КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ

**МЕТОДИЧНІ ВКАЗІВКИ**

**ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ З ДИСЦИПЛІНИ «Web-технології та Web-дизайн» JavaScript та Angular**

для студентів освітньої програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки»

освітнього ступеня «Бакалавр»

Київ – 2020

2

**Укладачі:** Федусенко О.В., к.т.н, доцент

Гамоцька С.Л.,

Доманецька І.М., к.т.н, доцент

Красовська Г.В. ,к.т.н, доцент

Методичні вказівки до виконання лабораторних робіт з дисципліни «Web технології та Web-дизайн» для студентів освітньої програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» освітнього ступеня «Бакалавр» Укл. / О.В. Федусенко, С.Л. Гамоцька, І.М. Доманецька, Г.В. Красовська. – К.: Вид. ГЛІФ МЕДІА, 2020. – 58с.

Рекомендовано Вченою радою факультету інформаційних технологій Київського національного університету імені Тараса Шевченка

Методичні вказівки містять цикл лабораторних робіт з дисципліни «Web технології та Web-дизайн» з тем JavaScript та Angular та організаційно-методичні рекомендації до проведення лабораторних занять, варіанти індивідуальних завдань та вимоги до змісту та оформлення звітів.

Розроблено з урахуванням стандарту вищої освіти України зі спеціальності «Комп`ютерні науки».

3

ЗМІСТ

Вступ ...................................................................................................4 Вимоги щодо захисту та оформлення звітів...................................5 Лабораторна робота №1. Об‘єкти та масиви.................................7 Лабораторна робота №2. Основи моделі DOM та події..............19 Лабораторна робота №3. Модель DOM ........................................27 Лабораторна робота №4. Основи ООП у JavaScript ....................33 Лабораторна робота №5. Основи фреймворка Angular...............38 Рекомендована література ..............................................................58

4

**Вступ**

Дисципліна «Web-технології та Web-дизайн» відноситься до розряду обов'язкових дисциплін освітньої програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» освітнього ступеня «Бакалавр». Завдання дисципліни полягає в створенні та розвитку у фахівця з комп’ютерних наук здатності реалізувати багаторівневу обчислювальну модель на основі архітектури клієнт

сервер, включаючи бази даних, знань і сховища даних, виконувати розподілену обробку великих наборів даних на кластерах стандартних серверів для забезпечення обчислювальних потреб користувачів, у тому числі на хмарних сервісах [1].

Метою виконання лабораторних робіт є закріплення та поглиблення знань, отриманих в процесі вивчення лекційного матеріалу, формування практичних навичок з програмування на мові JavaScript та ознайомлення з одним з найпотужніших та сучасних JavaScript фреймворків Angular.

Методичні вказівки містять п‘ять лабораторних робіт, що повністю охоплюють модуль теоретичного курсу, якій присвячено вивченню скриптової мови програмування JavaScript та фреймворку Angular.

Кожна лабораторна робота є обов’язковою до виконання.

В методичних вказівках наведено як теоретичний матеріал, який необхідно використовувати при написанні лабораторних робіт, так і приклади виконання деяких завдань.

Для виконання лабораторних робіт рекомендується використовувати середовище розробки Microsoft Visual Studio Code.

5

**Вимоги щодо захисту та оформлення звітів**

Для одержання позитивної оцінки з кожної роботи студент повинен у терміни визначені викладачем та у відповідності зі всіма наведеними вимогами розробити програмне забезпечення, оформити звіт та захистити роботу.

Максимальна оцінка, яку студент потенційно може отримати за виконану лабораторну роботу, зменшується у відповідності до часу відставання і визначається викладачем.

Звіти з лабораторних робіт здаються на перевірку в електронному вигляді. До кожного звіту додаються коди розроблених програм та скріншоти браузера з розробленою веб-сторінкою.

Всі завдання повинні виконуватись студентами індивідуально і не містити ознак плагіату як в оформленому звіті, так і в розробленому програмному забезпеченні.

Захист лабораторної роботи включає демонстрацію на комп’ютері розробленого програмного забезпечення з виконанням всіх запропонованих викладачем тестів.

Структура типового звіту з лабораторної роботи:

1. Титульна сторінка.

2. Назва лабораторної роботи.

3. Мета роботи.

До кожного завдання із лабораторної роботи надаються:

4. Опис завдання.

5. Специфікація функцій у вигляді таблиці, в якій описана кожна з функцій, проаналізовано вхідні та вихідні дані. Приклад специфікації наведено в таблиці 1.1.

6. Структура програми, на якій відображено зв‘язок файлів застосунку та функцій між собою та порядок їх виклику. Приклад структурної схеми наведено на рисунку 1.4.

7. Скріншоти тексту розробленого програмного забезпечення з коментарями. Скрипти на мові JavaScript надаються обов‘язково, коди HTML- та

6

CSS-файлів обов‘язкові лише для першої лабораторної роботи. 8. Скріншоти веб-сторінок з інтерфейсом та результатами виконання розроблених скриптів.

9. Висновки, що відображають особисто отримані результати виконання роботи, їх критичний аналіз.

Правила оформлення звітів визначаються Національним стандартом України «Звіти у сфері науки і техніки. Структура та правила оформлювання. ДСТУ 3008:2015»[2].

7

**Лабораторна робота №1. Об‘єкти та масиви**

**Мета роботи** – розглянути можливості роботи з датою в JavaScript, використання масивів та циклів в JavaScript, перевірки значень в коді скрипта. Навчитися створювати і використовувати об’єкти користувача в JavaScript з використанням моделі DOM.

**Завдання:**

Розробити дизайн веб сторінки та написати JavaScript для вирішення поставленого завдання відповідно до варіанту.

Розроблена HTML-сторінка повинна відповідати таким вимогам: • таблиці стилів повинні розташовуватися в окремих файлах; • верстка повинна бути адаптивною;

• скріпти для виконання завдань повинні бути розміщені в окремих файлах.

HTML-сторінка повинна містити:

• "шапку", що містить ПІБ студента, номер групи і номер варіанту; • опис завдань відповідно до варіанту.

**Заборонено використання jQuery для вирішення поставлених в лабораторній роботі завдань.**

**Теоретичні відомості**

**1.1 Робота з датою**

Для роботи з датою використовується об‘єкт Date().

Конструктор об’єкту має такий формат:

<змінна\_дати> = new Date ({Рік}, {Місяць}, {Число} [, {Години} [, {Хвилини} [, {Секунди} [, {Мілісекунди}]]]]),

де new – ключове слово, яке використовується для ініціалізації об’єкту вказаного типу.

