ПРАКТИЧНА РОБОТА №5
З навчальної дисципліни «Веб-технології»
на тему: «Основи програмування сценаріїв веб-сторінки за допомогою мови JavaScript»

Навчальний час: 2 год

Мета:

1. Ознайомитися з основними базовими поняттями мови JavaScript: змінні, події, функції, цикли.
2. Навчитись оголошувати змінні та привласнювати їм значення.
3. Вміти вводити цифрові дані за допомогою функції **prompt,** коректно їх перетворювати в цифровий формат, виконувати математичні операції за допомогою методів JavaScript-об'єкта **Math**

Хід роботи:

1. Ознайомитись із короткими теоретичними відомостями;
2. Виконати тренувальні завдання із порядку виконання роботи для здобуття навичок роботи із JS.
3. Виконати практичні завдання, роблячи скрін-шоти виконання скриптів у браузері та поміщаючи їх у звіт.
4. Оформити звіт, помістивши туди фрагменти коду для кожного сценарію та результат виконання (скрін-шот)
5. Відповісти на питання для самоперевірки.

Короткі теоретичні відомості

1. Змінні в JavaScript і їх значення

Змінні в JavaScript так само, як й в інших мовах програмування, являють собою контейнери, що містять *значення* змінної, але на відміну від таких мов як Pascal або Java немає обмежень на *тип* значення змінної. У різні моменти часу змінна може мати строкове значення, у ній може зберігатися ціле число або число із плаваючою крапкою.

В JavaScript бажано повідомляти змінні, що робиться в такий спосіб:

var ім'я_змінної;

При оголошенні змінна може одержати початкове значення:

var ім'я_змінної = значення;

Тут ім'я_змінної повинне бути унікальним для сценарію, імена змінних повинні підкорятися наступним правилам: складатися з латинських букв a-z й A-Z, цифр від 0 до 9 і символу підкреслення "_". Ім'я змінної не може починатися із цифри.

Оголошення декількох змінних можна зробити в такий спосіб:

var a, A, B, b=15;

Прописні й малі літери в іменах змінних JavaScript на відміну від HTML розрізняються, так що a й A - це різні змінні.

Наступні імена змінних є неприпустимими: 8ab, va$, sф, тому що перше ім'я починається із цифри, а друг і третє містить неприпустимі символи.

Крім цього імена змінних не повинні збігатися із зарезервованими словами JavaScript й Java.

В JavaScript змінним можуть бути привласнені значення п'яти типів.

Числові значення можуть бути цілочисельними, наприклад:

var xx = 10; // Змінна xx містить ціле десяткове значення 10

Якщо значення починається з 0x, то це означає, що змінної привласнено шістнадцятирічне значення:

var fx = 0x10; // Змінної fx привласнено шістнадцятирічне значення

// 0x10, що відповідає десятковому значенню 16

Цілочисельні значення, що починаються з 0 й утримуючі тіль-ки цифри від 0 до 7, сприймаються JavaScript як восьмеричне значен-ня:

var kx = 010; // Змінної fx привласнене восьмеричне

// значення 010, що відповідає десятковому // значенню 7

Крім цілочисельених значень змінним можуть привласнюва-тися значення із плаваючою крапкою, нижче наведені різні записи зна-чень із плаваючою крапкою:

var xl = 1.0, zl = -2.5, z2 = -1.23e-1б

Змінні JavaScript можуть приймати логічні значення true (іс-тина) і false (неправда):

var btrue = true, bfalse = false; // Логічні значення

Строкове значення полягає або в подвійні, або одинарні парні лапки.

var s1 = "Рядок, укладений у подвійні лапки";

var s2 = 'Рядок, укладений у подвійні лапки';

У рядок, обмежений подвійними лапками, можуть бути вклю-чені одинарні лапки й навпаки, наприклад:

var s3="Подвійні й одинарні (') лапки."

var s4='Одинарні й подвійні ("") лапки.'

Для вставки спеціальних символів у рядки використаються послідовності, що складаються із двох символів, що починаються зі зворотної косої риски:

\’ – одинарна лапка;

\‖ – подвійна лапка;

\\ - зворотна коса риска;

\n – переведення на новий рядок;

\r – повернення каретки;

\t – табуляція.

2. функції в JavaScript

Функції в JavaScript відіграють центральну роль, зокрема за допомогою функцій створюються класи об'єктів. Функції поділяють на користувацькі і вбудовані. Шаблон опису функції, що створена користувачем, має вигляд:

function *Ім'я_функції* (список_параметрів) {
Опис функції }
Правила іменування функцій такі ж, як і змінних. Параметри в списку розділяються комами, наприклад:
function f1 (a, b){
// Ця функція із двома параметрами нічого не робить
}
Список параметрів може бути порожнім:
function f2(){
// Ця функція з без параметрів також нічого не робить
}
Тіло функції уміщують в операторні дужки {...}, як показано вище
Функція може повертати значення, для цього необхідно вказа-ти це значення в операторі return:
function f3(){
return "Значення, що вертає функція" }
Ця функція повертає рядок "Значення, що вертає функція" Для виклику функції досить указати ім'я функції, а в дужках задати список значень параметрів:
f2()
Якщо функція повертає значення, то її виклик має вигляд:
s = f3()
Якщо у функції немає параметрів, кодування дужок при її ви-клику обов'язково.
У випадку, якщо забути вставити оператор return, функція по-вертає значення *undefined.*

3.Вбудовані функції в JavaScript

Функція isNaN.

Функція оцінює аргумент на невизначеність. Синтаксис:
isNaN(testValue)
де testValue - значення, яке ви хочете оцінити.
На платформах, які підтримують NaN, функції parseFloat і parseInt (див. нижче) повертають ―NaN‖, коли значення не є числом. isNaN повертає true якщо ―NaN,‖ і false в іншому випадку. В наступній програмі оцінюється значення floatValue і викликається відповідна процедура:

floatValue=parseFloat(toFloat) if (isNaN(floatValue))
{ notFloat() } else { isFloat() }

Функції parseInt і parseFloat.

Ці функції повертають числове значення рядкового аргументу.

Синтаксис parseFloat:

parseFloat(str)

parseFloat виконує лексикографічний розбір рядка і, якщо це можливо, повертає число з плаваючою крапкою.

Якщо в рядку зустрічаються символи відмінні від знаку (+ або -), цифри (0-9), десяткової крапки або експоненти, розбір припиняється і повертається значення, одержане до цього моменту, решта символів ігнорується. Якщо найперший символ не може бути перетворений в число повертається ―NaN‖ (not a number).

Синтаксис parseInt:

parseInt(str [, radix])

parseInt виконує лексикографічний розбір першого аргументу - рядка str, і повертає ціле по підставі, заданим другим необов'язковим аргументом radix. Наприклад radix 10 показує, що рядок потрібно перетворювати в десяткове число, 8 у вісімковее, 16 в шістнацяткове і так далі. При підставі більше десять для позначення цифр використовуються букви в алфавітному порядку. Перетворення виконується до першого неприпустимого символу, решта символів ігнорується, якщо перший символ не може бути перетворений повертається ―NaN‖.

4 Області дії змінних

Змінні, оголошеними в тілі сценаріїв, доступні в будь-якому його місці. Якщо ж змінна оголошена у функції, то доступ до неї є тільки в цій функції.

Хоча в JavaScript можна не повідомляти змінні, настійно рекомендується робити це. При явному оголошенні змінних легше ви-значити область їхньої дії, що у свою чергу прискорює налагодження сценаріїв.

5 Використання арифметичних операторів

Арифметичні оператори використовуються для складання математичних рівнянь. За допомогою цих рівнянь обчислюється необхідний результат, що потім може бути відображений браузером.

Для простих операцій (наприклад, додавання) як операнди можуть використовуватися змінні і літерали. Літерали можуть бути числовими і строковими значеннями.

