора циклу проводиться перебір властивостей об'єкта. Змінна циклу при кожному повторенні містить значення властивості об'єкта. Кількість повторень тіла циклу s

дорівнює числу властивостей, визначених для об'єкта t.

Приклад 3. Визначення властивостей елемента форми

Напишемо сценарій, за допомогою якого можна визначити властивості елемента форми "поле введення багаторядкового тексту".

Властивості об'єкта за допомогою оператора циклу формуються в рядку result, потім після перегляду всіх властивостей значення рядка result поміщається в поле введення багаторядкового тексту.

```
<HTML>
<HEAD>
<TITLE> Операції над об'єктами. Властивості текстового поля </TITLE>
<Script language = "JavaScript">
<! - //
function propobj (obj)
{Var result = ""
for (var i in obj)
{Result + = obj.data.value + "." + I + "=" + (obj.data) [i] + "\ r \ n"}
result + = "\ n \ r"
form1.data.value = result
}
//->
</Script>
</HEAD>
<BODY bgcolor = F8F8FF>
<CENTER>
<H4> Визначення властивостей об'єктів </H4>
<FORM name = "form1">
<Input type = "button" value = Виконати onClick = "propobj (form1)"> <hr>
<Textarea name = "data" cols = 30 rows = 10 id = 1> Текст </textarea> <hr>
<Input type = "reset" value = Очистити>
</CENTER>
</FORM>
</BODY>
</HTML>
```

Завдання

1. Перевірити приклади з лабораторної роботи.
2. Напишіть програму, яка "перевертає" задане натуральне число.

3. Напишіть сценарій, в якому визначається кількість "щасливих" шестизначних автобусних квитків, тобто Таких, в номерах яких сума перших трьох цифр дорівнює сумі трьох останніх.

4. Напишіть програму, яка визначить всі подільники заданого натурального числа.

Контрольні питання:

1. Що називають циклічним алгоритмом?
2. Які відрізняють цикли у програмуванні?
3. Які оператори JavaScript служать для програмування циклічних алгоритмів?
4. Як працює лічильний цикл?
5. Як працює цикл з передумовою?
6. Яким чином визначити властивості об’єкта за допомогою циклічного алгоритму?

Лабораторна робота №10

Обробка і представлення дат

Вбудований об'єкт Data застосовується для подання та обробки дати і часу. Він не має властивостей, але володіє декількома методами, що дозволяють встановлювати і змінювати дату і час. У мові JavaScript дата визначається числом мілісекунд, що пройшли з 1 січня 1970 року.

Об'єкт Data створюється оператором new за допомогою конструктора Data. Якщо в конструкторі відсутні параметри, то значенням new Data () буде поточна дата і час. Значенням змінної my_data1, визначеної в такий спосіб:

var my_data1 = new Data ()

буде об'єкт, відповідний поточної дату і час.

Параметром конструктора new Data може бути рядок формату "місяць, день, рік години: хвилини: секунди". Наведемо змінну my_data2 і дамо їй початкове значення:

var my_data2 = new DataC'Fv, 12, 1978 16:45:10 ")

Змінна my_data2 визначає дату 12 лютого 1978 року і час 16 годин 45 хвилин і 10 секунд. Значення годин, хвилин, секунд можна опустити, в цьому випадку вони будуть дорівнюють нулю:

var my_data3 = new Data ("Feb, 12, 1978")

Параметри конструктора new Data можуть визначати рік, місяць, число, час, хвилини, секунди за допомогою чисел. Дату 12 лютого 1978 року і час 16 годин 45 хвилин і 10 секунд можна задати так:

var my_data4 = new Data (78, 1, 12, 16, 45, 10)

Якщо час опустити, то опис буде наступним:

var my_data5 = new Data (78, 1, 12)

Всі числові представлення дати нумеруються з нуля, крім номера дня в місяці. Місяці представляються числами від 0 (січень) до 11 (грудень), тому другий параметр при завданні змінних my_data4 і my_data5 дорівнює 1.

Методами об'єкта Date можна отримувати і встановлювати окремо значення місяця, дня тижня, годин, хвилин і ін.

-Метод getDate повертає число в діапазоні від 1 до 31, що представляє число місяця.

-Метод getHours повертає годину доби. Значення повертається в 24-годинному форматі від 0 (опівночі) до 23.

-Метод getMinutes повертає хвилини як ціле від 0 до 59.

-Метод getseconds повертає число секунд як ціле від 0 до 59.

-Метод getDay повертає день тижня як ціле число від 0 (неділя) до 6 (субота).

-Метод getMonth повертає номер місяця в році як ціле число в інтервалі між 0 (січень) і 11 (грудень). Зверніть увагу, що номер місяця не відповідає стандартному способу нумерації місяців.

-Метод getYear видає рік об'єкта.

У наступному прикладі ці методи використовуються для формування поточного часу.

Приклад 1. Визначення поточного часу

Напишемо сценарій, який визначає поточний час і виводить його в текстове поле в форматі "чч: мм: сс".

