Міністерство освіти і науки України
Карпатський національний університет імені Василя Стефаника
Кафедра комп’ютерних наук та інформаційних систем

А. В. Ізмайлов

Методичні вказівки до виконання лабораторних робіт з дисципліни «Web-технології»

Івано-Франківськ
2025

УДК 004
І37

Рекомендовано до друку Вченою радою факультету математики та інформатики Карпатського національного університету імені Василя Стефаника (протокол № 9 від 22 жовтня 2025р.)

Рецензенти:

Іляш Ю.Ю., кандидат технічних наук, доцент, завідувач кафедри КНІС (Карпатський національний університет імені Василя Стефаника)

Семаньків М.В., кандидат технічних наук, доцент кафедри КНІС (Карпатський національний університет імені Василя Стефаника)

І37 Ізмайлов А.В. Методичні вказівки до виконання лабораторних робіт з дисципліни «Web-технології». Івано-Франківськ : КНУВС, 2025. 29 с.

Наведено методичні вказівки і завдання до лабораторних робіт з дисципліни «Web-технології». Призначено для проведення лабораторних занять з курсу «Web-технології» та інших, у яких вивчається мова програмування JavaScript.

Для здобувачів освіти за спеціальностями галузі F «Інформаційні технології». Може бути корисним для читачів, які займаються вивченням веб-програмування.

Зміст

Лабораторна робота №1

Застосування JavaScript для розв'язання прикладних задач за допомогою консолі браузера

***Мета роботи*:** Навчитись застосовувати JavaScript для розв'язання прикладних задач за допомогою консолі браузера.

1.1 Теоретичні відомості

JavaScript (далі JS) – це мова програмування, що використовується в складі HTML-сторінок для збільшення їх функціональності та можливостей взаємодії з користувачем. JS є однією зі складових динамічного HTML. JS є мовою із динамічною типізацією (тип змінної визначається у процесі роботи (runtime) сценарію). Рушій JS усередині браузера працює послідовно (у один потік). JS-код виконується послідовно, згідно порядку включення його компонентів у текст html-файлу.

Якщо сторінка містить JS-сценарій, то з її JS-складовою можна взаємодіяти за допомогою консолі браузера. Зі сторони JS вивід у консоль забезпечується доступом до властивості log об'єкту консолі (console.log()), аргументом якої є повідомлення, змінна або об'єкт для виводу в консоль. З боку консолі можливо здійснювати виклик визначених у JS-сценарії сторінки функцій із довільними аргументами та спостерігати за їх виводом і поведінкою загалом. Для цього необхідно увести JS-код, який виконуватиме необхідні дії.

Усі математичні функції та константи, наприклад число π або тригонометричні функції є властивостями та методами вбудованого об'єкта Math. Наприклад, для застосування числа π достатньо звернутись до Math.PI.

1.2 Хід роботи

1.2.1 У створеному на GitHub репозиторії для курсу створити нову підпапку для завдання лабораторної.

1.2.2 Написати JavaScript-функцію triangle, яка розв'язує прямокутний трикутник (знаходить всі його сторони та обидва гострі кути) за двома заданими елементами і їх «типами» (катет, прилеглий кут, гіпотенуза і т.д.) і яка може бути запущена у консолі для певних значень аргументів. На початку JS-сценарій повинен вивести за допомогою console.log() інструкцію з використання.

Перелік можливих «типів» значень наведено у таблиці 1.1:

Таблиця 1.1 – Позначення «типів» наявних елементів та пояснення до них

Позначення	**Що позначає**
leg	катет
hypotenuse	гіпотенуза
adjacent angle	прилеглий до катета кут
opposite angle	протилежний до катета кут
angle	один з двох гострих кутів (коли задана гіпотенуза)

Відповідно, функція повинна приймати аргументи у такому порядку:

значення аргумента 1, «тип» аргумента 1 (див. табл. 1), значення аргумента 2, «тип» аргумента 2 (див. табл. 1).

При цьому, передбачити всеможливі порядки слідування «типів» аргументів, наприклад:

```
triangle(4, "leg", 8, "hypotenuse");
triangle(8, "hypotenuse", 4, "leg");
```

При вказанні величин кутів використовувати **градуси** (при цьому врахувати, що тригонометричні функції-методи Math оперують **радіанами**).

Вивід результатів здійснити за допомогою console.log(). При виведенні скористатись класичними математичними позначеннями сторін (*c* – гіпотенуза, *a* та *b* – катети, *alpha* – гострий кут, який лежить навпроти катета *a*, *beta* – гострий кут, який лежить навпроти катета *b*). Якщо функція успішно виконала обчислення, повернути „success”, якщо користувач увів щось неправильно (за винятком випадків некоректних значень, про що сказано у наступному абзаці), наприклад, увів «hypotenus» замість «hypotenuse» або увів несумісну пару «типів», то вивести повідомлення (console.log()) про те, що користувачу слід ще раз перечитати інструкцію та повернути „failed”.