Якщо не вказаний жоден параметр, конструктор ініціалізує об’єкт поточною датою і часом.

Дати в JavaScript зберігаються в мілісекундах, починаючи з 00:00:00 1

8

січня 1970 року. Дати до цього моменту в скриптах використовувати не рекомендується. Це пов’язано з тим, що в сценаріях значно частіше приходиться мати справу з теперішнім і майбутнім часом, ніж з минулим. **1.2 Об‘єкти та масиви**

Об‘єкти у JavaScript виконують дві основні функції:

Перша - це асоціативний масив: структура, придатна для зберігання будь яких даних.

Друга - мовні можливості для об'єктно-орієнтованого програмування. **1.2.1 Асоціативні масиви**

Асоціативний масив - структура даних, в якій можна зберігати будь-які дані в форматі ключ-значення.

*Створення об'єктів*

Порожній об'єкт може бути створений одним з двох синтаксисів: 1. o = new Object ();

2. o = {}; // порожні фігурні дужки

*Операції з об'єктом*

Об'єкт може містити в собі будь-які значення, які називаються властивостями об'єкта.

Доступ до властивостей здійснюється за ім'ям властивості («по ключу»). Наприклад, створимо об'єкт person для зберігання інформації про людину: var person = {}; // поки порожній

Основні операції з об'єктами - це створення, отримання та видалення властивостей.

Для звернення до властивостей використовується запис «через крапку». person.name = 'Вася';

person.age = 25;

Щоб прочитати їх - також звернемося через крапку:

alert (person.name + ':' + person.age); // "Вася: 25“

Видалення здійснюється оператором *delete*:

delete person.age;

9

Залишилося тільки властивість name:

Іноді буває потрібно перевірити, чи є в об'єкті властивість з певним ключем.

Для цього є особливий оператор *in*.

if ( "name" in person)

{

alert ( "Властивість name існує!");

}

Втім, частіше використовується інший спосіб - порівняння значення з *undefined.*

Справа в тому, що в JavaScript можна звернутися до будь-якої властивості об'єкта, навіть якщо її немає. Помилки не буде. Але якщо властивість не існує, то повернеться спеціальне значення *undefined*:

var person = {};

alert (person.lalala); // undefined, немає властивості з ключем lalala *Доступ до властивостей*

1. Доступ через квадратні дужки.

Існує альтернативний синтаксис роботи з властивостями, що використовує квадратні дужки об'єкт [ 'властивість']:

var person = {};

person [ 'name'] = 'Вася'; // то ж що і person.name = 'Вася'

Записи person [ 'name'] і person.name ідентичні, але квадратні дужки дозволяють використовувати в якості імені властивості будь-який рядок. 2. Доступ до властивості через змінну

Квадратні дужки також дозволяють звернутися до властивості, ім'я якої зберігається в змінної:

var person = {};

person.age = 25;

var key = 'age';

alert (person [key]);

10

*Об'єкти: перебір властивостей*

Для перебору всіх властивостей з об'єкта використовується цикл за властивостями *for..in*. Ця синтаксична конструкція відрізняється від циклу *for (;;)* і має наступний синтаксис:

for (key in obj)

{

/ \* ... робити щось з obj [key] ... \* /

}

При цьому *for..in* послідовно перебере властивості об'єкта obj, ім'я кожної властивості буде записано в key і викликано в тілі циклу.

**1.2.2 Масиви з числовими індексами**

Масив - різновид об'єкта, який призначено для зберігання пронумерованих значень і пропонує додаткові методи для зручного маніпулювання такою колекцією.

Вони зазвичай використовуються для зберігання впорядкованих колекцій даних, наприклад - списку товарів на сторінці, студентів в групі і т.п. Синтаксис для створення нового масиву - квадратні дужки зі списком елементів всередині.

Порожній масив:

var arr = [];

Масив fruits з трьома елементами:

var fruits = ["Яблуко", "Апельсин", "Слива"];

Загальна кількість об'єктів, що зберігаються в масиві, міститься в його властивості *length*.

Довжина *length* - не кількість елементів масиву, а останній індекс + 1. Існує ще один синтаксис для створення масиву:

var arr = new Array ("Яблуко", "Груша", "і т.п.");

Крім того, у нього є одна особливість. Зазвичай *new Array (елементи, ...)* створює масив з даних елементів, але якщо у нього один аргумент-число *new Array (число)*, то він створює масив без елементів, але із заданою довжиною.

11

**Приклад виконання**

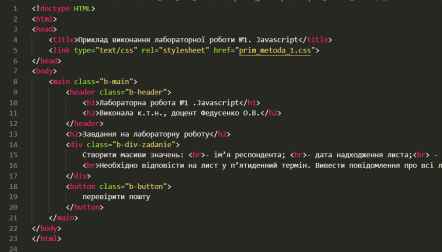
Розробимо веб-застосунок для виконання наступного завдання: Створити масиви значень:

− ім’я респондента;

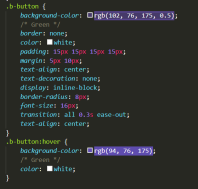
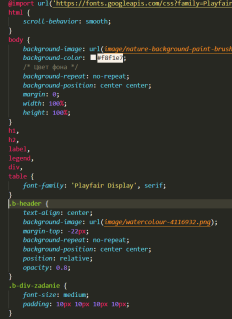
− дата надходження листа;

− адреса електронної пошти.

Необхідно відповісти на лист у п’ятиденний термін. Вивести повідомлення про всі листи, які надійшли з вказанням терміну, що залишився для відповіді. Листи з мінімальним терміном (1 день) підкреслити жовтим кольором. Листи, відповідь на які не була надана, виділити червоним кольором.

1. Створимо html сторінку, відповідно до вимог лабораторної роботи, для цього напишемо html код нашої сторінки (рис.1.1), стилі CSS винесемо у окремий файл (рис.1.2). Зовнішній вигляд сторінки наведено на рис.1.3. Рисунок 1.1 Html файл

12

Рисунок 1.2 CSS стилі головної сторінки

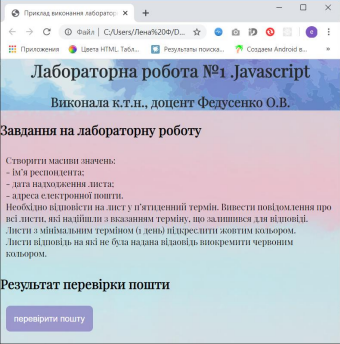


Рисунок 1.3 Зовнішній вигляд сторінки з лабораторною роботою

13

2. Розробимо Javascrip підпрограму у окремому файлі для виконання поставленого завдання. Вона буде складатися з декількох функцій специфікація яких наведена у таблиці 1.1

Таблиця 1.1 Специфікація функцій

| Функція | Опис |
| --- | --- |
| ras\_Diff\_Days(text) | Функція призначена для розрахунку терміну, що залишився для відповіді на лист.  Вхід:  text – дата для порівняння у вигляді рядка  Вихід:  термін листа |
| vivod(item,i,array) | Функція для формування рядка виводу на екран. Вхід:  item – поточний елемент масиву array  i – номер поточного елементу  array - масив  Вихід:  html – рядок виводу |
| ras() | Головна функція. В ній відбувається формування рядку виводу html та виклик функції vivod(). |

Розглянемо структуру розробленого програмного забезпечення.