В змінної res формується рядок, яка потім буде відображена в поле rest форми з ім'ям form1. Для того щоб уточнити час, слід ще раз натиснути кнопку Час і т. Д. Повністю сценарій приведений в прикладі 1.

```
<HTML>
<HEAD>
<TITLE> Визначення часу </TITLE>
<Script language = "JavaScript">
<! - //
function c1 ()
{Var d = document
var t = new Date ()
var h = t.getHours ()
var m = t.getMinutes ()
var s = t.getseconds ()
var res = ""
```

```
if (h <10)
res + = "0" + h
else res + = h
if (m <10) res + = ": 0" + m
else res + = ":" + m
if (s <10) res + = ": 0" + s
else res + = ":" + s
d.form1.rest.value = res
}
//->
</Script>
</HEAD>
<BODY bgcolor = "# FFFFCC">
<CENTER>
<IMG src = alarmWHT.gif> <br>
```

При натисканні кнопки <B> Час </B>, Ви дізнаєтеся, котра година

```
<FORM name = "form1">
<Input type = "button" value = Час onClick = "c1 ()">
<Input type = "text" size = 10 narae = "rest"> <br>
</FORM>
</BODY>
</HTML>
```

Можна зробити так, що через деякий заданий період значення часу буде оновлюватися. Для цього можна використовувати функцію setTimeout ("c1 ()", 3000). Функція setTimeout виконує зазначені в першому параметрі дії після закінчення інтервалу часу, що задається другим параметром. У наведеному прикладі через три секунди буде знову здійснений виклик функції c1.

Перераховані нижче методи дозволяють встановлювати різні значення для об'єкта Date.

-Метод setYear встановлює значення року для об'єкта Date.

-Метод setDate встановлює день місяця. Параметр повинен бути числом в діапазоні від 1 до 31.

-Метод setMonth встановлює значення місяця. Параметр повинен бути числом в діапазоні від 0 (січень) до 11 (грудень).

-Метод setHours встановлює годину для поточного часу, використовує ціле число від 0 (опівночі) до 23 для установки дати по 24-годинною шкалою.

-Метод setMinuts встановлює хвилини для поточного часу, використовує ціле число від 0 до 59.

-Метод setSeconds встановлює секунди для поточного часу, використовує ціле число від 0 до 59.

-Метод setTime встановлює значення об'єкта Date і повертає кількість мілісекунд, що пройшли з 1 січня 1970 року.

Приклад 2. П'ятниця 13

Напишемо сценарій, за допомогою якого визначаються всі дати в зазначеному році, що припадають на п'ятницю, 13 число.

При написанні сценарію будемо поступати таким чином: перебирати з допомогою циклу місяці і в кожному місяці встановлювати номер дня 13. Установка необхідної дати виконується використанням методів роботи з датою:

```
t.setYear (y)
 t.setMonth (i)
 t.setDate (13)
```

Далі слід перевірити, який номер дня відповідає цій даті. Якщо номер дорівнює п'яти ((t. Getoay ()) == 5), то день тижня - п'ятниця, знайдений місяць слід запам'ятати. Для формування відповіді використовується рядкова змінна c, після запам'ятовування назви місяця додається символ перекладу рядка.

```
 <HTML>
 <HEAD>
 <TITLE> В які місяці року 13 число потрапляє на п'ятницю? </TITLE>
 <Script language = "JavaScript">
 <! - //
 function def13 (obj)
 {Var t = new Date ()
 var c = ""
 var y = Number (obj.fye.value)
 for (var 1 = 0; i <= 11; i ++)
 {T.setYear (y)
 t.setMonth (i)
 t.setDate (13)
 if ((t.getDay ()) == 5)
 з = c + fmon (i) + "\ r \ n"
 }
 obj.res.value = з
 }
 function fmon (mont)
 {
 var s
```

```
switch (mont)
{Case 0 s = "січень"; break;
case 1 s = "лютий"; break;
case 2 s = "март"; break;
case 3 s = "квітень"; break;
case 4 s = "май"; break;
case 5 s = "червень"; break;
case 6 s = "липень"; break;
case 7 s = "серпень"; break;
case 8 s = "вересень"; break;
case 9 s = "жовтень"; break;
case 10: s = "листопад"; break;
case 11: s = "грудень"; break;
}
return s
}
//->
</Script>
</HEAD>
<BODY bgcolor = "# FFFFCC">
<H4> В які місяці заданого року число 13 потрапляє на п'ятницю? </H4>
<FORM name = "form1">
Введіть рік: <input type = "text" size = 8 name = "fye">
<Input type = "button" value = Найті onClick = "def13 (form1)"> <br>
<Textarea Cols = 30 rows = 4 name = res> </textarea> <br>
<Input type = "reset" value = Скасувати>
</FORM>
</BODY>
</HTML>
```

Завдання

1. Перевірте приклади з лабораторної роботи.

2. Напишіть сценарій, який за заданою датою визначає номер тижні в році.

3. Напишіть сценарій, який за датою народження людини визначає, під яким знаком зодіаку народилася людина.

4. У давньояпонському календарі був прийнятий 60-річний цикл, що складається з п'яти 12-річних підциклів. Підцикли позначалися назвами кольори: зелений, червоний, жовтий, білий,

чорний. Усередині кожного подцикла роки носили назви тварин: пацюка, корови, тигра, зайця, дракона, змії, коні, вівці, мавпи, курки, собаки і свині. Наприклад, 1984 рік (рік зеленої щури) був початком чергового циклу. Напишіть сценарій, який за заданою датою визначає назву року по старояпонского календарем.

Лабораторна робота №11

Робота з рядками

Рядкові літерали або рядкові змінні є в мові JavaScript об'єктом типу string, до якого можуть бути застосовані методи, визначені в мові. Створення нового об'єкта вимагає виклику функції-конструктора об'єкта. Для того щоб створити строковий об'єкт, треба застосувати конструктор newstring, наприклад:

s = newString ("результат =")

Об'єкт string має єдине властивість length (дліна_строкі). Вираз s.length видає значення 10, яка дорівнює довжині рядка, що міститься в строковому об'єкті s. Об'єкт string має два типи методів. З методами, безпосередньо впливають на саму рядок, ми зараз і познайомимося, розглядаючи приклади обробки текстової інформації.

Одним з часто використовуваних методів є метод виділення з рядка окремого символу. Метод charAt (ni) повертає символ, позицію якого визначає параметр ni. Символи в рядку перенумеровані, починаючи з 0.