Мова JavaScript підтримує ті ж правила по виконанню математичних операцій, які використаються при обчисленнях, — спочатку виконуються операції з найбільшим пріоритетом — вираження в дужках, потім операції множення (позначається символом "*") і розподілу (позначається символом "/"), а потім вирахування (позначається сим-волом "-" або "-") і підсумовування (позначається символом "+"). Операції, що мають однакові пріоритети, виконуються в порядку проходження ліворуч праворуч. Повний список усіх - математичних опера-цій й їхніх скорочень можна знайти у будь-якому довіднику JavaScript

6 Вирішення математичних завдань із використанням функцій prompt й alert

Змінні в мові JavaScript не є типізованими, тому їхній тип (число, текст і т.д.) визначається лише під час присвоєння їм значення.
Розглянемо найпростіший приклад, у якому буде вводиться число й до нього буде додаватися одиниця:

```
<SCRIPT type="text/javascript" language="JavaScript">
var x = prompt('Input x','');
x = parseInt(x, 10);
x += 1;
alert( x );
</SCRIPT>
```

Уведення значення числа здійснюється за допомогою функції prompt, а вивід результату - за допомогою функції alert. Перетворення значення змінної x з текстового типу в цілочисельний здійснюється за допомогою функції parselnt. Крім того, необхідно звернути увагу на те, що при кодуванні математичної операції тут використається скорочений запис, тобто x += 1, який записано замість x = x + 1 (подібно до мови C та C++). Якщо у відповідь на запит ви введете число 33, то функція alert виведе діалогове вікно із значенням «34».
Результат виконання JavaScript-програми може виводитися як за допомогою функції alert, так і методом write об'єкта document. Напе-вно, ви звернули увагу на другий параметр функції parselnt (його значення дорівнює 10) - він задає основу системи числення. Можете спробувати використати як значення цього параметра числа 2, 8 й 16 і подивитися на отриманий результат. При уведенні речовинних чисел функція parselnt усікає дробову частину значення. І це закономірно, тому що функція parselnt здійснює перетворення текстового

подання числа в цілочисельний тип. Перетворення в речовинний тип даних (дробові значення) здійснюється за допомогою функції parseFloat.

Ускладнимо наш приклад :

```
<SCRIPT type="text/javascript" language="JavaScript">
var x = prompt('Input x',");
x = parseFloat(x);
var y = Math.cos(x);
alert( 'y = ' + y );
</SCRIPT>
```

крім використання функції parseFloat обчислимо значення косинуса для уведеного користувачем значення. В JavaScript реалізація всіх математичних функцій виконана в одному JavaScript-об'єкті Math, тому при виклику будь-якої математичної функції перед її назвою пишеться назва об'єкта Math і відокремлюється від назви функції крапкою. Попутно "оптимізуємо" программу - у діалогове вікно з результатами буде виводитися не тільки значення змінної, але і її назва.

7 Умовні оператори

Умовні оператори використаються для керування виконанням сценарію. Оператор if дозволяє перевірити деяку умову, при істинності якої, буде виконаний операторний блок, що є наступним за умовою.

Синтаксис оператора умови:

```
if (умова) {
// Оператори сценарію, виконувані в тому випадку, якщо
// значення вираження умова дорівнює true
}
```

Якщо в операторному блоці необхідно виконати єдиний опе-ратор, то фігурні дужки можна опустити.

Умова повинна бути виразом, що приводиться до логічного значення, наприклад:

```
if (a>2) alert('Вираження (a>2) істинно');
```

При виконанні цього фрагмента сценарію діалогове вікно з повідомленням з'явиться тільки в тому випадку, якщо a>2.

Розроблювачі JavaScript, перебуваючи під впливом мови програмування C, зробили припустимим і таку умовоу:

```
if (a) {операторний блок}
```

Якщо змінна a має числове значення, то умова стане помилковою тільки в тому випадку, якщо a дорівнює 0. Якщо ж значенням a є об'єкт, то значення умови хибне, якщо a == null. Нагадаємо, що null є об'єктом.
За допомогою оператора else можна розширити функціональність умовного оператора так, що у випадку істинності умови виконувався б один операторний блок, а в противному випадку — іншої. Синтаксис розширеного оператора умови наведений нижче:

```
if {умова) {
// Оператори сценарію, виконувані в тому випадку, якщо
// значення вираження умова дорівнює true.
}
 else {
// Оператори сценарію, виконувані в тому випадку, якщо // значення вираження умова дорівнює false.
}
```

Якщо після else знаходиться один оператор, то фігурні дужки можна опустити.
Оператор вибору switch дозволяє програмі обчислювати ви-раз і співставляти значення виразу зі значенням в мітці case. Якщо збіг знайдений, програма виконує код написаний в цій мітці, інакше виконується мітка default. Синтаксис оператора switch виглядає так:

```
switch (expression){
case label :
statement;
break;
case label :
statement;
break;
...
default: statement;
}
```

Програма спочатку шукає label, співпадаючий із значенням expression, а потім виконує statement/оператор. Якщо співпадаючий label не знайдений, програма шукає необов’язковий default statement/оператор за замовчуванням і, якщо він знайдений, виконує його. Якщо default statement не знайдений, програма продовжує вико-нувати наступний оператор, після кінця оператора switch.
Необов’язковий оператор break, що асоціюється з міткою case label, гарантує, що програма перерве виконання оператора switch, як тільки буде виконаний

оператор знайденого збігу, і продовжить виконання з оператора, що йде після оператора switch. Якщо break відсутній, програма продовжує виконання наступного оператора усередині оператора switch.

8 Оператори циклу

Оператор циклу служать для багаторазового виконання операторного блоку, що є наступним за ним. В JavaScript є три оператори циклу, почнемо з найбільше широко використовуваного.

Оператори for

Синтаксис оператора циклу for має вигляд:

for (початкове_значення_лічильника;
умова_виконання;
зміна_значення_лічильника) {
Операторний блок, виконуваний у циклі }

Передбачається, що в циклі використається лічильник, лічильником може бути будь-яка змінна, у тому числі оголошена в операторі циклу. Умова_виконання — логічний вираз, що управляє виконанням циклу, цикл виконується поки вираз умова_виконання істинний.

Зверніть увагу, що елементи оператора for відділені один від одного символом ";".

Наприклад, оператор for (i = l; i<10; i + +) починає виконання при значенні змінної циклу i, рівному 1, і до виходу з циклу дев'ять разів повторює наступну за ним послідовність операторів JavaScript.

Наприклад:

```
for (i = l; i<10; i + +)
document.write (i);
```

У результаті браузер відобразить результат виконання кожного кроку циклу:

1; 2; 3;4; 5; 6; 7; 8; 9; 10

Оператор циклу for…in

Даний оператор дозволяє організувати цикл по іменах і зна-ченням властивостей об'єктів. Властивість являє собою іменоване зна-чення, що належить об'єкту. JavaScript дозволяє одержувати доступ й/або змінювати властивості об'єктів.

Синтаксис оператора for. . .in наступний:

for *(змінна* in *об'єкт*){
. . .
значення_властивості_об'єкта = об'єкт[змінна]
. . .
}

де індексна *змінна* пробігає всі значення імен властивостей *об'єкта,* а за допомогою оператора в операторному блоці циклу можна одержати доступ до значень властивостей.

Щоб розібратися з тим, як працює цикл for... in створимо функцію objectContent (), що буде повертати рядок з іменами, значеннями й типами властивостей об'єкта, що є єдиним параметром функції.

```
function objectContent(Obj){
var s = ""
for (var i in Obj)
s += i + '= ' + // Ім'я властивості
Obj[i] + // Значення властивості
'; typeof=' + // Тип значення властивості
typeof(Obj[i])+ // обчислюється за допомогою функції
'\n' // typeof()
return s;
}
```

Дану функцію зручно використовувати при налагоджені користувацьких об’єктів.

9 Оператор while

Третій оператор циклу while має наступний синтаксис:

```
while(умова) {
Операторний блок
}
```

Операторний блок виконується поки значення умови дорівнює true. Як приклад розглянемо наступний фрагмент:

```
var i =0 while (i<10){
i+ + ;
}
```

Тіло циклу буде виконано десять разів і значення змінної і після виконання циклу буде дорівнює 10. Якщо в умові використається значення лічильника циклу, то на відміну від оператора for значення лічильника повинне змінюватися в тілі циклу, як показано в даному прикладі.