14

prim\_metoda\_1.html

prim\_metoda\_1\_zadanie.js prim\_metoda\_1.css ras()

vivod(item,i,array)

ras\_Diff\_Days(text)

Рисунок 1.4 Структурна схема програмного застосунку

3. У файлі prim\_metoda\_1\_zadanie.js створимо масив об‘єктів, у які занесемо дані про листи (рис. 1.5).



Рисунок 1.5 Масив початкових значень

Також необхідно оголосити змінну, у якій ми будемо формувати рядок для виведення на html-сторінку.

15

var html=””;

4. Для виведення даних необхідно трохи змінити html файл, а саме: • додати перед описом кнопки блок *div* з *id=”rezult”* та заголовок *h2* з текстом «Результати перевірки пошти»;

• підключити файл зі скриптом у секції *head* за допомогою тегу *<script>*.

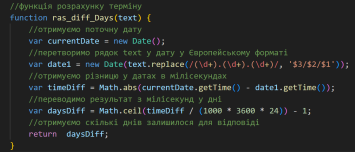
5. Функцію розрахунку терміну відповіді на лист наведено на рис. 1.6. 

Рисунок 1.6 Функція виведення окремого запису

6. Функцію, яка формує рядок виводу для окремого запису масиву об‘єктів наведено на рис. 1.7.

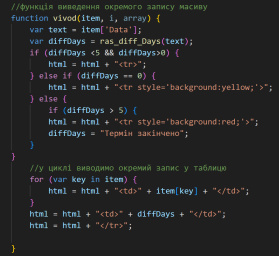
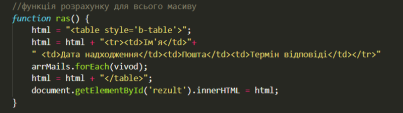


Рисунок 1.7 Функція виведення окремого запису

7. Головну функцію наведено на рис.1.8.

16

Метод *forEach* виконує вказану функцію один раз для кожного елементу масиву.

Рисунок 1.8 Головна функція

8. Результати виконання веб застосунку наведено на рис. 1.9. Рисунок 1.9 Результати виконання тестового прикладу

17

**Варіанти завдань**

1. Створити масиви значень:

− ім’я друга;

− дата народження;

− номер телефону (допускаються пусті значення).

За 3 дні потрібно відправити вітальну листівку, а в день народження поздоровити іменинника особисто по телефону. Вивести повідомлення про необхідність відправлення листівки чи телефонного дзвінка. Вивести додаткове повідомлення, якщо номер телефона не вказано.

2. Створити масиви значень:

− концертна група чи виконавець;

− місце проведення концерту (допускається пусте значення); − дата проведення концерту.

Квиток потрібно придбати за 5 днів до концерту. Вивести повідомлення про те, що сьогодні останній термін придбання квитка, чи сьогодні йде концерт. Якщо не вказано місце проведення концерту, вивести додаткове повідомлення „Невідома адреса проведення концерту”.

3. Створити масиви значень:

− назва предмету;

− дата іспиту;

− аудиторія (допускається пусте значення).

За день до іспиту проводиться консультація. Вивести повідомлення про те, що сьогодні повинна бути консультація чи іспит. Якщо не вказана аудиторія, вивести додаткове повідомлення „Невідоме місце проведення консультації/іспиту”.

4. Створити масиви значень:

− назва предмету;

− початок пари;

− прізвище викладача (допускається пусте значення).

18

Тривалість пари 80 хвилин. Вивести інформацію про те, яка зараз пара, опираючись на поточний час. Якщо в даний момент немає жодної пари, вивести одне з повідомлень:

− пари ще не почалися;

− перерва після <номер> пари;

− пари закінчилися.

Якщо не вказано прізвище викладача, вивести додаткове повідомлення. 5. Створити масиви значень:

− тематика виставки;

− місце проведення;

− організатор виставки (допускається пусте значення).

Кожна виставка працює протягом тижня. Вивести інформацію, скільки днів залишилося до кожної з них, яка(і) виставка(и) проводиться сьогодні, скільки днів залишилось до її(їх) закриття. Якщо не вказаний організатор виставки, вивести додаткове повідомлення.

19

**Лабораторна робота №2. Основи моделі DOM та події**

**Мета роботи:** навчитися створювати скрипти на мові JavaScript з використанням моделі DOM та прив‘язувати їх виконання до певних подій. **Завдання:**

Додати до розробленої у першій лабораторній роботі сторінки головне меню, яке буде реалізовувати виклики сторінок з лабораторними роботами 2,3,4. Розробити дизайн веб-сторінки для лабораторної роботи № 2 та написати JavaScript для вирішення поставленого завдання відповідно до варіанту. Розроблена HTML-сторінка повинна відповідати таким вимогам: • Таблиці стилів повинні розташовуватися в окремих файлах. • Верстка повинна бути адаптивною.

• Скрипти для виконання завдань повинні бути розміщені у окремих файлах.

HTML-сторінка повинна містити:

• "шапку", що містить ПІБ студента, номер групи і номер варіанту; • опис завдань відповідно до варіанту;

• єдине стилістичне оформлення з головної сторінкою.

**Заборонено використання jQuery для вирішення поставлених завдань.**

**Теоретичні відомості**

**2.1 Модель DOM**

DOM (Document Object Model) – це стандартизований підхід до пошуку та маніпулювання вузлами та елементами Web-документа[3]. DOM дозволяє виконувати такі задачі, як пошук вузлів за визначеним критерієм, додавати, вилучати та замінювати вузли Web-документа, звертатись до атрибутів та змінювати вміст елементів HTML, маніпулювати стилями CSS, а також під‘єднувати обробники подій. Практично, в браузерах HTML DOM реалізується у вигляді об‘єктно-орієнтованого програмного інтерфейсу (API), де об‘єктами є

20

вузли Web-документа, властивостями – їх атрибути, а методи визначають логіку обробки даних вузлів.