Приклад 1. Висновок символів рядка в "стовпчик"

Напишемо сценарій, при виконанні якого заданий текст виводиться в "стовпчик", тобто на кожному рядку розміщується по одному символу.

При вирішенні завдання з заданої рядки послідовно вибираються символи. Формується новий рядок, в якій за кожним символом ставиться послідовність символів, що забезпечує перехід на новий рядок. Коли рядок результату сформована, то вона розміщується в текстовому полі форми, тим самим для початкового рядка здійснюється висновок в "стовпчик".

```
<HTML>
<HEAD>
<TITLE> Висновок символів рядка в "стовпчик" </TITLE>
<Script language = "JavaScript">
<! - //
function ttest (s)
{Var sres = "Прочитаний текст:" + "\ r \ n" + s + "\ r \ n" +
```

```
'Текст в "стовпчик":' + "\ r \ n"
var cur = ""
for (var i = 0; i <= s.length-1; i + = 1)
{C = s.charAt (i); cur + = c + "\ r \ n"}
sres + = cur
return sres
}
//->
</Script>
</HEAD>
<BODY bgcolor = "# FFFFCC">
<H4> Символи поточного рядка в стовпчик </H4>
<FORM name = "form1">
Введіть рядок: <input type = "text" size = 20 name = "st1"> <hr>
<Input type = "button" value = Виполніть onClick = "form1.res.value = ttest (form1.st1.value)">
<Input type = "reset" value = Очистити> <hr>
<Textarea cols = 20 rows = 7 name = res> </textarea>
</FORM>
</BODY>
</HTML>
```

Метод substr (n1, n2) дозволяє виділяти з рядка підрядок. Параметр n1 задає позицію першого символу підрядка; параметр n2 визначає кількість символів в підрядку. Наприклад, якщо рядок s = "збірник", то в результаті виконання substr (0,4) буде виділена подстрока "збір".

Приклад 2. Обчислення кількості повторень рядки в тексті

Напишемо програму, яка визначає, скільки разів заданий слово зустрічається в певному тексті.

У тексті слова розділяються пробілами. Після того як чергове слово знайдено, перегляд триває з символу, наступного за знайденим словом.

```
<HTML>
<HEAD>
<TITLE> Кількість даних слів в тексті </TITLE>
<Script language = "JavaScript">
<! - //
function numword (obj)
{Var h = obj.data.value
var s = obj.textin.value
s = " + s + "
```

```
h = " + h + "
var m = h.length
var res = 0
var i = 0
while (i <= s.length-1)
{Ch = s.substr (i, m)
if (ch == h) {res + = l; i = i + m-1}
else
i ++
}
obj.result.value = res
}
//->
</Script>
</HEAD>
<BODY bgcolor = "# FFFFCC">
<H4> Кількість даних слів в тексті </H4>
<FORM name = "form1">
Введіть текст: <br>
<Textarea name = "textin" rows = 4 cols = 20> </textarea> <hr>
Введіть слово: <input type = "text" name = "data" size = "8"> <hr>
<Input type = "button" value = "0пределіть" onClick = "numword (form1)"> <hr>
Кількість слів в тексті: <input type = "text" name = "result" size = 8> <hr>
<Input type = "reset" value = "Скасувати">
</FORM>
</BODY>
</HTML>
```

Завдання

1. Перевірити приклади з лабораторної роботи.
2. Слова в заданому тексті розділяються пробілами. Напишіть програму, яка визначає кількість слів у тексті.
3. Напишіть програму, в якій всі слова A замінені словом B, де A і B - задані слова, можливо, різної довжини.
4. Напишіть програму, яка "стискає" заданий текст, тобто замінює всі підряд йдуть прогалини на один.

Лабораторна робота №12

Масиви

Тип Array введений в JavaScript для можливості маніпулювання самими різними об'єктами, які може відображати Navigator. Це - список всіх гіпертекстових посилань даної сторінки, список всіх картинок на даній сторінці, список всіх елементів форми і т.п. Користувач може створити і свій власний масив, використовуючи, конструктор Array (). Робиться це в такий спосіб:

```
new_array = new Array ()
new_array5 = new Array (5)
colors = new Array ( "red", "white", "blue")
```

Розмірність масиву може змінюватися. Можна спочатку визначити масив, а потім привласнити одному з його елементів значення. Як тільки це відбудеться, зміниться і розмірність масиву:

```
colors = new Array ()
colors [5] = "red".
```

В даному випадку масив буде складатися з 6 елементів, тому що першим елементом масиву вважається елемент з індексом 0.

Для масивів визначені чотири методи: join, reverse, sort, concat. Join об'єднує елементи масиву в рядок символів, як аргумент в цьому методі задається роздільник:

```
colors = new Array ( "red", "white", "blue")
string = acolors.join ( "+")
```

В результаті виконання присвоювання значення рядку символів string ми отримаємо наступний рядок: string = "red + white + blue". Інший метод, reverse, змінює порядок елементів масиву на зворотний, метод sort відсортовує їх в лексикографічному порядку, а метод concat об'єднує два масиви.

У масивів є дві властивості: length і prototype. Length визначає число елементів масиву. Якщо потрібно виконати деяку рутинну операцію над усіма елементами масиву, то можна скористатися циклом типу:

```
color = new Array ( "red", "white", "blue")
n = 0 while (n! = colors.length)
{... оператори тіла циклу ...}
```

Властивість prototype дозволяє додати властивості до об'єктів масиву. Однак найчастіше в програмах на JavaScript використовуються вбудовані масиви, в основному графічні образи (Images) і гіпертекстові посилання (Links).