Забезпечити перевірку на некоректність уведених значень (від’ємні або нульові аргументи, катет менший за гіпотенузу, негострі кути, тощо) при якій у випадку некоректних даних повертається рядок із відповідним повідомленням (за бажанням, можна додати вивід повідомлення у консоль).

Приклади тестових запусків наведено на рисунку 1.1:

Рисунок 1.1 – Приклади тестових запусків

1.2.3 Надати викладачеві посилання на зроблену веб-сторінку, розміщену на сервері GitHub Pages (посилання повинне бути у форматі: https://username.github.io/userrepo/userdirectory), а також, посилання на безпосередньо сам репозиторій (посилання повинне бути у форматі: https://github.com/username/userrepo).

1.2.4 Зробити висновки щодо виконаної роботи.

1.3 Зміст звіту

1. Титульна сторінка та тема роботи.
2. Мета роботи.
3. Короткі теоретичні відомості (обсягом до 1 сторінки).
4. Хід роботи (результати виконаної роботи).
5. Висновки

1.4 Контрольні запитання

1.4.1 За що відповідає JavaScript у трійці HTML, CSS, JavaScript?

1.4.2 Що зумовлено тим, що JavaScript є мовою із динамічною типізацією?

1.4.3 Як задати змінну у JavaScript?

1.4.4 Як задати функцію у JavaScript?

1.4.5 Які існують області видимості у JavaScript?

1.4.6 Що таке scope chain і які його особливості?

1.4.7 Які типи даних визначено у JavaScript?

1.4.8 Поясніть різницю між операторами «==» та «===».

Лабораторна робота №2

Застосування JavaScript для розв’язання прикладних задач обробки даних

***Мета роботи*:** Навчитись застосовувати JavaScript для розв’язання прикладних задач обробки даних.

1.1 Теоретичні відомості

JavaScript (далі JS) – це мова програмування, що використовується в складі HTML-сторінок для збільшення їх функціональності та можливостей взаємодії з користувачем. JS є однією зі складових динамічного HTML. JS є мовою із динамічною типізацією (тип змінної визначається у процесі роботи (runtime) сценарію). Рушій JS усередині браузера працює послідовно (у один потік). JS-код виконується послідовно, згідно порядку включення його компонентів у текст html-файлу.

Якщо сторінка містить JS-сценарій, то з її JS-складовою можна взаємодіяти за допомогою консолі браузера. Зі сторони JS вивід у консоль забезпечується доступом до властивості log об’єкту консолі (console.log()), аргументом якої є повідомлення, змінна або об’єкт для виводу в консоль. З боку консолі можливо здійснювати виклик визначених у JS-сценарії сторінки функцій із довільними аргументами та спостерігати за їх виводом і поведінкою загалом. Для цього необхідно увести JS-код, який виконуватиме необхідні дії.

Усі математичні функції та константи, наприклад число π або тригонометричні функції є властивостями та методами вбудованого об’єкта Math. Наприклад, для застосування числа π достатньо звернутись до Math.PI.

У JS, у зв’язку із динамічною типізацією, масиви можуть містити дані різних типів, у т.ч. функції та об’єкти поруч із примітивами. При цьому вони є об’єктами зі збереженням усіх відповідних властивостей.

Для роботи із зовнішніми бібліотеками та модулями слід оформлювати їх JS-код у вигляді Immediately Invoked Function Expression (IIFE) – функції, яка виконується одразу після того, як вона була визначена.

1.2 Хід роботи

1.2.1 У створеному на GitHub репозиторії для курсу створіть нову підпапку для завдання лабораторної.

1.2.2 Напишіть JavaScript-застосунок, який проходить по масиву імен (записаних літерами англійського алфавіту) та виводить у консоль або hello або goodbye людині з цим ім’ям. Якщо ім’я починається з літери **j** або **J**, то застосунок виводить Goodbye SomeName, інакше – Hello SomeName.

При цьому, Ваш уявний колега, якого перемістили на інший проект, уже встиг написати за Вас (ну, принаймні почав) 2 зовнішні бібліотеки, заготовку головного скрипта та html-файл (з яким, ніби не мало б виникнути проблем). Тепер Ваше завдання використати його напрацювання і довести їх до робочого стану. При цьому слід дотриматись наступних вимог, встановлених Вашим уявним тімлідом (Team lead):

- змінні speakWord у бібліотеках не повинні вийти у глобальну область видимості;
- масив заданий Вашим колегою є цілком придатним до вжитку, але теж не має вийти у глобальну область видимості (так, Ваш уявний тімлід любить оптимізоване використання пам’яті).