10 Оператори break і continue

Оператор break служить для переривання виконання циклу й використовується звичайно разом з умовним оператором. Розглянемо наступний приклад:

```
var i, j=l, k=l;
for (i=l; i<=10; i++) {
```

```
k++
if (i==5) break;
j++
}
// Після того, як лічильник циклу одержить значення,
// рівне п'яти, керування буде передано функції alert (...) ,
// наступної за даним, рядком
alert('i=' + i + . ' j =' + j+ ' k=' + k);
```

Після виконання даного фрагмента сценарію значення і й j рівні 5, а k одержить значення, рівне 6.

Оператор continue використовується для пропуску операторів у тілі циклу, що випливають за ним, після чого керування передається операторові циклу для переходу до наступної ітерації. Розглянемо наступний приклад:

```
var l=1, m = 1; while (l<=10) {
l + +;
if (l>=5) continue;
m+ + ;
}
alert(' l='+l+' m='+m)
```

Після виходу із циклу значення l дорівнює 11, а значення m дорівнює 4. Дійсно оператор m++ буде виконаний тільки три рази, на четвертій ітерації значення l стане рівним п'яти й завдяки виконанню оператора continue, збільшення значення m (оператор m++) більше не буде здійснюватися. Початкова частина циклу (оператор l++) буде виконана 10 разів, при цьому l одержить значення 11, оператор continue передасть керування оператору while, значення виразу-умови якого 11<=10 хибне, що у свою чергу приведе до завершення виконання циклу.

11 Оператор with

Цей оператор спрощує роботу із властивостями й методами об'єктів. Справа в тому, що доступ до об'єкта здійснюється так: *ім'я_об'єкта.ім'я_властивості.* Якщо потрібно багаторазово звертати-ся до властивостей об'єкта, то стільки ж раз потрібно вказувати ім'я об'єкта. Оператор with дозволяє зробити це однократно. Пояснимо це на прикладі посилань на об’єкт Math під час обчислень:

```
with (Math) {
a = PI * r*r
x = r * cos(theta)
y = r * sin(theta)
}
```

Порядок виконання роботи:

Тренувальні завдання

1. Створіть сценарій зміни кольору фону вікна браузера при використані обробника подій у відкриваючому тезі **<INPUT>.** Для вирішення завдання створіть форму з чотирма кнопками, кожна з яких після клацання по ним змінює поточний колір на червоний, синій, жовтий або білий відповідно. Зміна кольору фону здійснюється привласненням кодової назви кольору властивості **bgColor** об'єкта **document**.

Запустіть PHP Storm і створіть новий файл у ньому файл **TestJS.html** , в якому введіть наступний сценарій JavaScript:

Лістинг 1– Зміна кольору вікна браузера

```
1.<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2.<HTML>
3.<HEAD><TITLE>Background</TITLE></HEAD>
4.<BODY>
5.<FORM name="frm">
6.<INPUT type="button" value=" red "
7.onclick="document.bgColor='red'">
8.<INPUT type="button" value="blue"
9.onclick="document.bgColor='blue'">
10.<INPUT type="button" value="green"
11.onclick="document.bgColor='green'">
12.<INPUT type="button" value="white"
13.onclick="document.bgColor='white'">
14.</FORM>
15.</BODY>
16.</HTML>
```

Запустіть браузер. Відкрийте у вікні браузера створений файл і пересвідчиться в тому, що з'являється вікно із відповідним змістом і перевірте, як працюють кнопки.

Рис. 1.– Зовнішній вигляд форми приклада із зміною кольору фону вікна браузера.

2. Створіть сценарій, який при завантажені і закриті HTML-документа видає вікна повідомлень із відповідними текстами «Привіт» і «До побачення».

У попередньому документі видаліть непотрібний текст та помістіть туди лістинг 2.

Лістинг 2 – Функція alert і обробники подій onload і onunload

```
1.<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2.<HTML>
3.<HEAD><TITLE>Alert</TITLE></HEAD>
4.<BODY onload="javascript:alert('Привіт!');"
5.onunload="alert('До побачення!')">
6.Звичайний текст HTML-документа
7.</BODY>
8.</HTML>
```

Запустіть браузер. Відкрийте у вікні браузера створений файл і пересвідчиться в тому, що скрипт працює.

3. За допомогою конструкції document.write (метод write JavaScript-об’єкта document) здійсніть динамічну верстку тіла HTML-документа відповідно до рис. 2.

Рис. 2 – Електронна візитна картка із рамкою і логотипом

У будь-якому графічному редакторі створіть логотип візитної картки і збережіть його у папці **images** під ім’ям **visit_card.gif**.

У нашому документі введіть наступний сценарій JavaScript (лістинг 3):

Лістинг 3 – Динамічна верстка електронної візитної картки

```
1. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2. <HTML>
3. <HEAD><TITLE>Visit card</TITLE></HEAD>
4. <BODY>
5. <SCRIPT type="text/javascript" language="JavaScript">
6. <!--
7. document.write('<TABLE width="200" border="1" align="CENTER">');
8. document.write('<TR><TD align="CENTER">');
9. document.write(' <TABLE width="100%" border="0">');
```

```
10. document.write(' <TR>');
11. document.write(' <TD align="CENTER" valign="CENTER">');
12. document.write(' <IMG src="images/visit_card.gif">');
13. document.write(' </TD>');
14. document.write(' <TD align="CENTER">');
15. document.write(' <B>Pavlo Bondar</B><BR>');
16. document.write(' <I>computer engineering</I><BR>');
17. document.write(' <FONT color="blue">8-067-888-88-88</FONT>');
18. document.write(' </TD>');
19. document.write(' </TR>');
20. document.write(' </TABLE>');
21. document.write('</TD></TR>');
22. document.write('</TABLE>');
23. //-->
24. </SCRIPT>
25. </BODY>
26. </HTML>
```

Запустіть браузер. Відкрийте у вікні браузера створений файл і пересвідчиться в тому, що візитна картка генерується.

4. Створіть сценарій, що обчислює час обробки документа із використанням всіх можливих способів впровадження коду сценарію JavaScript. Сценарій повинен реалізовувати обчислення часу обробки браузером документа HTML і відображати цю інформацію у діалоговому вікні. Один із варіантів роботи сценарію показано нижче на рис.3

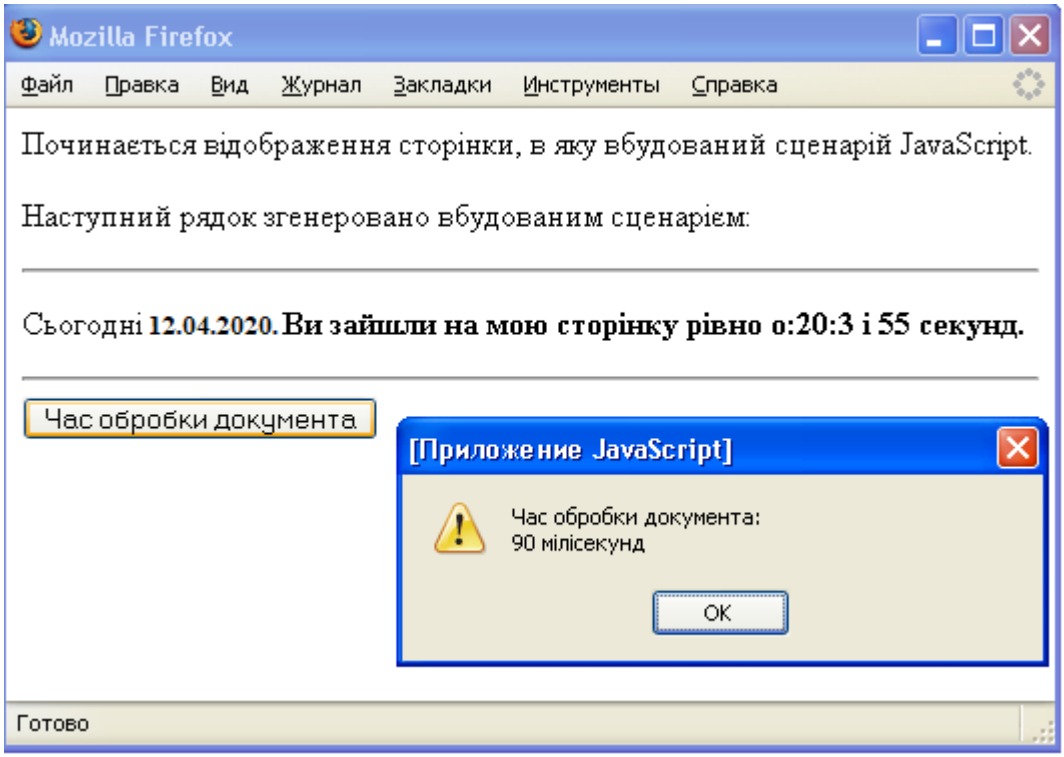


Рис.3 - Зовнішній вигляд сторінки що обчислює час обробки документа

Введіть текст HTML-документа, який реалізує необхідний сценарій JavaScript (лістинг 4.1).