У новій версії мови з'явився конструктор для цього типу

об'єктів:

```
new_image = new Image ()
 new_image = new Image (width, height)
```

Приклад 1. Створення мультиплікації з використанням масивів.

Часто для створення мультиплікації формують масив графічних об'єктів, які потім прокручують один за іншим:

```
img_array = new Array ()
img_array [0] = new Image (50,100)
img_array [1] = new Image (50,100)
  ...
img_array [99] = new Image (50,100)
```

У об'єкта Image існує 10 властивостей, з яких, мабуть, найважливішим є src. Так, для присвоювання конкретних картинок елементам масиву img_array слід скористатися такою послідовністю команд:

```
img_array [0] .src = "image1.gif"
img_array [1] .src = "image2.gif"
 ...
 img_array [99] .src = "image100.gif"
```

В даному випадку можна було скористатися і циклом для присвоєння імен, так як вони можуть бути складені з констант і значення індексного змінної.

Розширюючи приклад з масивом Image, побудуємо тепер документ, в якому буде вбудована мультиплікація, визначена нашим масивом:

Приклад 1. Мультиплікація.

```
<HTML>
<HEAD>
<SCRIPT>
<! - //
function multi_pulti ()
{
img_array = new Array ()
for (var i = 0; i <4; i ++)
img_array [i] = new Image (50,100)
img_array [0] .src = "e1.jpg"
img_array [1] .src = "e2.jpg"
img_array [2] .src = "e3.jpg"
img_array [3] .src = "e4.jpg"
var t = new Date ()
p = -1
```

```
}
function s ()
{
p = p + 1
document.images [0] .src = img_array [p] .src
setTimeout ( "s ()", 100)
if (p == 3) p = -1
}
//->
</SCRIPT>
</HEAD>
<BODY onLoad = "multi_pulti ()">
<Img src = "e1.jpg">
<br>
<Input type = "Button" name = "But" value = "Подивитися" onClick = "s ()">
</BODY>
</HTML>
```

Далі розглянемо кілька класичних задач, присвячених роботі з масивами. Наведемо функції роботи з масивами, які цінні самі по собі і можуть застосовуватися в різних програмах.

Приклад 2. Бінарний пошук з формуванням таблиці результатів

Напишемо функцію, яка реалізує алгоритм бінарного пошуку таким чином, щоб під час роботи програми формувалася таблиця значень змінних i, j, k і деяких виразів.

```
<HTML>
<HEAD>
<TITLE> Бінарний пошук. Таблиця проміжних значень </TITLE>
<Script language = "JavaScript">
<! - //
var v = new Array (2, 3, 5, 6, 6, 7,10,11, 20, 25)
function testtab (obj, v, t)
{Var res = "i j k v [k] t <= v [k]" + "\ r \ n"
var i = 0
var j = v.length-1
var k
while (i <j)
{K = Math.round ((i + j) /2+0.5) -1
res = res + i + "" + j + "" + k + "" + "v [" + k + "] =" + v [k] + "" + t + "<=" + v [k] + " \ r \ n "
```

```
if (t <= v [k])
j = k
else
i = k + l
}
res + = "v [" + i + "] =" + v [i] + "\ r \ n"
obj.resultl.value = res
if (v [i] == t)
{Return i}
else return -1
}
function test (obj)
{Obj.datal.yalue = v}
//->
</Script>
</HEAD>
<BODY bgcolor = silver>
<H4> Реалізація алгоритму бінарного пошуку </H4>
<FORM name = "forml">
<Pre>
Масив: <INPUT type = "text" size = 40 name = "datal"> <hr>
Елемент: <INPUT type = "text" size = 20 name = "data2"> <hr>
Результат пошуку: <INPUT type = "text" size = 20 name = "result"> <hr>
Таблиця проміжних значень: <BR>
<Textarea cols = 50 rows = 7 name = "result1"> </textarea> <hr>
</PRE>
<Input type = "button" value = 0пределіть onClick = "test (form1); forml.result.value = testtab (form1, v, forml.data2.value)">
<Input type = "reset" value = Скасувати>
</FORM>
</BODY>
</HTML>
```

Завдання

1. Перевірити приклад 2 з лабораторної роботи.
2. Створити найпростіший мультиплікаційний сюжет з використанням масиву.
3. Задано одновимірний масив дійсних чисел. Напишіть сценарій, який визначає число позитивних елементів масиву.

4. Задано одновимірний масив дійсних чисел. Напишіть сценарій, що дозволяє знайти максимальний елемент в масиві.

Лабораторна робота №13

Форми

Перевірка інформації, введеної в форму

Форми широко використовуються в Інтернет. Інформація, введена в форму, часто посилається назад на сервер або відправляється по електронній пошті на якусь з адрес. Проблема полягає в тому, щоб переконатися, що введена користувачем у форму інформація коректна. Легко перевірити її перед пересиланням в Інтернет можна за допомогою мови JavaScript.

Спершу нам необхідно створити простий скрипт. Припустимо, HTML-сторінка містить два елементи для введення тексту. У перший з них користувач повинен вписати своє ім'я, в другій елемент - адреса для електронної пошти. Ви можете ввести туди якусь інформацію і натиснути клавішу. Спробуйте також натиснути клавішу, не ввівши в форму ніякої інформації.

Введіть ваше ім'я:

Введіть Вашу адресу e-mail:

Що стосується інформації, введеної в перший елемент, то Ви будете отримувати повідомлення про помилку, якщо туди нічого не було введено. Будь-яка представлена в елементі інформація буде розглядатися на предмет коректності. Звичайно, це не гарантує, що користувач введе не те ім'я. Браузер навіть не заперечуватиме проти чисел. Наприклад, якщо Ви введете '17', то отримаєте запрошення 'Ні 17!'. Так що ця перевірка не може бути ідеальна.