Також, один з Ваших уявних колег нагадав Вам дещо із початкової школи, а саме, що є такі корисні методи об’єкта типу string як, charAt() та toLowerCase(). У відповідь на запитання щодо того, що вони роблять, він відправив Вас на MDN (Mozilla Developer Network) і побажав успіхів.

1.2.3 Через декілька днів та недоспаних ночей Ваш фінальний білд (build) був показаний уявному замовнику і згідно філософії гнучкої

методології розробки ПЗ (не тої, що Waterfall) у нього виникла ідея щодо додаткового функціоналу. Ідея у тому, щоб додати до успішно реалізованого у попередньому пункті застосунку, ще один оригінальний спосіб селекціонування імен. Який саме – ваша ініціатива. Серед прикладів названо як прості варіанти, наприклад, залежність від останньої літери, так і більш складні – сума ASCII-кодів літер імен та порівняння її значення із деяким порогом. Цей функціонал можна реалізувати на уже наявному масиві або на новоствореному для підвищення демонстративності. Вивід результатів здійснити у консоль після короткої анотації застосованого підходу.

1.2.4 Надати викладачеві посилання на зроблену веб-сторінку, розміщену на сервері GitHub Pages (посилання повинне бути у форматі: https://username.github.io/userrepo/userdirectory), а також, посилання на безпосередньо сам репозиторій (посилання повинне бути у форматі: https://github.com/username/userrepo).

1.2.5 Зробіть висновки щодо виконаної роботи.

1.3 Зміст звіту

1. Титульна сторінка та тема роботи.
2. Мета роботи.
3. Короткі теоретичні відомості (обсягом до 1 сторінки).
4. Хід роботи (результати виконаної роботи).
5. Висновки

1.4 Контрольні запитання

1.4.1 За що відповідає JavaScript у трійці HTML, CSS, JavaScript?

1.4.2 Що зумовлено тим, що JavaScript є мовою із динамічною типізацією?

1.4.3 На що вказує вказівник this, розташований всередині функції, яка не є конструктором у JavaScript?

1.4.4 Яка особливість створення конструкторів у JavaScript?

1.4.5 На що вказує вказівник this, розташований всередині літерала об’єкта у JavaScript?

1.4.6 Чи можуть масиви у JavaScript містити різнотипні елементи? Чому?

1.4.7 Як дізнатись поточну довжину масиву у JavaScript?

1.4.8 Наведіть приклад розрідженого масиву та його утворення у JavaScript.

1.4.9 Що таке замикання (closure) у JavaScript?

1.4.10 Чи визначені простори імен у JavaScript?

1.4.11 Як визначається IIFE у JavaScript?

Лабораторна робота №3

Об’єктно-орієнтоване програмування у JavaScript

***Мета роботи*:** Навчитись створювати та застосовувати прийоми об’єктно-орієнтованого програмування у JavaScript.

1.1 Теоретичні відомості

Якщо у JavaScript (JS) згадується властивість об’єкта, яка ще не була визначена, то рушій JS автоматично створює цю властивість. Цей механізм не спрацьовує, якщо об’єкт попередньо не було задано (створено). До властивості об’єкта можна отримати доступ:

- за допомогою оператора «.»;
- за допомогою квадратних дужок, у яких вказаний ідентифікатор властивості у вигляді рядкового літерала (obj[“property”]).

Створення об’єкта за допомогою new Object(); передбачає оголошення змінної типу об’єкт, після чого її необхідно «наповнити» властивостями, зокрема за допомогою оператора «.» або квадратних дужок. Створення об’єкта за допомогою синтаксису літералу об’єкта реалізує оголошення змінної типу об’єкт за допомогою фігурних дужок {}, у яких можна одразу вказати усі необхідні пари властивість-значення (властивість відділяється від значення оператором «:»), які розділені оператором «,». Таким переліком пар властивість-значення реалізують ініціалізацію змінної-об’єкта.

Для оголошення конструктора використовується службове слово **constructor**. Відповідна функція буде викликана кожного разу при застосуванні оператора **new**. Класи можуть мати як властивості та методи, які викликаються з-під об’єктів класу, так і статичні властивості та методи (задаються службовим словом **static**), які викликаються лише напряму з-під класу.

Класи можуть мати ґеттери (*getters*, задаються службовим словом **get**) та сеттери (*setters*, задаються службовим словом **set**). Для вказання необхідності

успадкування від іншого класу використовується службове слово **extends**. Для виклику методу батьківського класу використовується службове слово **super**.

У JS функції є типом даних першого класу (First-Class Data Type). Це означає, що з ними можна поводитися як зі змінними або об’єктами, наприклад, передавати з одного контексту в інший, асоціювати зі змінною, передавати у якості аргумента, повертати з функції, тощо.

У JS функції є об’єктом, який має спеціальні властивості. Крім цього функції можна присвоювати довільні властивості та звертатись до них, як до властивостей об’єкта.