Вузли документа відповідно до DOM поділяються наступним чином: • Увесь документ – це вузол-документ.

• Кожний елемент HTML – вузол-елемент.

• Текстовий вміст елемента HTML – це текстовий вузол. • Атрибут елемента HTML – це вузол-атрибут.

• Коментар – це вузол-коментар.

Розглянемо приклад, нехай у нас є html документ який має наступну структуру (рис. 2.1) його об‘єктна модель наведена на рис. 2.2. 

Рисунок 2.1 Простий html документ

Рисунок 2.2 Об‘єктна модель html документу

Серед методів відбору елементів документа можна виділити наступні: • *getElementById* – отримати елемент за його атрибутом id. • *getElementsByTagName* – отримати елементи за назвою тега.

21

• *getElementsByName* – отримати елементи за їх атрибутом name. Для маніпуляції елементами документа використовують наступні методи: • *appendChild –* додати елемент у кінець списку.

• *removeChild –* вилучити елемент.

• *replaceChild –* замінити елемент.

• *getAttribute/setAttribute –* отримати/встановити значення атрибута та інші.

**2.2 Події у JavaScript**

Зміни в веб сторінках можуть відбуватися або циклічно, або у відповідь на дії користувача. Прикладом реакції на дії користувача служить збільшення розміру малюнка після клацання по ньому мишкою. У всіх випадках сигналом до змін служить якась подія. У HTML події включені в число атрибутів елементів. Нижче наведені обробники подій, які використовуються більшістю елементів:

• onclick виникає, коли миша клацає на елементі;

• ondblclick виникає, коли миша двічі клацає на елементі;

• onmousedown виникає, коли кнопка миші "натиснула" на елемент; • onmouseup виникає, коли кнопка миші відпущена над елементом; • onmouseover виникає, коли миша проходить над елементом; • onmousemove виникає, коли миша переміщається в той момент, коли знаходиться над елементом;

• onmouseout виникає, коли миша забирається з елемента; • onkeypress виникає, коли клавіша натиснута і відпущена над елементом;

• onkeydown виникає, коли клавіша натиснута над елементом. • onkeyup виникає, коли клавіша відпущена над елементом.

**Приклад виконання**

Розробимо веб-застосунок для виконання наступного завдання:

22

На Вашій сторінці є поле введення в яке записано число. При наведенні на нього мишки вивести квадрат цього числа та корінь, якщо воно додатне, квадрат та куб, якщо воно від‘ємне.

1. Змінимо html файл який був розроблений у попередній лабораторній роботі, додавши до нього опис головного меню для виклику лабораторних робіт № 2,3,4 (рис. 2.3). Для оформлення головного меню напишемо окремий CSS файл (рис. 2.4). Не забуваємо підключити наш новий CSS-файл до файлу з розміткою у секції *head* за допомогою тегу *<link>*. Вигляд головної сторінки наведено на рис. 2.5.



Рисунок 2.3 Html файл

23

 Рисунок 2.4 CSS-файл з меню

Рисунок 2.5 Головна сторінка з меню

2. Розробимо html файл для виконання завдання другої лабораторної роботи, по структурі він буде подібний до файлу першої роботи, але необхідно додати числове поле введення, також для виклику цього html файлу необхідно

24

прописати його адресу у головному меню. Зовнішній вигляд сторінки показано на рис. 2.6.

Рисунок 2.6 Сторінка з завданням

3. Розробимо скрипт для виконання завдання. Нам необхідно написати дві функції: одна для розрахунку та виведення числа, коли миша наведена на елемент, інша для очищення, коли миша не наведена на елемент (рис. 2.7). Ці функції ми підключимо у html файлі (рис. 2.8) на обробника події *onmouseover –* наведення миші на елемент, *onmouseout –* відведення миші від елемента. Результат виконання для додатного та від‘ємного чисел показано на рис 2.9.

25



Рисунок 2.7 Фрагмент html коду сторінки



Рисунок 2.8 JavaScript-код

 Рисунок 2.9 Результат виконання коду

**Варіанти завдань**

Варіанти до завдання два наведено в таблиці 2.1

26

Таблиця 2.1 Варіанти завдання 2

| **Варіант** | **Завдання** |
| --- | --- |
| 1 | На Вашій сторінці є посилання. При наведенні на нього миші вивести додаткову інформацію про це посилання в діалоговому вікні.  Ввести два числа (поля вводу) і слово „сума”, „різниця”, „добуток” чи „ділення” (з клавіатури). Вивести результат виконання операції. Передбачити можливість неправильного вводу слова користувачем. |
| 2 | На Вашій сторінці є малюнок. При наведенні миші на цей малюнок він повинен змінитися. Коли Ви відведете мишку в сторону, малюнок має набути початкового вигляду.  Ввести № телефона (поле вводу) і назву міста (з клавіатури). Якщо введено місто, якого немає в списку, то повідомити про це користувача, інакше вивести № телефона, поставивши перед ним код міста. |
| 3 | На Вашій сторінці є поле введення. При наведенні на нього мишки вивести додаткову інформацію про це поле.  Ввести № квартири (з клавіатури), № під’їзду і поверх (поля вводу). У будинку 3 під‘їзди, 9 поверхів і 108 квартир. Проаналізувати правильність введення даних і видати відповідне повідомлення. |
| 4 | На Вашій сторінці є кнопка. При наведенні на неї вивести текст.  Ввести групу (з клавіатури), прізвище студента та курс (поля вводу). Якщо неправильно вказано курс (перша цифра номеру групи), вивести попередження, інакше вивести повне ім’я студента. |

27

**Лабораторна робота №3. Модель DOM**

**Мета роботи** – навчитися створювати скрипти JavaScript з використанням моделі DOM та додавати власні html елементи під час виконання скриптів. **Завдання:**

Розробити дизайн веб сторінки для лабораторної роботи № 3 та написати JavaScript для вирішення поставлених завдань відповідно до варіанту. Розроблена HTML-сторінка повинна відповідати таким вимогам: • Таблиці стилів повинні розташовуватися в окремих файлах. • Верстка повинна бути адаптивною.

• Скрипти для виконання завдань повинні бути розміщені у окремих файлах.

HTML-сторінка повинна мати:

• "шапку", що містить ПІБ студента, номер групи і номер варіанту; • опис завдань відповідно до варіанту;

• єдине стилістичне оформлення з головної сторінкою.