Другий елемент форми дещо складніше. Спробуйте ввести просту рядок - наприклад Ваше ім'я. Зробити це не вдасться до тих пір, поки Ви не вкажете @ у Вашому імені ... Ознакою того, що користувач правильно ввів адресу електронної пошти служить наявність символу @. Цій умові буде відповідати і одиночний символ @, навіть незважаючи на те, що це безглуздо. В Інтернет кожну адресу електронної пошти містить символ @, так що перевірка на цей символ тут доречна.

Як скрипт працює з цими двома елементами форми і як виглядає перевірка? Це відбувається наступним чином:

```
<Html>
<Head>
<Script language = "JavaScript">
<! - Приховати

function test1 (form) {
  if (form.text1.value == "")
    alert ( "Будь ласка, введіть рядок!")
  else {
   alert ( "Hi" + form.text1.value + "! Форма заповнена коректно!");
  }
}
function test2 (form) {
  if (form.text2.value == "" ||
      form.text2.value.indexOf ( '@', 0) == -1)
        alert ( "Невірно введена адреса e-mail!");
  else alert ( "OK!");
}
//->
</Script>
</Head>
<Body>
<Form name = "first">
Введіть Ваше ім'я: <br>
<Input type = "text" name = "text1">
<Input type = "button" name = "button1" value = "Перевірка"
onClick = "test1 (this.form)">
<P>
Введіть Вашу адресу e-mail: <br>
<Input type = "text" name = "text2">
<Input type = "button" name = "button2" value = "Перевірка"
nClick = "test2 (this.form)">
</Body>
</Html>
```

Розглянемо спочатку HTML-код в розділі body. Тут ми створюємо лише два елементи для введення тексту і дві кнопки. Кнопки викликають функції test1 (...) або test2 (...), в залежності від того, яка з них була натиснута. Як аргумент до цих функцій ми передаємо комбінацію this.form, що пізніше дозволить нам адресуватися в самій функції саме до тих елементів, які нам потрібні.

Функція test1 (form) перевіряє, чи є даний рядок порожній. Це

робиться за допомогою if (form.text1.value == "") .... Тут 'form' - це змінна, куди заноситься значення, отримане при виконанні функції від 'this.form'. Ми можемо витягти рядок, введену в розглянутий елемент, якщо до form.text1 пріпішем 'value'. Щоб переконатися, що рядок не є марною, ми порівнюємо її з "". Якщо ж виявиться, що введений рядок відповідає "", то це означає, що насправді нічого введено не було. І наш користувач отримає повідомлення про помилку. Якщо ж щось було введено вірно, користувач отримає підтвердження - ok.

Наступна проблема полягає в тому, що користувач може вписати в поле форми одні пробіли. І це буде прийнято, як коректно введена інформація! Якщо є бажання, то Ви звичайно можете додати перевірку такої можливості і виключити її.

Розглянемо тепер функцію test2 (form). Тут знову порівнюється введений рядок з порожньою - "" (щоб упевнитися, що щось дійсно було введено читачем). Однак до команди if ми додали ще дечого. Комбінація символів || називається оператором OR (АБО).

Команда if перевіряє, чим закінчується перше або друге порівняння. Якщо хоча б одне з них виконується, то і в цілому команда if має результатом true, а отже буде виконуватися наступна команда скрипта. Словом, Ви отримаєте повідомлення про помилку, якщо або надана Вами рядок порожня, або в ній відсутній символ @. (Другий оператор в команді if стежить за тим, щоб введений рядок містила @.)

Перевірка на присутність певних символів

У деяких випадках Вам знадобиться обмежувати інформацію, що вводиться в форму, лише деяким набором символів або чисел. Досить згадати про телефонні номери - представлена інформація повинна містити лише цифри (передбачається, що номер телефону, як такої, не містить ніяких символів). Нам необхідно перевіряти, чи є введені дані числом. Складність ситуації полягає в тому, що більшість людей вставляють в номер телефону ще й різні символи - наприклад: 01234-56789, 01234/56789 or 01234 56789 (з символом пробілу всередині). Не слід примушувати користувача відмовлятися від таких символів в телефонному номері. А тому ми повинні доповнити наш скрипт процедурою перевірки цифр і деяких символів.

Telephone:

Вихідний код цього скрипта:

```
<Html>
<Head>
<Script language = "JavaScript">
<! - hide
function check (input) {
```

```
  var ok = true;
  for (var i = 0; i <input.length; i ++) {
    var chr = input.charAt (i);
    var found = false;
    for (var j = 1; j <check.length; j ++) {
      if (chr == check [j]) found = true;
    }
    if (! found) ok = false;
  }
   return ok;
}
function test (input) {
  if (! check (input, "1", "2", "3", "4",
        "5", "6", "7", "8", "9", "0", "/", "-", "")) {
    alert ( "Input not ok.");
  }
  else {
    alert ( "Input ok!");
  }
}
//->
</Script>
</Head>
<Body>
<Form>
Telephone:
<Input type = "text" name = "telephone" value = "">
<Input type = "button" value = "Check"
  onClick = "test (this.form.telephone.value)">
</Form>
</Body>
</Html>
```

Функція test () визначає, які з введених символів визнаються коректними.

Надання інформації, введеної в форму

Які існують можливості для передачі інформації, внесеної в форму? Найпростіший спосіб полягає в передачі даних форми по електронній пошті (цей метод ми розглянемо детальніше).