Лістинг 4.1 – Обчислення часу обробки документа

```
1.<HTML>
2.<HEAD>
3.<SCRIPT LANGUAGE="JavaScript">
4.<!--
5.// Оголошення змінної tBegin та її ініціалізація
6.// об'єктом Date, який містить поточну дату і час
7.var tBegin = new Date();
8.//-->
9.</SCRIPT>
10.<SCRIPT SRC=time.js LANGUAGE="JavaScript"></SCRIPT>
11.</HEAD>
12.<BODY onload="tEnd = new Date();">
13.<P>Починається відображення сторінки, в яку вбудований
14.сценарій JavaScript.</P>
15.<p>Наступний рядок згенеровано вбудованим сценарієм:</p>
16.<hr>
17.<SCRIPT>
18.<!--
19.document.write("<p>Сьогодні <b>", time(),"</b></p>");
```

```
20.//-->
21.</SCRIPT>
22.<hr>
23.<input type=button
24.onclick="delta = tEnd.getTime()-tBegin.getTime();
25.s = 'Час обробки документа:\n'+delta+
26.' мілісекунд';
27.alert(s)"
28.value="Час обробки документа">
29.</BODY>
30.</HTML>
```

Для обчислення загального часу обробки документа в сценарії, розташованому в елементі HEAD, задається змінна tBegin, у якій зберігається час початку обробки документа (без урахування часу, що пішов на обробку тегу <HEAD> і самого тегу <SCRIPT>).

Другий елемент **SCRIPT** підключає файл **time.js**, у якому перебуває визначення функції **time(),** що повертає поточну дату. Ця функція викликається в сценарії третього елемента SCRIPT, розташованого в тілі документа HTML (елемент BODY). У ньому для відображення на сторінці отриманою функцією time() дати використовується метод write() об'єкта document браузера. Цей об'єкт представляє всю заван-тажену в браузер сторінку, і в момент її формування зазначеним методом можна "писати" на сторінку все, що потрібно. Однак після завершення обробки документа (при цьому генерується подія load об'єкта document) запис цим методом у документ приведе до того, що в ньому буде втримуватися тільки те, що записано цим методом.
Для визначення часу завершення обробки документа огляда-чем використається згадувана подія load й в оброблювачі події onload елемента BODY створюється змінна tEnd, у якій і зберігається час завершення обробки документа.
Для одержання загального часу обробки документа по кліку на кнопці виконується код, визначений у її оброблювачі події onclick, що обчислює різницю між часом, що зберігається в змінних tBegin й tEnd і відображає результат функцією alert().

Збережіть створений документ під ім’ям **time.html** у папці зі своїм проектом даної практичної роботи.

За допомогою PHP Storm створіть файл **time.js**, який розташовується в тім же каталозі, що й файл **time.html**. Цей файл містить код JavaScript, що визначає функцію одержання поточної дати у формі: число.місяць.рік. Код файлу наведено нижче у лістингу 4.2:

Лістинг 4.2 – Зміст файлу time.js, що визначає поточну дату

```
1. function time()
2. {
3. Now = new Date();
4. var s, s1, s2, s3, s4, s5, s6;
5. s1=Now.getDate();
6. s2=parseInt(Now.getMonth())+1;
7. s3=Now.getFullYear();
8. s4=Now.getHours();
9. s5=Now.getMinutes();
10. s6=Now.getSeconds();
11. s=s1+"-"+s2+"-"+s3+". Ви зайшли на мою сторінку рівно о:"
12. +s4+":"+s5+" і "+s6+" секунд."
13. return s
14. }
```

Запустіть браузер, відкрийте файл **time.html** і перевірте чи працює обробник подій.

5. Напишіть сценарій JavaScript, що створює бланк відповіді студента за зразком (див. рис.4) із використанням користувацької функції без параметрів blank( ). Для цього виконайте наступні дії.

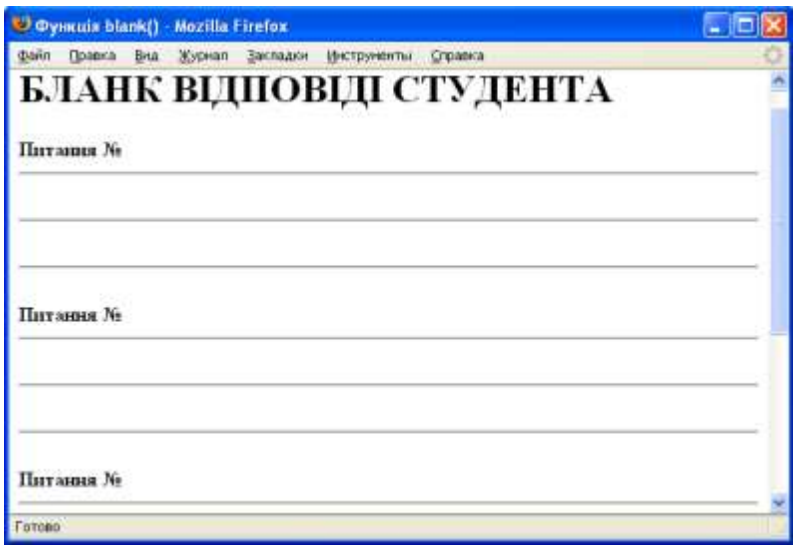


Рис.4 – Результат виконання функції blank()

Відкрийте документ **TestJS.html**
У відкритому файлі зробіть зміни відповідно до лістингу 5 наведеному нижче:
Лістинг 5 – Сценарій javascript із «пустою» функцією blank()

```
1.<HTML>
2.<HEAD>
3.<TITLE>Функція blank()</TITLE>
4.<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
5.CHARSET=windows-1251">
6.<SCRIPT LANGUAGE="JavaScript">
7.function blank(){
8.//тут має бути опис функції
9.}
10.</SCRIPT>
11.</HEAD>
12.<BODY>
13.<h1>БЛАНК ВІДПОВІДІ СТУДЕНТА</h1>
14.<SCRIPT type="text/javascript" language="JavaScript">
15.<!--
16.blank();
17.blank();
18.blank();
19.blank();
20.//-->
21.</SCRIPT>
22.</BODY>
23.</HTML>
```

Запустіть браузер і пересвідчиться у тому, що виводиться тільки заголовок першого рівня **«БЛАНК ВІДПОВІДІ СТУДЕНТА».**
Для того, щоб запрацювала функція blank(), завданням якої є виведення тексту, наведеному на рис.4, необхідно у рядок 8 записати тіло функції. Один із можливих варіантів запису тіла функції може бути таким:

```
document.write("<b>Питання №</b><br><hr><br><hr><br><hr><br>")
```

Відкрийте у вікні браузера файл із збереженими змінами і переконайтеся в тому, що функція blank(), працює.

Зробіть невеличке дослідження, помістивши функцію blank() (рядки 7-9), у середину скрипта, що знаходиться в тілі документа: перед викликом функції (перед рядком 16), а потім після її виклику (після рядка 19). Зробіть висновки.

6. У наведеному нижче сценарії JavaScript (лістинг 6) зробіть зміни так, щоб функція citata() виконувалася.