**Заборонено використання jQuery для вирішення поставлених завдань.**

**Теоретичні відомості**

У різних DOM-вузлів можуть бути різні властивості. Наприклад, у вузла, відповідного тегу <a>, є властивості, пов'язані з посиланнями, а у відповідного тегу <input> - властивості, пов'язані з полем введення і т.д. Текстові вузли відрізняються від вузлів-елементів. Але у них є спільні властивості і методи, тому що всі класи DOM-вузлів утворюють єдину ієрархію. Кожен DOM-вузол належить відповідному вбудованому класу та має свої властивості:

• Властивість *nodeType* надає спосіб дізнатися «тип» DOM-вузла. • Отримавши DOM-вузол, ми можемо дізнатися ім'я його тега з властивостей *nodeName* і *tagName*.

• Властивість *innerHTML* дозволяє отримати HTML-вміст елемента у вигляді рядка.

28

• Властивість *outerHTML* містить HTML елемента цілком. Це як *innerHTML* плюс сам елемент.

• Властивість *innerHTML* є тільки у вузлів-елементів. У інших типів вузлів, зокрема, у текстових, є свої аналоги: властивості *nodeValue* і *data*. • Властивість *textContent* надає доступ до тексту всередині елемента за вирахуванням всіх <тегів>.

• Атрибут та DOM-властивість «*hidden*» вказує на те, чи ми бачимо елемент чи ні.

У DOM-елементів є додаткові властивості, які, зокрема, залежать від класу: • *value* - значення для <input>, <select> і <textarea>

• *href* - адреса посилання «href» для <a href="...">

• *id* - значення атрибута «id» для всіх елементів (HTMLElement). • і багато інших.

**Приклад виконання**

Розробимо веб-застосунок для виконання наступного завдання: • На сторінці є блок у якій занесено певний текст, при натисканні на кнопку тест розбивається на речення, які виводяться у нумерований список (кожне речення в окремий тег li).

• На сторінці є таблиця, в якій зберігається квадратна матриця. При натисканні на кнопки знаходиться мінімальний елемент головної діагоналі та виділяється червоним, а усі інші числа на діагоналі виділяються зеленим кольором.

1. Розробимо html-файл для виконання завдання третьої лабораторної роботи, по структурі він буде подібний до файлу другої роботи, але необхідно додати дві кнопки та таблицю з числами; також для звернення до цього html файлу необхідно прописати його адресу у головному меню. Зовнішній вигляд сторінки наведено на рис. 3.1. Для оформлення таблиці створимо власний файл зі стилями (рис.3.2).

29

Рисунок 3.1 Сторінка з завданням



Рисунок 3.2 Стильове оформлення таблиці

2. Розробимо скрипт для виконання першого завдання. Нам необхідно написати функцію для аналізу тексту та розбиття його на речення. Для отримання тексту з блоку div будемо використовувати властивість вузла моделі DOM *textContent*. Для розбиття отриманого тексту на речення будемо використовувати метод *split().* Метод *split()* розбиває об'єкт *String* на масив рядків шляхом поділу рядка зазначеним підрядком та повертає масив рядків. Текст функції наведено на рис. 3.3, її необхідно підключити до обробника події *onclik* першої кнопки. Результат виконання скрипту наведено на рис. 3.4.

30



Рисунок 3.3 Функція для вирішення першого завдання 

Рисунок 3.4 Результат виконання скрипту для першого завдання

3. Розробимо функцію для виконання другого завдання. Нам необхідно спочатку отримати дані з таблиці – функція generate\_arr(), потім провести розрахунок мінімального значення та номера його чарунки – функція min, а вже після цього оформити таблицю по новому – функція color(). Для підключення до події onclick у html файлі розробимо окрему функцію, яка буде визивати усі попередні функції – table\_obrab() (рис. 3.5). Результат виконання скрипту JavaScript наведено на рис. 3.6.

31

 Рисунок 3.5 Функції JavaScript для другого завдання

Рисунок 3.6 Результат виконання скрипту для другого завдання

**Варіанти завдань**

**Завдання 1**

1. На Вашій сторінці є поле вводу, у яке через кому вводяться назви країн. При натисканні на кнопку зробіть так, щоб ці країни записалися в список ul під поле (кожна країна – окремий li).

2. У поле вводу вводиться країна і натискається Enter. При натисканні на Enter зробіть так, щоб введені країни записувалися в абзац під поле вводу через кому. Тобто по кожному натискання Enter в абзац будуть додаватися все нові і нові країни.

32

3. На сторінці є два блоки div, які містять текст. Обійдіть всі блоки div і обріжте текст в них таким чином, щоб він став довжиною 10 символів. В кінець обрізаного тексту додайте три крапки.

4. Дано блок з кнопкою 'закрити блок'. При натисканні на цю кнопку блок повинен зникнути з форми. Кнопка розміщується всередині блоку і повинна зникнути разом з ним. Блоків може бути будь-яка кількість, допускається щоб один з блоків був всередині іншого. Кожен блок закриває своя кнопка.

**Завдання 2**

1. Дана таблиця з числами. При натисканні на кнопку знайдіть комірку, в якій зберігається максимальне число, і зробіть її фон червоним. 2. Дана таблиця з числами. При натисканні на кнопку в поле введення під таблицею відобразіть числа з таблиці. Виведення виконати через кому в порядку зростання чисел.

3. Дана таблиця з числами. При натисканні на кнопку в останній ряд таблиці в кожну клітинку запишіть суму чисел в стовпці таблиці. 4. Дана таблиця з числами. При натисканні на комірку, вона активується і стає червоного кольору. Активувати можна багато комірок. Під таблицею кнопка. При натисканні цієї кнопки в абзац нижче виведіть суму активованих комірок. Реалізуйте також кнопку "скинути активовані комірки". 5. Дана таблиця. При натисканні на комірку в ній з'являється поле введення з вмістом цієї комірки і кнопка "зберегти". Можна редагувати текст в полі, зберегти - і текст комірки зміниться на той, що був в полі введення.

33

**Лабораторна робота №4. Основи ООП у JavaScript**

**Мета роботи** –навчитися створювати власні класи у JavaScript **Завдання;**

Розробити дизайн веб сторінки для лабораторної роботи № 4 та написати JavaScript для вирішення поставленого завдання відповідно до варіанту. Розроблена HTML-сторінка повинна відповідати таким вимогам: • Кожну таблицю стилів необхідно реалізувати в окремому файлі. • Верстка повинна бути адаптивною.

• Скріпти для виконання завдань повинні бути реалізовані у окремих файлах.

HTML-сторінка повинна містити:

• "шапку", що містить ПІБ студента, номер групи і номер варіанту; • опис завдань відповідно до варіанту;

• єдине стилістичне оформлення з головної сторінкою.