Якщо Ви хочете, щоб за змінами, що вносяться в форму даними стежив сервер, то Ви повинні використовувати інтерфейс CGI (Common Gateway Interface). Останнє дозволяє автоматично обробляти

дані. Наприклад, сервер міг би створювати базу даних з відомостями, доступну для деяких з клієнтів. Інший приклад - пошукові сторінки, такі як Yahoo. Зазвичай в них представлена форма, що дозволяє створювати запит для пошуку в своїй основі даних. В результаті користувач отримує відповідь незабаром після того, як натискає на відповідну кнопку. Йому не доводиться чекати, поки люди, які відповідають за підтримання даного сервера, прочитають зазначені ним дані і знайдуть необхідну інформацію. Все це автоматично виконує сам сервер. JavaScript не дозволяє робити таких речей.

За допомогою JavaScript Ви не зможете створити книгу читацьких відгуків, оскільки JavaScript позбавлений можливості записувати дані в який-небудь файл на сервері. Робити це Ви можете тільки через інтерфейс CGI. Звичайно, Ви можете створити книгу відгуків, для якої користувачі надсилали відомості по електронній пошті. Однак в цьому випадку Ви повинні заносити відгуки вручну. Так можна робити, якщо Ви не припускаєте отримувати щодня по 1000 відгуків.

Відповідний скрипт буде простим текстом HTML.

```
<Form method = post action = mailto: your.address@goes.here
enctype = "text /plain">
Чи подобається Вам ця сторінка?
  <Input name = "choice" type = "radio" value = "1"> Зовсім ні. <br>
  <Input name = "choice" type = "radio" value = "2" CHECKED> Марна трата
часу. <br>
  <Input name = "choice" type = "radio" value = "3"> Найгірший сайт в
Мережі. <br>
  <Input name = "submit" type = "submit" value = "Send">
</Form>
```

Параметр enctype = "text /plain" використовується для того, щоб пересилати саме простий текст без будь-яких кодованих частин.

Якщо Ви хочете перевірити форму перш, ніж вона буде передана в мережу, то для цього можете скористатися програмою обробки подій onSubmit. Ви повинні помістити виклик цієї програми в тег <form>. наприклад:

```
function validate () {
  //check if input ok
  //...
  if (inputOK) return true
```

```
   else return false;
}
...
<Form ... onSubmit = "return validate ()">
...
```

Форма, складена таким чином, не буде послана в Інтернет, якщо в неї внесені некоректні дані.

Виділення певного елемента форми

За допомогою методу focus () Ви можете зробити вашу форму більш дружньою. Так, Ви можете вибрати, який елемент буде виділений в першу чергу. Або Ви можете наказати браузеру виділити ту форму, куди були введені невірні дані. Зробити це Ви можете за допомогою наступного фрагмента скрипта:

```
function setfocus () {
  document.first.text1.focus ();
}
```

Ця запис могла б виділити перший елемент для введення тексту в скрипті. Ви повинні вказати ім'я для всієї форми - в даному випадку вона називається first - і ім'я одного елемента форми - text1. Якщо Ви хочете, щоб при завантаженні сторінки даний елемент виділявся, то для цього Ви можете доповнити тег <body> атрибутом onLoad. Це буде виглядати як:

```
<Body onLoad = "setfocus ()">
```

Залишається ще доповнити приклад наступним чином:

```
function setfocus () {
  document.first.text1.focus ();
  document.first.text1.select ();
}
```

Спробуйте наступний код:

bla bla bla

При цьому не тільки буде виділено елемент, але і що знаходиться в нім.

Завдання

Виконати приклади, наведені в лабораторній роботі.

Контрольні питання:

1. Для чого іноді потрібно обмежувати інформацію, що вводиться в форму? Як це зробити на JavaScript?
2. Які існують способи передачі даних форми?
3. Для чого призначений метод focus ()?

4. З якою метою використовується програма обробки подій onSubmit?

Лабораторна робота №14

Об'єкт Image

Зображення на web-сторінці

Розглянемо тепер об'єкт Image, який став доступний, починаючи з версії з 1.1 мови JavaScript (тобто з Netscape Navigator 3.0). За допомогою об'єкта Image Ви можете вносити зміни в графічні образи, присутні на web-сторінці. Зокрема, це дозволяє нам створювати мультиплікацію.

Зауважимо, що користувачі браузерів старіших версій (таких як Netscape Navigator 2.0 або Microsoft Internet Explorer 3.0 - тобто використовують версію 1.0 мови JavaScript) не зможуть запускати скрипти, наведені в цій частині опису. Або, в кращому випадку, на них не можна буде отримати польний ефект.

Давайте спочатку розглянемо, як з JavaScript можна адресуватися до зображень, представлених на web-сторінці. В даному мові все зображення з'являються у вигляді масиву. Масив цей називається images і є властивістю об'єкта document. Кожне зображення на web-сторінці отримує порядковий номер: перше зображення отримує номер 0, друге - номер 1 і т.д. Таким чином, до першого зображенню ми можемо звернутися, записавши document.images [0].

Кожне зображення в HTML-документі розглядається як об'єкт Image. Об'єкт Image має певні властивості, до яких і можна звертатися з мови JavaScript. Наприклад, Ви можете визначити, який розмір має зображення, звернувшись до його властивостей width і height. Тобто по запису document.images [0] .width Ви можете визначити ширину першого зображення на web-сторінці (в пікселях).

На жаль, відстежувати індекс всіх зображень може виявитися скрутним, особливо якщо на одній сторінці у Вас їх досить багато. Ця проблема вирішується призначенням зображенням своїх власних імен. Так, якщо Ви заводите зображення за допомогою тега

<Img src = "img.gif" name = "myImage" width = 100 height = 100>

то Ви зможете звертатися до нього, написавши document.myImage або document.images ["myImage"].