Лістинг 6 – Сценарій із помилкою виконання функції citata()

```
1.<HTML>
2.<BODY>
3.<SCRIPT>
4.<!--
5.function citata() {
6.document.write("\”Найкраща помилка та, якої допускаються у навчанні.\”(Г. Сковорода)");
7.}
8.//-->
9.</SCRIPT>
10.</BODY>
11.</HTML>
```

Для цього необхідно виконати наступні дії.

У результаті виконання коду файла у вікні браузера нічого не відображається тому, що оператори функції citata() не виконуються, оскільки до них немає звертання. Відкрийте файл у редакторі і переробіть програму таким чином, щоб у вікні браузера з’явилось таке речення:

”Найкраща помилка та, якої допускаються у навчанні.” (Г. Сковорода)

Переконайтесь, що функції citata() працює.

7.Створіть Web-сторінку, яка містить три HTML-посилання, що передають текстові значення. Інформація передається у функцію say(message), що виводить передані дані за допомогою методу alert().

Зразок Web-сторінки показано на рис.5.

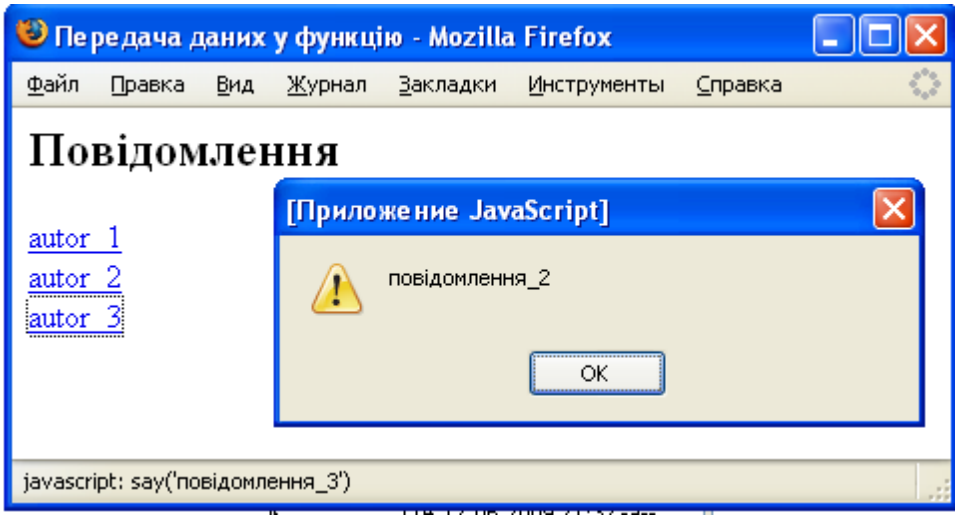


Рис. 5 – Результат передачі даних у функцію say(message)

Для виконання завдання необхідно створити два скрипти: перший у голові Web-сторінки із описом функції say(message), а другий у тілі документа із викликом функції по HTML-посиланню.

Код першого скрипта може мати наступний вигляд:

```
<SCRIPT LANGUAGE="JavaScript">
<!--
function say(message){
alert(message)
}
//-->
</SCRIPT>
```

Для виклику функції по HTML- посиланню можна скористатися наступним прикладом:

```
<a href="javascript: say('повідомлення_1')">autor_1</a><br>
```

Перевірте правильність роботи сценарію у вікні браузера.

8. У наведеному коді (лістинг 7) обчислення суми трьох чисел здійснюється функцією із параметрами sum(x,y,z). Обчислення значення передається в основну програму за допомогою оператор return. На основі коду обчислення суми трьох чисел, створити власний сценарій, який додатково обчислює середнє значення і добуток трьох чисел. Результат роботи сценарію показано на рис.6.

Лістинг 7 –Обчислення суми трьох чисел (функція sum())

```
<HTML>
<BODY>
<SCRIPT>
<!--
```

```
function sum (x,y,z) {
d=x+y+x;
document.write("Сума від "+x+", "+y+", "+z+" дорівнює - ");
return d
}
sum (1,3,5)
document.write (d)
//-->
</SCRIPT>
</BODY>
</HTML>
```

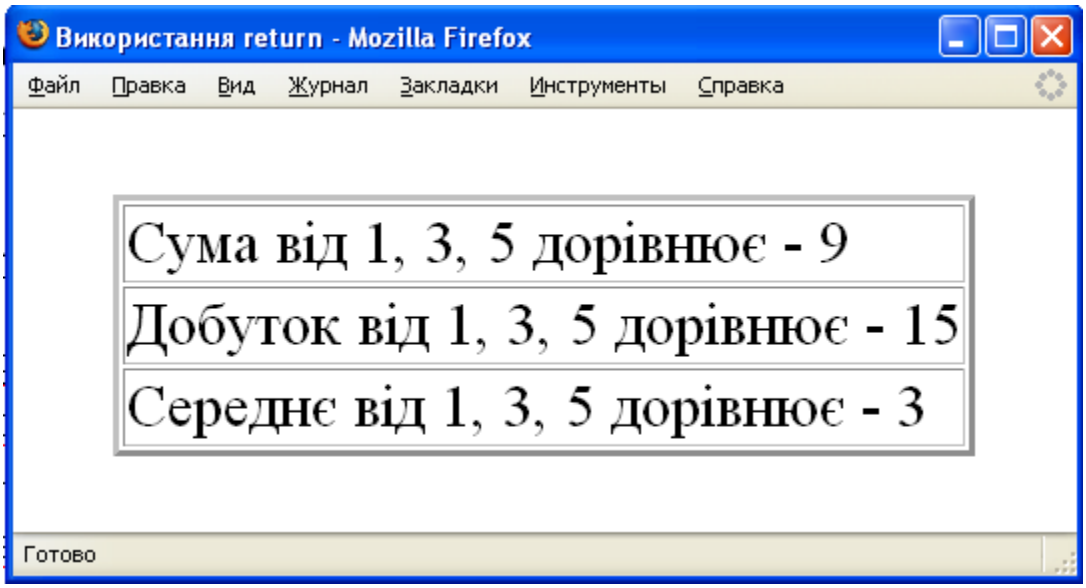


Рис. 6 – Результат передачі даних у функцію say(message)

У ваш документ **TestJS.html** введіть сценарій відповідно до лістингу 7. Добавте дві функції function dobutok (x,y,z) і function sered (x,y,z), які проводять обчислення відповідно добутку і середнього значення. Не забудьте зробити звертання до цих функцій.

9. За допомогою конструкції if створіть сценарій, який поточний час переводить у дванадцятигодинний формат.

У вашому документі **TestJS.html** введіть наступний сценарій JavaScript:

Лістинг 8 – Поточний час у дванадцятигодинному форматі

```
1.<!DOCTYPE HTML>
2.<HTML>
3.<HEAD><TITLE>Поточний час</TITLE></HEAD>
4.<BODY>
5.<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
```

```
6.var today = new Date()
7.var hours = today.getHours()
8.var minute = today.getMinutes()
9.if (minute < 10) minute = '0'+minute
10.if (hours < 12) {var time12 = hours + ':' + minute + ' am'}
11.else {var time12 = hours - 12 + ':' + minute + ' pm'}
12.document.write('поточний час - ', time12)
13.</SCRIPT>
14.</BODY>
15.</HTML>
16.<HTML>
```

Відкрийте у вікні браузера створений файл і пересвідчиться в тому, що скрипт працює (див. рис. 7).

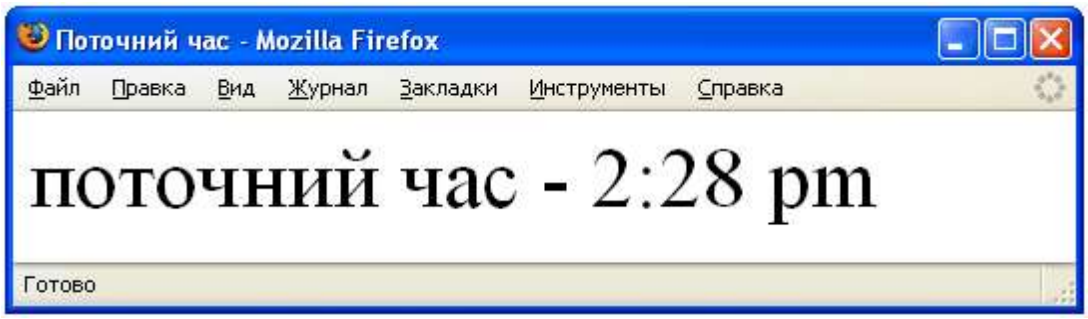


Рис. 7 – Результат виконання лістинга 8 у вікні браузера.