**Заборонено використання jQuery для вирішення поставлених завдань.**

**Теоретичні відомості**

**4.1 Методи об‘єктів**

При оголошенні об'єкта можна вказати властивість-функцію. Властивості функції називають «методами» об'єктів. Їх можна додавати і видаляти в будь який момент, в тому числі і явним присвоєнням. Для повноцінної роботи метод повинен мати доступ до даних об'єкта. Для доступу до поточного об'єкту з методу використовується ключове слово *this*.

Використання *this* гарантує, що функція працює саме з тим об'єктом, в контексті якого викликана. Через *this* метод може не тільки звернутися до будь якого властивості об'єкта, а й передати будь-куди посилання на сам об'єкт вцілому.

Будь-яка функція може мати в своєму складі *this*. При цьому абсолютно неважливо, чи оголошена вона в об'єкті чи поза ним. Значення *this* називається

34

контекстом виклику і буде визначено в момент виклику функції. Якщо одну і ту ж функцію запускати в контексті різних об'єктів, вона буде отримувати різний *this*.

**4.2 Конструктори**

Конструктором стає будь-яка функція, викликана через *new*.

function Animal (name)

{

this.name = name;

this.canWalk = true;

}

var animal = new Animal ( "їжачок");

Зауважимо, що, технічно, будь-яка функція може бути використана як конструктор. Тобто, будь-яку функцію можна викликати за допомогою *new*. Якось особливим чином вказувати, що вона є конструктором не потрібно.

Але, щоб виділити функції, задумані як конструктори, їх називають з великої літери: Animal, а не animal. У стандарті ES6 з‘явилася можливість використання ключового слова constructor.

Детальніше - функція, запущена через new, робить наступне: • Створюється новий порожній об'єкт.

• Ключове слово *this* отримує посилання на цей об'єкт. • Функція виконується. Як правило, вона модифікує *this* (тобто цей новий об'єкт), додає методи, властивості.

• Повертається *this*.

В результаті виклику new Animal ("їжачок"); отримуємо такий об'єкт: animal = { name: "їжачок",

canWalk: true}

**Приклад виконання**

Розробимо веб-застосунок для виконання наступного завдання:

35

Створити сторінку зі скриптом, в якому створюється об'єкт трикутник, що буде мати такі властивості: катети a та b, гіпотенузу c, площу S. Перші два поля передаються в конструктор, останні два поля необхідно обчислити. Створити масив з декількома об‘єктами даного класу. Вивести після натискання кнопки всі значення властивостей на екран у вигляді таблиці.

1. Розробимо html файл для виконання завдання четвертої лабораторної роботи, по структурі він буде подібний до файлу другої роботи, але необхідно додати кнопку для виведення масиву, також для виклика цього html файлу необхідно прописати його адресу у головному меню. Зовнішній вигляд сторінки показано на рис. 4.1.

Рисунок 4.1 Сторінка з завданням

2. Створимо клас для збереження інформації про фігуру та опишемо його конструктор і методи. Основним буде метод виведення інформації на екран (рис 4.2).



Рисунок 4.2 Клас трикутника

36

3. Далі необхідно створити функцію для виведення декількох об‘єктів (рис. 4.3), результати роботи показано на рис. 4.4.



Рисунок 4.3 Основна частина скрипту



Рисунок 4.4 Результат виконання

**Варіанти завдань**

1. Розробити сторінку зі скриптом, в якому створюється об'єкт Машина з трьома властивостями: колір, марка, модель. Створити масив з декількома об‘єктами даного класу. Вивести після натискання кнопки всі значення властивостей на екран у вигляді таблиці.

2. Розробити сторінку зі скриптом, в якому створюється об'єкт Холодильник з 4-ма властивостями: колір, марка, модель, ціна. Створити масив з декількома об‘єктами даного класу. Вивести після натискання кнопки всі значення властивостей на екран у вигляді таблиці.

3. Розробити сторінку зі скриптом, в якому створюється об'єкт Комп'ютер з 3-ма властивостями: модель, виробник, ціна. Створити масив з декількома

37

об‘єктами даного класу. Вивести після натискання кнопки всі значення властивостей на екран у вигляді таблиці.

4. Розробити сторінку зі скриптом, в якому створюється об'єкт Квартира з 4-ма властивостями: адреса, кількість кімнат, ціна, ремонт. Створити масив з декількома об‘єктами даного класу. Вивести після натискання кнопки всі значення властивостей на екран у вигляді таблиці.

38

**Лабораторна робота №5. Основи фреймворку Angular**

**Мета роботи.** Навчитися програмувати застосування з використанням фреймворку Angular.

**Завдання:**

Необхідно написати проект на фреймворку Angular, який визначає чи потрапляє точка на координатній площині в задану область, чи ні, відповідно до варіанту завдань, і містить HTML-сторінку, яка формує дані для відправки їх на обробку. Вхідними даними є:

• xn- початкове значення координати х,

• xk - кінцеве значення координати х,

• yn- початкове значення координати у,

• yk- кінцеве значення координати у,

• хh – крок зміни координати х,

• уh – крок зміни координати у,

• R – значення радіуса для фігури.

Скрипт повинен виконувати перевірку даних (валідацію) значень, що вводяться користувачем в поля форми і повертати HTML-сторінку з таблицею, яка містить отримані параметри і результат обчислень – факт входження або не входження точки в задану область. Крім того, якщо точка потравляє в область,

то такий рядок виділяємо зеленим кольором, якщо ні – червоним. Будь-які некоректні значення (наприклад, букви в координатах точки або від’ємне значення радіуса) повинні блокуватися.

Крім того, відповідь має містити інформацію про поточний час і тривалість виконання скрипта.

Розроблена HTML-сторінка повинна відповідати таким вимогам: • Таблиці стилів повинні розташовуватися в окремих файлах. • Верстка повинна бути адаптивною.

HTML-сторінка повинна містити:

39

• "шапку", що включає ПІБ студента, номер групи і номер варіанту; • картинку з заданим варіантом;

• форму для введення значень.

**Теоретичні відомості**

Angular - це JavaScript фреймворк, тому багато мов програмування, що дозволяють працювати з Angular, також ґрунтуються на JavaScript. Серед них можна відзначити такі як: ES5, ES6 (ES2015), TypeScript. Окремо стоїть мова Dart, яка, на відміну від перерахованих вище, не має ніякого відношення до JavaScript.

Перші два варіанти базуються на версіях ECMAScript - специфікації мови JavaScript.

TypeScript - це надбудова над JavaScript; код на TypeScript повинен бути попередньо транспірований. Він строго типізований (у кожної змінної в TypeScript є свій тип даних), завдяки строгій типізації дається доступ до цілого набору інструментів, включаючи

• вбудовану документацію,

• перевірку синтаксису,

• навігацію по коду,

• можливість рефакторінга.