Завантаження нових зображень

Хоча звичайно і добре знати, як можна отримати розмір зображення на web-сторінці, це не зовсім те, чого б ми хотіли. Ми

хочемо здійснювати зміну зображень на web-сторінці і для цього нам знадобиться атрибут src. Як і в разі тега <img>, атрибут src містить адресу представленого зображення. Тепер - в мові JavaScript 1.1 - Ви маєте можливість призначати нову адресу зображення, вже завантаженому в web-сторінку. І в результаті, зображення буде завантажено з цього нового адреси, замінивши на web-сторінці старе. Розглянемо наприклад запис:

<Img src = "img1.gif" name = "myImage" width = 100 height = 100>

Тут завантажується зображення img1.gif і отримує ім'я myImage. У наступному рядку відображається той img1.gif замінюється вже на нове - img2.gif:

document.myImage.src = "img2.src";

При цьому нове зображення завжди отримує той же розмір, що був у старого. І Ви вже не можете змінити розмір поля, в якому це зображення розміщується.

Упереджувальний завантаження зображення

Один з недоліків такого підходу може полягати в тому, що після запису в src нового адреси починає процес завантаження відповідного зображення. І оскільки цього не було зроблено заздалегідь, то ще пройде деякий час, перш ніж нове зображення буде передано через Інтернет і встане на своє місце. У деяких ситуаціях це допустимо, однак часто подібні затримки неприйнятні. І що ж ми можемо зробити з цим? Звичайно, вирішенням проблеми була б випереджала завантаження зображення. Для цього ми повинні створити новий об'єкт Image. Розглянемо наступні рядки:

hiddenImg = new Image ();

hiddenImg.src = "img3.gif";

У першому рядку створюється новий об'єкт Image. У другому рядку зазначається адреса зображення, яке в подальшому буде представлено за допомогою об'єкта hiddenImg. Як ми вже бачили, запис нового адреси в атрибуті src змушує браузер завантажувати зображення із зазначеної адреси. Тому, коли виконується другий рядок нашого прикладу, починає завантажуватися зображення img2.gif. Але як мається на увазі самою назвою hiddenImg ("прихована картинка"), після того, як браузер закінчить завантаження, зображення на екрані не з'явиться. Воно буде лише буде збережено в пам'яті комп'ютера (або точніше в кеші) для подальшого використання. Щоб викликати

зображення на екран, ми можемо скористатися рядком:

document.myImage.src = hiddenImg.src;

Але тепер зображення вже негайно витягується з кешу і показується на екрані. Таким чином, зараз ми управляли упреждающей завантаженням зображення.

Звичайно браузер повинен був до моменту запиту закінчити попереджувальну завантаження, щоб потрібне зображення було показано без затримки. Тому, якщо Ви повинні попередньо завантажити велику кількість зображень, то може мати місце затримка, оскільки браузер буде зайнятий завантаженням всіх картинок. Ви завжди повинні враховувати швидкість зв'язку з Інтернет - завантаження зображень не стане швидше, якщо користуватися тільки що показаними командами. Ми лише намагаємося трохи раніше завантажити зображення - тому і користувач може побачити їх раніше. В результаті і весь процес пройде більш гладко.

Зміна зображень в зв'язку з подіями

Ви можете створити гарні ефекти, використовуючи зміну зображень в якості реакції на певні події. Наприклад, Ви можете змінювати зображення в той момент, коли курсор миші потрапляє на певну частину сторінки. Перевірте, як працює такий приклад, просто помістивши курсор миші на картинку (втім, при цьому Ви отримаєте повідомлення про помилку, якщо користуєтеся браузером, що підтримує лише JavaScript 1.)

Image 1

Вихідний код цього прикладу виглядає наступним чином:

```
<A href = "#"
  onMouseOver = "document.myImage2.src = 'img2.gif'"
  onMouseOut = "document.myImage2.src = 'img1.gif'">
<Img src = "img1.gif" name = "myImage2" width = 160 height = 50 border = 0> </a>
```

При цьому можуть виникнути такі проблеми:

• користувач користується браузером, які не мають підтримки JavaScript 1.1.

• Друге зображення не було завантажено.

• Для цього ми повинні писати нові команди для кожного зображення на web-сторінці.

• Ми хотіли б мати такий скрипт, який можна було б використовувати в багатьох web-сторінках знову і знову, і без великих переробок.

Тепер ми розглянемо повний варіант скрипта, який міг би вирішити ці проблеми. Хоча скрипт і став набагато довше - але написавши його один раз, Ви не більше будете турбуватися про ці проблеми.

Щоб цей скрипт зберігав свою гнучкість, слід дотримуватися дві умови:

• Чи не обмовляється кількість зображень - не повинно мати значення, скільки їх використовується, 10 або 100

• Чи не обмовляється порядок проходження зображень - повинна існувати можливість змінювати цей порядок без зміни самого коду

Подивимося цей код в роботі:

