# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет “Львівська політехніка”**



**Інститут післядипломної освіти**

**ЗВІТ**

**Про виконання лабораторної роботи №2**

**«Опрацювання мовою Java Script**

**подій на сторінці HTML»**

**з дисципліни «Програмування інтернет застосувань»**

Виконав:

слухач групи ПЗС-11

Гринчук Тарас

Прийняв:

ст. викл.Сенів М.М.

« »\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 р.

∑ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

ЛЬВІВ – 2014

**Тема роботи**: Опрацювання мовою Java Script подій на сторінці HTML.

**Мета роботи:** Оволодіти структурою та методами мови JavaScript для опрацювання подій.

## 1. Теоретичні відомості

Події та обробники подій є дуже важливою частиною у програмування на JavaScript. ***Події*** (*Events*), головним чином, ініціюються тими або іншими діями користувача. Події – це дії, які відбуваються, внаслідок того, що робить користувач. Наприклад, якщо користувач клацає по деякій кнопці, відбувається подія Click. Якщо миша перетинає яке-небудь посилання - відбувається подія MouseOver. Існує певний набір подій, які розпізнає той чи інший броузер.

Ми можемо примусити нашу JavaScript-програму реагувати на деякі з них. І це може бути виконано за допомогою спеціальних програм ***обробки подій***. Так, в результаті клацання по кнопці може створюватися випадаюче вікно. Це означає, що створення вікна повинно бути реакцією на подію Click. Програма - обробник подій, яку ми повинні використати в даному випадку, називається onClick. І вона повідомляє комп'ютер, що потрібно робити, якщо відбудеться дана подія.

Обробник подій записується в документ як атрибут тега HTML, до якого ви приписуєте код JavaScript. Наприклад, ви створили ***функцію*** JavaScript, і назвали її compute. Ви можете примусити броузер виконувати цю функцію, коли користувач натискає на кнопку, в якій до onClick приписаний обробник результату кнопки:

INPUT TYPE="button" VALUE="Calculate" onClick="compute(this.form)"

Ви можете поміщати будь-які твердження JavaScriptусередині кавичок onClick. Ці твердження будуть виконані, коли користувач натискатиме на кнопку. Якщо Ви хочете включити більш ніж одне твердження, то окремі твердження записуються через крапку з комою (;). Взагалі, це – непогана ідея визначати ***функцію*** для обробників подій тому що:

* це робить ваш код мобільним, оскільки ви можете використовувати ту ж саму функцію в багатьох різних місцях.
* це робить ваші твердження більш легкими для читання.

В даному прикладі використовується this.form, щоб звернутися до поточної форми. Ключове слово звертається до об'єкту (об'єкту кнопки у вищезазначеному прикладі). Потім конструкція this.form звертається до форми, що містить кнопку. Далі – обробник події onClick робить запит до функції compute(), з поточною формою this.form, як параметр функції.

Події звертаються до тегів HTML таким чином:

* події **Focus, Blur, Change**: text fields, textareas, і selections;
* подія **Click**: buttons, radio buttons, checkboxes, submit buttons, reset buttons, links;
* подія **Select**: text fields, textareas;
* подія **MouseOver**: links.

Ви можете використовувати в скрипті безліч різних типів функцій обробки подій. Щоб дізнатись про усі існуючі обробники подій, звертайтеся до відповідного довідника. Наведемо лише деякі з них:

* **onLoad** - виконання скрипта або функції при завантаженні;
* **onChange** - породжується при зміні значення елемента форми;
* **onClick** - породжується при виборі об'єкту (button, checkbox і т.п.);
* **onSelect** - породжується при виборі текстового об'єкту (text, textarea);
* **onSubmit** - при натисненні на кнопку Submit;
* **onUnload** - при переході до іншої сторінки.

## 2. Хід роботи

**Варіант 5.** Створити скрипт, у вікні якого після подвійного кліку виникає геометрична фігура і рухається в межах прямокутника. Кожен наступний подвійний клік збільшує кількість фігур на одиницю. Клавіша “delete” знищує всі фігури.

В редакторі notepad++ наберемо текст HTML документа:

<html>

<head>

<title>Лабораторна робота №2</title>

</head>

<body onload="startCanvas();">

Подвійний клік миші - додати фігуру;<br>

Клавіша delete - очистити екран.<br>

<canvas id="canvas" width="1000" height="1000" ondblclick = "canvasClk()"></canvas>

<script language="JavaScript">

var visible = true; //перемикач між малюванням фігур та очистці екрану

var shape = new Array(100); //масив координат фігур по вертикалі

var k = 0; //к-сть фігур

function startCanvas() {

var canvas = document.getElementById('canvas');

// Перевірка чи "бачить" браузер canvas

if (canvas.getContext) {

var ctx = canvas.getContext('2d'); // Получаємо 2D контекст

if(visible) {

//збільшуємо координату у для всіх фігур

for(var i = 0; i < k; i++) {

shape[i] += 5;

//якщо вийшли за канву - то малюємо зверху

if(shape[i] > canvas.height) shape[i] = 10;

//намалювати прямокутник

ctx.strokeRect(15+i\*60, shape[i], 50, 50);

}

}

else // Очистка всієї канви

ctx.clearRect(0, 0, canvas.width, canvas.height);

}

//перемикач: міняємо на протилежний стан

visible = !visible;

//повторюємо кожні 100 мілісекунд

setTimeout("startCanvas()",100);

}

//подвійний клік на канві

function canvasClk() {

//ініціалізуємо новий елемент масиву координат

shape[k++] = 10;

}

//подія нажаття клавіші клавіатури

window.onkeydown = pressed;

function pressed(e) {

key = e.keyCode || e.which;