Angular є фреймворком, який досить швидко оновлюється. Для того, щоб отримати останню версію Angular необхідно використовувати сайт https://angular.io/.

**1. Встановлення Angular**

**1.1 Встановлення NodeJs**

Для роботи з Angular, необхідно встановити NPM. Його можна встановити окремо, з офіційного сайту NPM, а також разом з node.js, щоб відразу можна було створити тестовий додаток і запустити його.

40

NPM (Node Package Manager) - утиліта командного рядка, яка взаємодіє з репозиторіями openSource-проектів.

NPM - це пакетний менеджер, що використовується для установки бібліотек, пакетів та програм, а також їх залежностей.

NPM потрібен для установки бібліотек для Angular, а також щоб виконувати скрипти, для трансляції коду TypeScript і запуску додатка на Angular. Щоб встановити NPM, завантажте його зі сторінки завантажень NodeJs - https://nodejs.org/uk/(рис. 5.1).

Node.js - це середовище виконання JavaScript, створена на базі JavaScript движка V8 з Chrome. В основі Node.js лежить подієво-керована модель з операціями I/O, які не блокуються, що робить її легкою і ефективною.

Іншими словами: Node.js надає можливість писати продуктивний серверний код з використанням JavaScript.

Рисунок 5.1. Сторінка завантаження NodeJs

Завантажте та встановить пакет, усі налаштування залишаємо за замовченням.

Після завантаження запустіть Node.js command prompt (рис. 5.2).

41



Рисунок 5.2 Пошук Node.js command prompt

У вікні, що відкриється, необхідно виконати такі команди (рис.5.3): node -v

npm -v



Рисунок 5.3 Перевірка версій Node.js command prompt

**1.2 Встановлення Angular CLI**

Наступним кроком встановлюємо angular CLI (Command Line Interface - інтерфейс командного рядка Angular), він же - Angular framework. Для цього у відкритій нами консолі виконаємо команду:

npm install -g @angular/cli

Рекомендації щодо цієї команди ви також можете знайти на офіційному сайті.

Результати встановлення Angular наведено на рис.5.4.

42

Рисунок 5.4 Встановлення Angular

**2. Створення першого проекту на Angular**

Для створення нового проекту необхідно в консолі виконати команду: ng new my-app

ng - всі команди angular framework починаються з «ng»,

new - команда створення нового проекту,

my-app - ім'я проекту (можете задати його будь-яким, використовуючи латиницю).

Створення проекту займе певний час. По завершенню ми можемо запустити проект, використовуючи той же command prompt:

cd my-app

ng serve --open

Якщо все пройшло успішно, у вас відкриється адреса localhost, зі стандартним портом, де ви побачите сторінку вашого першого застосунку на Angular (рис. 5.5):

43

Рисунок 5.5 Перший проект на Angular

**3. Структура проекту на Angular**

Якщо ми відкриємо наш проекту у Microsoft Visual Code, то побачимо сукупність файлів проекту (рис. 5.6):



Рисунок 5.6 Файли проекту на Angular

44

Специфікація файлів проекту Angular наведена у таблиці 2.1. Таблиця 2.1 Специфікація основних файлів та папок проекту Angular

| Назва файлу, папки | Опис |
| --- | --- |
| angular.json | Файл відповідає за налаштування Angular CLI. |
| .editorconfig | Файл конфігурації загальної стилістики для проекту. |
| .gitignore | Сповіщає систему контролю версій Git, які файли та директорії не потрібно враховувати. |
| karma.conf.js | Файл конфігурації для модульних тестів. |
| package.json | Описує залежності проекту та вказує які саме пакети потрібні для роботи. |
| tsconfig.json | Файл конфігурації для TypeScript. |
| tslint.json | Файл конфігурації правил форматування. |
| node\_modules | Папка, в якій зберігається увесь початковий код усіх бібліотек, що приєднанні до проекту |
| src | Папка, в якій зберігається увесь початковий код застосунку. Тобто розробка відбувається саме у цій папці. |
| src/index.html | Файл, в якому відбувається розробка  функціоналу проекту. |
| src/main.ts | Файл, який запускає весь проект. |
| src/polyfills.ts | Файл підключення необхідних бібліотек. |
| src/app | Папка, в якій відбувається розробка. |

**4. Створення компонентів**

Компонент в Angular складається з декількох частин (рис. 5.7).

Template

•Views

•Bindings

Class

•Properties •Methods

45

Metadate

•DecoratorComponent

Рисунок 5.7 Структура компоненту на Angular

1. Шаблон (Template) - це фрагмент призначеного для користувача інтерфейсу, що визначає зовнішній вигляд програми. Шаблон складається з: • Html-розмітка - задає зовнішній вигляд компонента;

• Прив'язки (bindings) і директиви (directives), які надають додатку інтерактивність і прив'язують розмітку до логіки класу.

2. Клас - код на TypeScript, що підтримує представлення і складається з:

• Властивості (properties), визначають дані класу.

• Методи (methods), визначають логіку компонента.

3. Метадані (metadata) - надають додаткові інформацію про клас для Angular; метадані визначаються декоратором - функцією, яка додає метадані в клас, його методи або аргументи методів.

Декоратори обертають метадані, тим самим визначаючи очікувану поведінку класу. Наприклад, якщо ми напишемо в класі декоратор Component, то він вкаже Angular, що клас є компонентом. У той час як, наприклад, декоратор @Routes, говорить про те, що метадані містять відомості про маршрутизацію.

За замовчуванням Angular створив компонент, якій зберігається у файлі src/app/app.component.ts (рис.5.8).

46

Рисунок 5.8 Компонент створений за замовченням Angular

Змінювати html код у проекті можна у файлі src/app/ app.component.html. Змінимо зовнішній вигляд нашої сторінки. Для цього видалимо стандартний шаблон та опишемо заголовок першого рівня (рис. 5.9).

Рисунок 5.9. Заголовок першого рівня

Для запуску нашого проекту у терміналі напишемо команду npm run start

Angular CLI збере проект та напише його адресу на локальному хості (рис. 5.10).

47

Рисунок 5.10 Збирання проекту

Зовнішній вигляд проекту наведено на наступному рисунку.



Рисунок 5.11 Зовнішній вигляд проекту

**Приклад виконання**

Розробимо адаптивний веб-застосунок для перевірки належності точки області, наведений нижче(рис. 5.12).