Отримайте той же результат із використанням тернарного оператора, змінивши рядки 10-11 лістингу 8. Приклад синтаксису тернарного оператора, що виводить години без урахування хвилин показано нижче:

var hours= (hours<12)? hours +" am": hours-12+" pm"

10. За допомогою конструкції switch створіть сценарій правильного відмінювання слова —рік. Результат виконання завдання показано на рис.8.

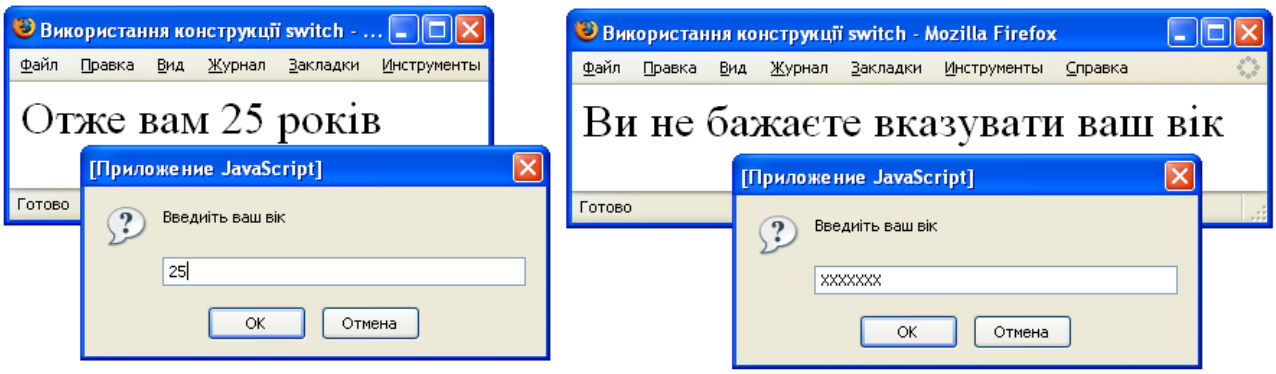


Рис. 8 – Результат сценарію відмінювання слова «рік».

Лістинг 9 – Вибір правильного відмінювання слова “рік”

```
<!DOCTYPE HTML>
<HEAD>
<TITLE>Використання конструкції switch</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
var age = parseInt(prompt('Введіть ваш вік', ''))
if (!isNaN(age)) {
var last = age%10
var last2 = age%100
var def = (last>1)+(last>4)+(last==0)*2+ (last2>10)*(last2<20)*2
switch (def) {
case(0):
var text = " рік"
case(1):
var text = " роки"
default:
var text = " років"
}
document.write("Отже вам ", age, text)
} else {
document.write("Ви не бажаєте вказувати ваш вік ")
}
} </SCRIPT>
</BODY>
</HTML>
```

11. За допомогою вбудованої функція eval() напишіть сценарій, який виконує прості арифметичні обчислення (див. рис.9).

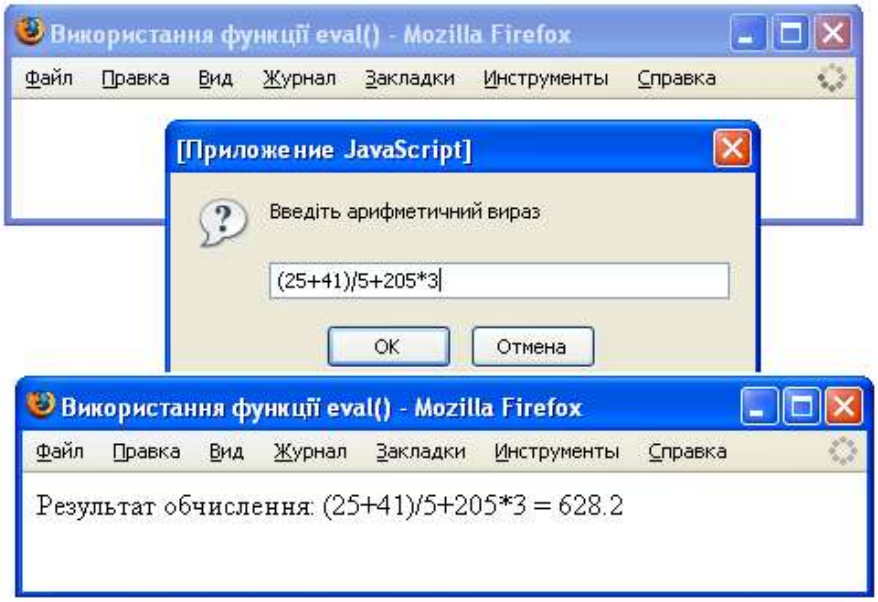


Рисунок 9- Використання функції eval()

Функція eval() є дуже корисною. Вона призначена для обчислення значення рядка символів, що передаються в якості аргумента. Вона може містити JavaScript-код без прямого зв'язку із основним кодом JavaScript-програми. У лістингу 10 показано, як вирішується поставлене завдання.

Лістинг 10 – Використання функції eval()

```
<HTML>
<HEAD>
<TITLE>Використання функції eval()</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
var text = prompt('Введіть арифметичний вираз', '')
var res = eval(text)
document.write ('Результат обчислення: ', text, ' = ', res)
</SCRIPT>
</BODY>
</HTML>
```

Протестуйте програму, ввівши різні арифметичні вирази, наприклад: [(258-32)+(45+2/3)]/100.

Самостійна робота студента

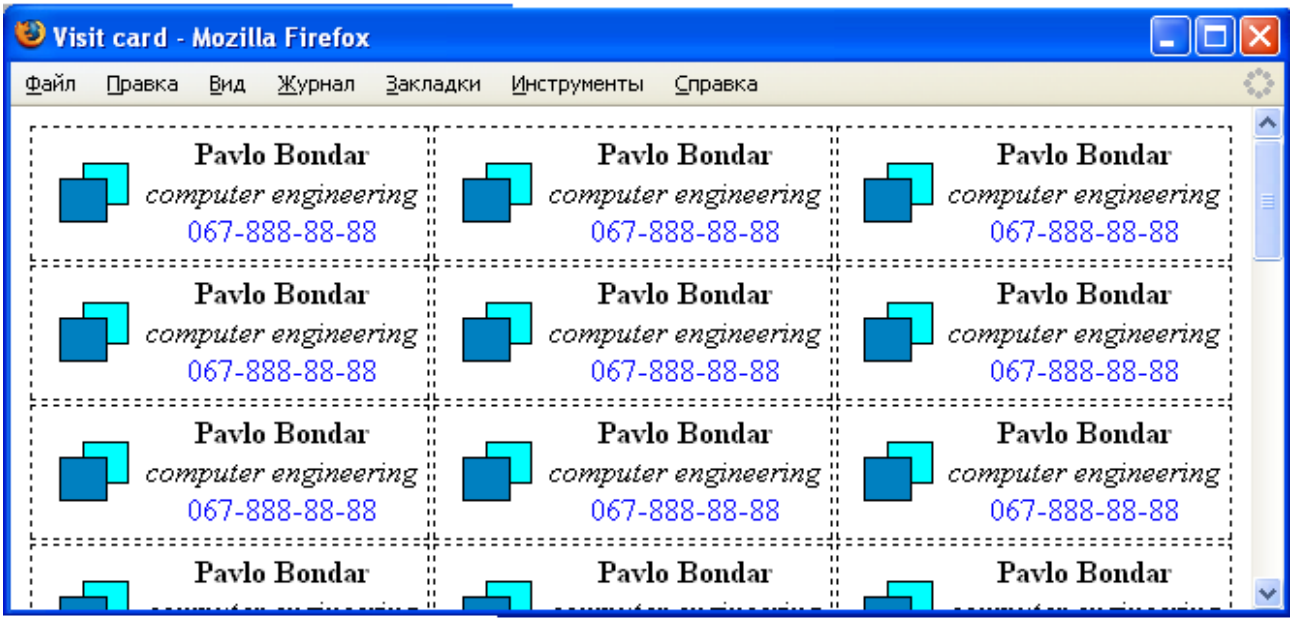
Звіт до практичної роботи №5 має включати виконані 4 задачі, тобто код кожного із 4-х завдань та скрін-шотами їх виконання у браузері.