//якщо delete - то обнуляємо лічильник фігур

if(key == 46) k = 0;

}

</script>

</body>

</html>

Запустимо даний документ у браузері (рис. 2.1):

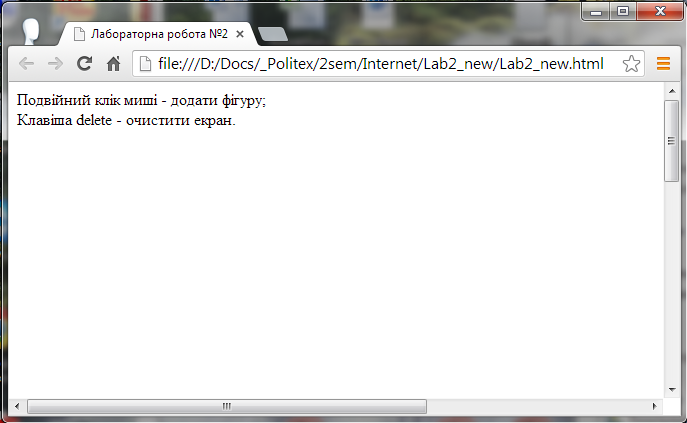


Рис. 2.1. Вигляд html-документа у браузері

Зробимо подвійний клік на формі вікна (рис. 2.2):

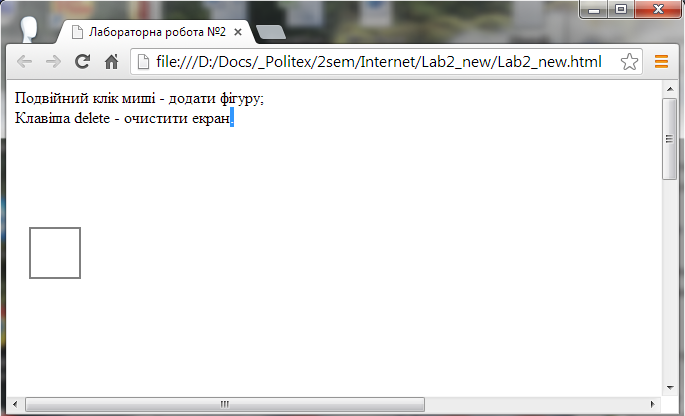


Рис. 2.2

Зробимо ще декілька подвійних натискань мишою (рис. 2.3):

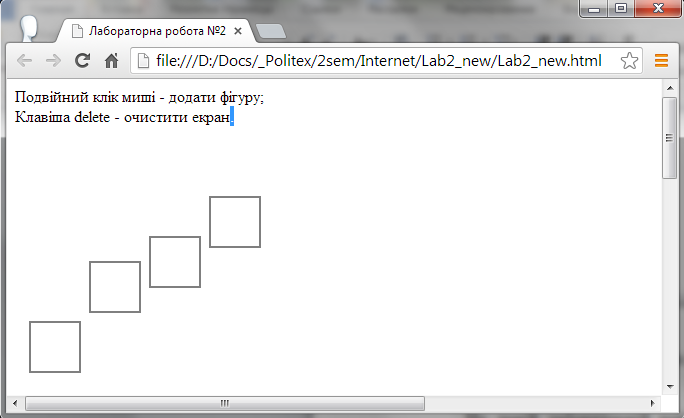


Рис. 2.3

Натиснемо на клавіатурі клавішу delete (рис. 2.4):

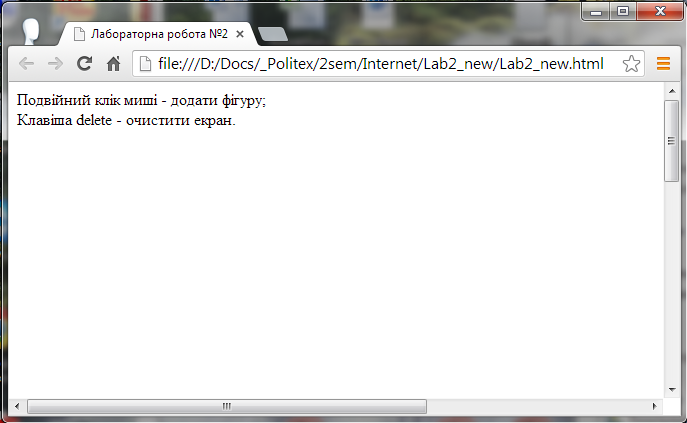


Рис. 2.4

Якщо зараз зробити подвійний натиск на вікні – то фігура знову почне переміщатися з початкової позиції. (рис. 2.5):

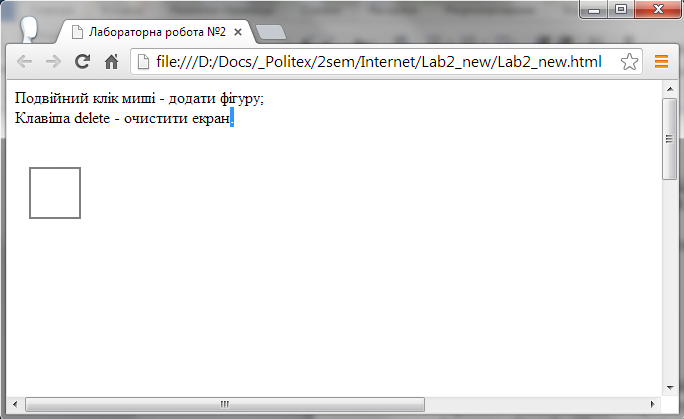


Рис. 2.5

## ВИСНОВКИ

На даній лабораторній роботі я вивчив структуру об’єктів html-документа в JavaScript, навчився опрацьовувати основні події, які виникають при роботі з web-документом на скриптовій мові програмування JavaScript.