Рисунок 5.12 Зовнішній вигляд області

1. Спочатку нам необхідно розробити інтерфейс взаємодії з користувачем, його можна поділити на дві частини. Частина, яка буде знаходитись у головному файлі і ніяк не буде пов‘язана з розрахунком (шапка

48

сторінки, підвал сторінки, малюнок з завданням) та частина, що пов‘язана з розрахунком (форма для введення, кнопка для розрахунку тощо). 2. Створимо першу частину інтерфейсу для цього у файлі app.component.html



Рисунок 5.13 Перша частина інтерфейсу

Стилі напишемо у файлі style.CSS.



Рисунок 5.14. Стилі для першої частини інтерфейсу

Запустимо наш проект за допомогою команди

npm run start

і отримаємо результат (рис. 5.15).

49



Рисунок 5.15 Зовнішній вигляд першої частини інтерфейсу

3. Для подальшої роботи нам необхідно розробити власний компонент. Для цього створимо нову папку у нашому проекті MyComponent і у папці створимо файли для нового компоненту point.component.ts та point.component.html, при необхідності можемо створити і файл для збереження CSS.

4. У файлі нового компонента point.component.ts нам необхідно імпортувати бібліотеку компонентів за допомогою наступного коду: 5. import { Component } from '@angular/core';

Далі опишемо наш компонент. У компоненті необхідно вказати селектор та поєднати його з файлом шаблону (html файлом).

@Component

({

selector: 'app-my',

templateUrl: './point.component.html'

})

Ми вказали селектор – app-my

50

Та пов‘язали з файлом шаблону - point.component.html.

6. Далі нам потрібно імпортувати наш новий компонент та додати його до нашого проекту. Для цього нам необхідно змінити файл app.module.ts таким чином (рис. 5.16)

Рисунок 5.16 Файл app.module.ts

7. Тепер ми можемо написати наш селектор у головному html файлі для роботи з нашим компонентом. Головний файл app.component.html буде мати такий вигляд (рис. 5.17).

Рисунок 5.17 Файл app.component.html з новим селектором

8. Створимо новий клас PointComponent для розрахунків (рис. 5.18).

51

Рисунок 5.18 Клас PointComponent

9. У файлі point.component.html створимо інтерфейс для введення даних, усі стилі будуть знаходитися у головному файлі style.CSS (рис. 5.19). Для передачі даних з форми у компонент призначимо кожному полю введення ім‘я за допомогою наступної команди:

#name – надає посилання на елемент введення (input).

52

 Рисунок 5.19 Форма для введення даних та її CSS - стилі

Для верстання форми використовується технологія Grid. Інтерфейс веб застосунку буде мати такий вигляд (рис. 5.20).

Рисунок 5.20 Інтерфейс веб-застосунку

10. Додаємо до інтерфейсу кнопку для проведення розрахунку. Цю кнопку необхідно зв’язати з методом компонента setall. Для цього будемо використовувати подію натискання на кнопку click, яку необхідно вказати у круглих дужках. У метод setall передаємо вміст полів вводу. Також напишемо для кнопки власні CSS властивості (рис. 5.21)

53

<button class="b-button"

(click)="setall(Inputxn.value,Inputxk.value,Inputhx.value,Inputyn.value, Inputyk.value,Inputhy.value,Inputr.value)"> Розрахунок

</button>



Рисунок 5.21 CSS – стилі для кнопки

11. Нам необхідно створити шаблон таблиці для виведення результатів розрахунків. Для цього ми будемо використовувати структурну директиву ngFor, яка дозволяє перебирати елементи масиву в шаблоні. В якості параметрів дана директива приймає значення для перебору

let item of items

Кожен елемент масиву items, який ми перебираємо, розміщується у зміну item. Змінні, які ми бажаємо вивести вказуються у подвійних фігурних дужках: {{item}}

Таким чином, наша таблиця буде мати такий вигляд (рис. 5.22).  Рисунок 5.22 Таблиця для виведення

54

12. Перевіримо роботу нашого веб застосунку, отримані результати наведено на рис. 32.

Рисунок 5.23 Результати роботи застосунку

13. Відповідно до завдання нам необхідно зафарбувати точки, що входять до області у зелений, а ті, що не входять – у червоний. Для цього будемо використовувати директиву ngStyle, яка дозволяє додати власний стиль. Для визначення кольору напишемо окремий метод у нашому класі.

getColor(b:boolean) {

return b === true ? 'green' : 'red'; }

А у шаблоні змінимо опис комірки, для виведення значення про належність точки до певної області .

<td \*ngFor="let bb of belar" [ngStyle]="{backgroundColor: getColor(bb)}">{{bb}} </td>

Результат, отриманий після внесених змін, наведено на рис. 5.24

55

Рисунок 5.24 Результати роботи застосунку

**Варіанти завдань**

Варіанти завдань наведено у таблиці 5.2.

Таблиця 5.2 Варіанти завдань

| №  варіанту | Завдання |
| --- | --- |
| 1. |  |

56

| 2. |  |
| --- | --- |
| 3. |  |
| 4. |  |
| 5. |  |

57

| 6. |  |
| --- | --- |
| 7. |  |

58

**Рекомендована література**

1. Стандарт вищої освіти України першого (бакалаврського) рівня ступеня «бакалавр» за галуззю знань 12 «Інформаційні технології» спеціальністю 122 «Комп’ютерні науки».

2. Національний стандарт України «Звіти у сфері науки і техніки. Структура та правила оформлювання. ДСТУ 3008:2015» [Електронний ресурс] – Режим доступу: http://www.knmu.kharkov.ua/attachments/3659\_3008-2015.PDF

3. Document Object Model (DOM) Standard. [Електронний ресурс] – Режим доступу: http://www.w3c.org/DOM/.

4. Цеслів О.В. WEB-програмування : навч. посібнник / О.В. Цеслів ; М во освіти і науки, молоді та спорту України, Нац. техн. ун-т України "Київ. політехн. ін-т". – Київ : НТУУ "КПІ", 2011. – 296 с.

5. MDN Web Docs. [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/ru/docs/Web/JavaScript

6. Angular Docs. [Електронний ресурс] – Режим доступу: https://angular.io/docs.

59

Навчальне видання

Федусенко О.В.

Гамоцька С.Л.

Доманецька І.М.

Красовська В.В.

**МЕТОДИЧНІ ВКАЗІВКИ**

**ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ З ДИСЦИПЛІНИ «Web-технології та Web-дизайн» JavaScript та Angular**

Видавництво ТОВ «ГЛІФ МЕДІА»

м. Київ, вул. Полупанова, 21

Формат 60х841/16. Папір офсетний.

Умовн. друк. арк. 2,0. Наклад 100 прим.