Завдання 1.

У завданні 3 тренувальних завдань було розглянуто створення електронної візитної картки, яка тільки одна відтворювалася у вікні браузера (див. рис. 2). Але, для фінального друку візитних карток, доцільно сформувати цілий аркуш, де б вони розташовувалися у вигляді таблиці із декількох рядків і стовпчиків, як показано на рисунку.

Отже, напишіть сценарій, що формує у вікні браузера аркуш з візитними картками у вигляді таблиці розміром 12×3. Передбачити запит у користувача особистих даних (ім'я, професію, телефон) за допомогою функції prompt().

(Використайте введення ваших особистих даних та сформуйте власні візитівки)



Формування аркуша із візитними картками у вікні браузера.

Завдання виконуємо у наступній послідовності.

У PHP Storm відкрийте шаблон для створення JavaScript сценарію (проект створений для тренувальних вправ). Сценарій буде розташовуватися між тегами <BODY>…</BODY>.

Створіть першу частину сценарію, в якій створюються змінні із значеннями особистих даних користувача, наприклад:

var name = prompt('Input name','');
var position = prompt('Input position','');
var phone = prompt('Input phone','');

Наступним кроком є створення змінної var visitcard, в якій збе-рігається 1 примірник візитної картки (інформація про користувача і логотип) у вигляді комірки таблиці. Логотип візитної картки можна створити заново в любому графічному редакторі або скористатися го-товим із попередньої лабораторної роботи (папка **images,** ім'я **visit_card.gif**). Наприклад:

```
var visitcard = ' <TABLE width="100%" border="0">'
+' <TR>'
+' <TD align="CENTER" valign="CENTER">'
+' <IMG src="images/visit_card.gif">'
+' </TD>'
+' <TD align="CENTER">'
+' <B>' + name + '</B><BR>'
+' <I>' + position + '</I><BR>'
+' <FONT color="blue">' + phone + '</FONT>'
+' </TD>'
+' </TR>'
+' </TABLE>';
```

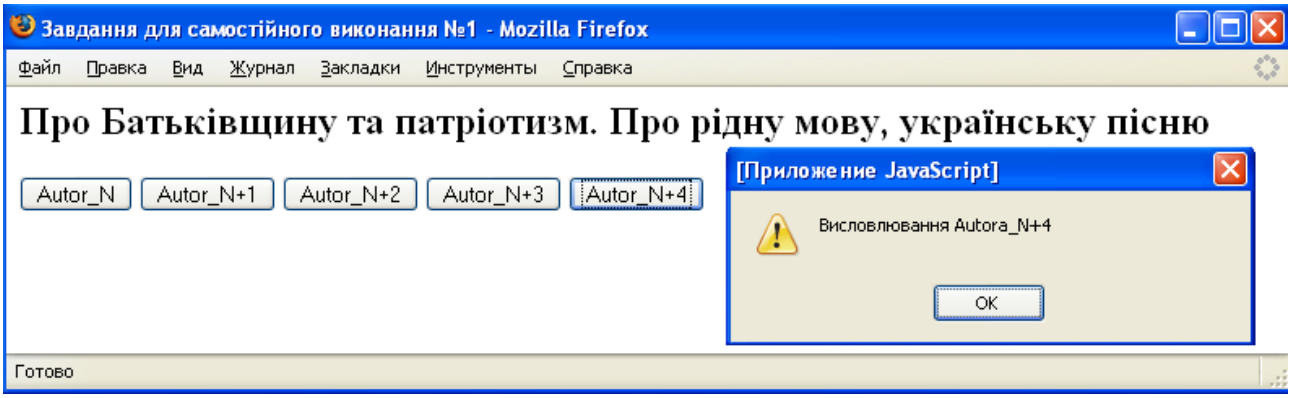
Завершується сценарій циклом, який створює таблицю 12×3.

```
document.write('<TABLE width="100%" border="0" align="CENTER">');
for(var y=0; y<12; y++)
{
document.write('<TR>');
for(var x=0; x<3; x++)
{
document.write( '<TD align="CENTER" style="border-style: dashed; border-width:thin;">');
document.write( visitcard );
document.write('</TD>');
}
document.write('</TR>');
}
document.write('</TABLE>');
```

Завдання 2.

Напишіть скрипт, який при натисканні на кнопку із прізвищем автора, видає його вислів. Приклад виконання завдання показано на рисунку. Значення N вибирають із таблиці за вашим номером варіанту(порядковий номер у списку групи)

Наприклад, якщо порядковий номер студента у списку групи дорівнює 10, студент створює п’ять кнопок із прізвищами (10 - Ч. Айтматов, 10+1=11 - П. Гольбах, 10+2=12 - Т. Шевченко, 10+3=13 Д. Павличко -, 10+4=14 - Л. Костенко)



Приклад виконання самостійного завдання №2

Таблиця– Вихідні дані для виконання самостійного завдання №2

№	Автор вислову	Зміст вислову
1	Сенека	Люблять батьківщину не за те, що вона велика, а за те, що своя
2	Овідій	І дим батьківщини солодкий.
3	Д. Байрон	Той, хто не любить своєї країни, нічого любити не може
4	Г. Сковорода	Кожному мила своя сторона.
5	Д. Лихачев	Усвідомлена любов до свого народу не поєднується з ненавистю до інших.
6	Горацій	Ті, що виїздять за море, міняють небо, а не душу.
7	Цицерон	Для нас дорогі батьки, дорогі діти, близькі, родичі; але всі уявлення про любов до чого-небудь поєднані в одному слові «вітчизна».
8	М. Шумило	Мова — це глибина тисячоліть.
9	Ч. Айтматов	Той, хто в біді кидає свій народ, стає його ворогом.
10	П. Гольбах	Де немає свободи, там немає і вітчизни.
11	Т. Шевченко	В своїй хаті своя й правда, І сила, і воля.
12	Д. Павличко	Вітчизна — ось і альфа, і омега!
13	Л. Костенко	Нації вмирають не від інфаркту. Спочатку їм відбирає

		мову.
14	Т.Шевченко	Нема на світі України, немає другого Дніпра
15	П. Чубинський	Ще не вмерла України і слава, і воля.
16	Л. Толстой	Я дуже люблю народну українсь-ку мову, звучну, барвисту й таку м'я-ку.
17	В. Голобородько	Мова вмирає, коли наступне покоління втрачає розуміння значення слів.
18	М. Гоголь	М. Гоголь Дивуєшся дорогоцінності мови нашої: в ній що не звук, то подарунок, все крупно, зернисто, як самі перла.
19	Е. Челебі	Українці — стародавній народ, а мова "їхня мова багатша і всеосяжніша, ніж персидська, китайська, монгольська і всілякі інші.
20	О. Пахльовська	Раби — це нація, котра не має Слова. Тому й не зможе захистить себе.
21	І. Репін	Відчуваю й усвідомлюю, яка це красива й легка українська мова.
22	Олександр Олесь	Для всіх ти мертва і смішна, Для всіх ти бідна і нещасна, Моя Україно пре-красна, Пісень і волі сторона.
23	Л. Толстой	Л. Толстой Слово є вчинок.
24	А. Міцкевич	Народна пісня-духовне обличчя нації
25	О. Довженко	Українська пісня — це бездонна душа українського народу, це його слава.
26	П. Тичина	Я єсть народ, якого Правди сила ніким звойована ще не була.
27	А. Франс	Немає магії сильнішої, ніж магія слів.
28	Старий Заповіт	Споконвіку було Слово.
29	Володимир Мономах	Коли щось умієте, того не забувайте, а чого не вмієте — то того учітесь...
30	Г. Сковорода	Безперервно думай, щоб пізнати себе
31	Т. Шевченко	Борітеся — поборете!