Розглянемо скрипт:

```
<Html>
<Head>
<Script language = "JavaScript">
<! - hide
  //ok, у нас браузер з підтримкою JavaScript
  var browserOK = false;
  var pics;
//->
</Script>
<Script language = "JavaScript1.1">
<! - hide
  //браузер з підтримкою JavaScript 1.1!
  browserOK = true;
  pics = new Array ();
//->
</Script>
<Script language = "JavaScript">
<! - hide
var objCount = 0; //кількість зображень на web-сторінці
function preload (name, first, second) {
  //попередньо \ я завантаження зображень і розміщення їх в масиві
  if (browserOK) {
```

```
        pics [objCount] = new Array (3);
        pics [objCount] [0] = new Image ();
        pics [objCount] [0] .src = first;
        pics [objCount] [1] = new Image ();
        pics [objCount] [1] .src = second;
        pics [objCount] [2] = name;
        objCount ++;
      }
    }
    function on (name) {
      if (browserOK) {
        for (i = 0; i <objCount; i ++) {
         if (document.images [pics [i] [2]]! = null)
          if (name! = pics [i] [2]) {
            //повернути в вихідну систему \ яние все інше зображення \ я
            document.images [pics [i] [2]]. src = pics [i] [0] .src;
          } Else {
            //показувати другу картинку, оскільки курсор перетинає дане зображення
            document.images [pics [i] [2]]. src = pics [i] [1] .src;
          }
        }
      }
    }
    function off () {
      if (browserOK) {
        for (i = 0; i <objCount; i ++) {
         //повернути в вихідну систему \ яние все зображення \ я
         if (document.images [pics [i] [2]]! = null)
          document.images [pics [i] [2]]. src = pics [i] [0] .src;
        }
      }
    }
    //заздалегідь файли зображення - Ви повинні тут вказати
    //зображення, які потрібно завантажити заздалегідь, а також об'єкт Image,
    //до якого вони відносяться (перший аргумент). Саме цю частину
    //потрібно коригувати, якщо Ви хочете використовувати скрипт
    //стосовно до інших зображень (звичайно це не звільняє
```

```
//Вас від обов'язку підредагувати в документі також і розділ body)
preload ( "link1", "img1f.gif", "img1t.gif");
preload ( "link2", "img2f.gif", "img2t.gif");
preload ( "link3", "img3f.gif", "img3t.gif");
//->
</Script>
<Head>
<Body>
<A href = "link1.htm" onMouseOver = "on ( 'link1')"
  onMouseOut = "off ()">
<Img name = "link1" src = "link1f.gif"
  width = "140" height = "50" border = "0"> </a>
<A href = "link2.htm" onMouseOver = "on ( 'link2')"
  onMouseOut = "off ()">
<Img name = "link2" src = "link2f.gif"
  width = "140" height = "50" border = "0"> </a>
<A href = "link3.htm" onMouseOver = "on ( 'link3')"
  onMouseOut = "off ()">
<Img name = "link3" src = "link3f.gif"
  width = "140" height = "50" border = "0"> </a>
</Body>
</Html>
```

Даний скрипт поміщає все зображення в масив pics. Створює цей масив функція preload (), яка викликала в самому початку. Виклик функції preload () виглядає просто як:

```
preload ( "link1", "img1f.gif", "img1t.gif");
```

Це означає, що скрипт повинен завантажити з сервера два зображення: img1f.gif і img1t.gif. Перше з них - це та картинка, яка буде представлена, поки курсор миші не влучає у область зображення. Коли ж користувач поміщає курсор миші на зображення, то з'являється друга картинка. При виконанні функції preload () в якості першого аргументу ми вказуємо слово "link1" і тим самим ставимо на web-сторінці об'єкт Image, якому і будуть належати обидва попередньо завантажених зображення. Якщо Ви подивіться в нашому прикладі в розділ <body>, то виявите зображення з тим же ім'ям link1. Ми користуємося НЕ порядковий номер, а саме ім'я зображення для того, щоб мати можливість переставляти зображення на web-сторінці, що не переписуючи при цьому сам скрипт.

Обидві функції on () і off () викликаються за допомогою програм обробки подій onMouseOver і onMouseOut. Оскільки сам елемент

image не може відстежувати події MouseOver і MouseOut, то ми зобов'язані зробити на цих зображеннях ще й посилання.

Можна бачити, що функція on () повертає всі зображення, крім зазначеного, в початковий стан. Робити це необхідно тому, що в іншому разі виділення можуть виявитися відразу кілька зображень (справа в тому, що подія MouseOut не буде зареєстрований, якщо користувач перемістить курсор з зображення відразу за межі вікна).

Зображення - без сумніву могутній засіб поліпшення Вашої web-сторінки. Об'єкт Image дає Вам можливість створювати дійсно складні ефекти. Однак зауважимо, що не всяке зображення або програма JavaScript здатне поліпшити Вашу сторінку. Якщо Ви пройдетеся по Мережі, то зможете побачити безліч прикладів, де зображення використані найжахливішим способом. Чи не кількість зображень робить Вашу web-сторінку привабливою, а їх якість. Сама завантаження 50 кілобайт поганий графіки здатна викликати роздратування. При створенні спеціальних ефектів з зображеннями за допомогою JavaScript пам'ятайте про це і ваші відвідувачами /клієнтами будуть частіше повертатися на Ваші сторінки.

Контрольні питання:

1. Для чого призначений об'єкт Image?
2. Як з JavaScript можна адресуватися до зображень?
3. Для чого служить атрибут src тега <img>?
4. Щоб скрипт зміни

Рекомендована література

1. Попов В. Практикум по Интернет-технологиям /В. Попов. – Санкт-Петербург : Питер, 2012. – 480 с.

2. Тарнавський Ю. А. Практикум з Інтернет-технологій: метод. вказівки до викон. лаб. робіт /Ю. А. Тарнавський. – Київ : МАУП, 2004. – 136 с.

3. Гаевский, А. Ю. 100% самоучитель по созданию Web-страниц и Web-сайтов: HTML и JavaScript /А. Ю. Гаевский, В. А. Романовский. - Москва : Технолоджи - 3000 : Триумф, 2008. - 457 с.

4. Алексеев А.П. Введение в Web-дизайн. М.: СОЛОН-ПРЕСС, 2008. — 192 с.

5. Флэнаган Д. JavaScript. Подробное руководство, 6-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2012. – 1080 с., ил.

6. Морган, Ник. JavaScript для детей. Самоучитель по программированию / Ник Морган; пер. с англ. — М. : Манн, Иванов и Фербер, 2016. — 288 с.

7. Дмитриева М. В. Самоучитель JavaScript — СПб.: БХВ-Петербург, 2001. — 512 с.: ил.