Завдання 3

Створити скрипт «Математичні функції й константи», що виконує математичні операції. Скрипт використовує об'єкт Math з різними властивостями й методами. Вісім властивостей дозволяють сформувати вісім констант (у тому числі постійну Ейлера e й число π). Шістнадцять методів об'єкта Math призначені для формування шістнадцяти елементарних математичних функцій. Web-сторінка має виводити інформацію у порядку, що наведений нижче, а саме:

Всі вісім констант для всіх варіантів, п’ять функцій по номеру варіанту та генератор випадкових чисел.

Значення N відповідає порядковому номеру у списку групи і наступних 4 функції по порядку.

Наприклад, якщо порядковий номер студента у списку групи дорівнює 10, тоді функції, які реалізовує студент – під номером 10, 11, 12, 13 та14.

КОНСТАНТИ

1. Значення константи e = 2.718281828459045
2. Значення константи π = 3.141592653589793
3. Десятковий логарифм постійної Ейлера (e) = 0.4342944819032518
4. Двійковий логарифм постійної Ейлера (e) = 1.4426950408889634
5. Натуральний логарифм числа 10 = 2.302585092994046
6. Натуральний логарифм числа 2 = 0.6931471805599453
7. Корінь квадратний із двох = 1.4142135623730951
8. Корінь квадратний з однієї другої = 0.7071067811865476

ФУНКЦІЇ

1. Результат зведення в степінь: $4^2 = 16$
2. Результат добування квадратного кореня з 4: y = 2
3. Результат добування квадратного кореня з -9: y = NaN
4. Результат обчислення модуля |-9|: y = 9
5. Результат обчислення косинуса нуля радіан: y =1
6. Косинус 180 градусів (π радіан): y = -1
7. Результат обчислення арккосинуса нуля : y =1.5707963267948966
8. Результат обчислення синуса нуля радіан: y = 0
9. Синус 90 градусів (π /2 радіан): y = 1
10. Результат обчислення арксинуса одиниці: y = 1.5707963267948966
11. Тангенс 45 градусів (π /4 радіан): y = 0.9999999999999999
12. Результат обчислення арктангенса одиниці: y = 0.7853981633974483
13. Результат зведення e в степінь 2: y = 7.38905609893065
14. Натуральний логарифм від e^2: y = 2
15. Результат обчислення десяткового логарифма від числа 100: y = 2
16. Результат обчислення двійкового логарифма від числа 8: y = 3
17. Тангенс 45 градусів (з округленням): y = 1
18. Результат округлення числа 1,00123 у більшу сторону: y = 2
19. Результат округлення числа -3.1234 у більшу сторону: y = -3
20. Результат округлення числа 1,00123 у меншу сторону: y = 1
21. Результат округлення числа -3.1234 у меншу сторону: y = -4
22. Результат округлення числа 1,44 за правилами арифметики: y = 1
23. Результат округлення числа 1,54 за правилами арифметики: y = 2
24. Вибір найбільшого із двох чисел 4 й 2: y = 4
25. Вибір найменшого із двох чисел 4 й 2: y = 2

ГЕНЕРАТОР ВИПАДКОВИХ ЧИСЕЛ
Формування випадкових чисел в інтервалі [0;1]: y = 0.44051778077089376

Завдання 4

Створити скрипт «розгалужений обчислювальний процес», який виконує обчислення по формулах, що наведені у таблиці

У сценарію, що розробляється передбачити ввід із клавіатури меж зміни значень аргументу *x* та кроку зміни значень аргументу.

У звіті навести розроблену програму та результати розрахунків при зміні аргументу *x* від -6 до 6 із кроком 2 (тобто 7 значень функції при відповідних значеннях аргументу).

Номер варіанту обирається згідно номера у списку групи.

Таблиця– Вихідні дані для виконання самостійного завдання №4

Вар.	Математичний вираз	Вар.	Математичний вираз
1,17	$V=\begin{cases} x^2+e^x, & x<4 \\ \pi+2, & x=4 \\ \sin^2 x+\ln x, & x>4 \end{cases}$	9,25	$U=\begin{cases} \sin x+\operatorname{tg} x, & x\leqslant -1 \\ x^2+2\cdot x^3, & -1<x\leqslant 1 \\ \sqrt{x^3}-e^x, & x>1 \end{cases}$
2,18	$Z=\begin{cases} \sin x+\cos x, & x<2 \\ x^{4.1}, & x=2 \\ \ln x+\sqrt{x}, & x>2 \end{cases}$	10,26	$W=\begin{cases} \lvert x\rvert+\operatorname{tg} x, & x<-4 \\ 23, & x=-4 \\ \cos x+x, & x>-4 \end{cases}$
3,19	$W=\begin{cases} \cos^2 x, & x<2 \\ \operatorname{tg} x+x^{3.7}, & x=2 \\ \lvert 1-x^2\rvert, & x>2 \end{cases}$	11,27	$U=\begin{cases} \sin x+\operatorname{tg} x, & x\leqslant -3 \\ 2, & -3<x\leqslant 4 \\ \sqrt{x^3}-\cos x, & x>4 \end{cases}$
4,20	$Y=\begin{cases} e^3-\sin x, & x<0 \\ 89, & x=0 \\ \lg x+\sqrt{2\cdot x}, & x>0 \end{cases}$	12,28	$G=\begin{cases} \operatorname{tg} x-x^2, & x<4 \\ 35\cdot\pi, & x=4 \\ \ln x+3, & x>4 \end{cases}$

5,21	$F=\begin{cases}\lvert x\rvert + e^x, & x<4\\ \pi, & x=4\\ ctgx + \ln x, & x>4\end{cases}$	13,29	$H=\begin{cases}\lvert \sin x\rvert, & x<4\\ \sqrt{x-1}, & x=4\\ \lg x+1, & x>4\end{cases}$
6,22	$U=\begin{cases}\sin x - tgx, & x<0\\ 0, & x=0\\ ctg^3 x, & x>0\end{cases}$	14,30	$Z=\begin{cases}\sin 3x+1, & x<2\\ \cos x^2, & x=2\\ \lg x+\sqrt{x}, & x>2\end{cases}$
7,23	$F=\begin{cases}tg^3 x, & x<0\\ \pi^3, & x=0\\ \sqrt{x}-\lg x, & x>0\end{cases}$	15,31	$V=\begin{cases}\cos 4x-\pi, & x<2\\ \lvert ctg\, x\rvert^3, & x=2\\ 5+\sqrt[3]{x}, & x>2\end{cases}$
8,24	$V=\begin{cases}\cos 4x-5, & x<2\\ \lvert tg\, x\rvert^3, & x=2\\ 2+\sqrt[3]{x}, & x>2\end{cases}$	16,32	$U=\begin{cases}\sin x + tgx, & x<2\\ x^2+5\cdot x^3, & x=2\\ \sqrt{x^3}, & x>2\end{cases}$

Питання для самоперевірки

1. Як включити оператори JavaScript в документ HTML? Наведіть приклади.
2. Як встановити зв’язок HTML-документу із зовнішнім файлом JavaScript ? Наведіть приклади.
3. Яке призначення мають коментарі? Чим відрізняються од-норядкові коментарі від багаторядкових? Поясніть на прикладах.
4. Для чого призначені функції alert й prompt? Наведіть приклади.
5. Які математичні операції підтримуються в мові JavaScript?
6. Яка послідовність виконання математичних операцій у математичних виразах прийнята у мові JavaScript?
7. Які тригонометричні методи об’єкта Math ви знаєте і яке їхнє призначення?
8. Для чого призначені оператори for й if ?
9. Чим відрізняються конструкції оператора if й if ...else?
10. Чим відрізняються оператори if й switch?
11. Для чого призначене ключове слово default у конструкції оператора switch?

12. Для чого наприкінці рядка для кожного варіанта умови в операторі switch використовується ключове слово break? Що буде, якщо опустити це слово?
13. У чому полягає відмінність операторів for, while й do...while? Як коректно замінити оператор for оператором while?
14. Як замінити оператор switch комбінацією операторів if ...else?
15. Яке призначення функцій parseInt і parseFloat?