Звідси випливає можливість створення функцій, усередині яких створюються функції, що згодом повертаються у якості результату.

Слід пам’ятати, що якщо біля ідентифікатора функції знаходяться дужки (оператор виклику), то функція буде виконана, якщо ж вони відсутні – функція буде розглядатись як об’єкт (наприклад, при передачі у якості аргумента).

1.2 Хід роботи

1.2.1 У створеному на GitHub репозиторії для курсу створіть нову підпапку для завдання лабораторної.

1.2.2 У завданнях 1.2.3 – 1.2.10 роботи **ЗАБОРОНЕНО** використовувати синтаксис класів, уведений у ES6.

1.2.3 Створити за допомогою new Object() об’єкт car1 із наступними властивостями, які наведені у таблицях 3.1, 3.2:

Таблиця 3.1 – Властивості об’єкта car1

Ім’я властивості	**Значення властивості**
color	вказати довільний колір англійською мовою
maxSpeed	вказати довільне ціле число

driver	об’єкт із властивостями, наведеними у таблиці 3.2
tuning	true
number of accidents	0

Таблиця 3.2 – Властивості об’єкта driver (для car1)

Ім’я властивості	**Значення властивості**
name	Рядок, який містить ім’я та прізвище виконавця лабораторної роботи
category	«C»
personal limitations	«No driving at night»

1.2.4 Створити за допомогою синтаксису літерала об’єкта об’єкт car2 із наступними властивостями, які наведені у таблицях 3.3, 3.4:

Таблиця 3.3 – Властивості об’єкта car2

Ім’я властивості	**Значення властивості**
color	вказати довільний колір англійською мовою
maxSpeed	вказати довільне ціле число
driver	об’єкт із властивостями, наведеними у таблиці 3
tuning	false
number of accidents	2

Таблиця 3.4 – Властивості об’єкта driver (для car2)

Ім’я властивості	**Значення властивості**
name	Рядок, який містить ім’я та прізвище виконавця практичної роботи
category	«B»
personal limitations	null

1.2.5 До створеного об’єкта car1 додати метод drive, який виводить у консоль повідомлення «I am not driving at night». Продемонструвати роботу створеного методу.

1.2.6 До створеного об’єкта car2 додати метод drive, який виводить у консоль повідомлення «I can drive anytime». Продемонструвати роботу створеного методу.

1.2.7 Створити конструктор для «класу» Truck, який приймає аргументи з таблиці 3.5 та присвоює їх значення властивостям із відповідними іменами:

Таблиця 3.5 – Аргументи конструктора Truck

Ім’я властивості	**Значення властивості**
color	Довільний колір англійською мовою
weight	Довільне ціле число
avgSpeed	Довільне дійсне число
brand	Довільний рядок із маркою машини
model	Довільний рядок із моделлю машини

1.2.8 За допомогою prototype створити для «класу» Truck «статичний» метод AssignDriver, який присвоює об’єкту «класу» Truck властивість driver у вигляді об’єкта і приймає аргументи з таблиці 3.6 та присвоює їх значення властивостям із відповідними іменами об’єкта-властивості driver:

Таблиця 3.6 – Аргументи методу AssignDriver

Ім’я властивості	**Значення властивості**
name	Рядок, який містить ім’я та прізвище виконавця практичної роботи
nightDriving	Булеве значення
experience	довільне ціле число

1.2.9 Додати у конструктор класу Truck створення методу trip, який:

- У випадку відсутності у об’єкта властивості driver повертає у консоль «No driver assigned»;
- Якщо властивість driver присутня, то метод виводить у консоль повідомлення із такими компонентами:
    - «Driver [name]»;
    - Якщо nightDriving має істинне значення, то додати до повідомлення «drives at night», якщо хибне – «does not drive at night»;
    - Додати до повідомлення «and has [experience] years of experience».

1.2.10 Створити 2 об’єкти «класу» Truck, у один з яких додати за допомогою методу AssignDriver «водія», для якого nightDriving матиме істинне значення, а для іншого – хибне. Продемонструвати роботу методу trip для обох об’єктів.

1.2.11 У завданнях 1.2.12 – 1.2.24 роботи **слід** використовувати механізм класів, уведений у ES6.

1.2.12 Створити клас Square (квадрат) із властивостями, які наведені у таблиці 3.7:

Таблиця 3.7 – Властивості класу Square

Ім’я властивості	**Значення властивості**
a	ціле число (сторона квадрата)

1.2.13 Визначити для класу Square конструктор, який створює об’єкт (квадрат) із заданою стороною a, яка передається конструктору в якості аргумента.

1.2.14 Визначити для класу Square статичний метод help, який виводитиме у консоль стислу довідку про квадрат та його особливості, як геометричної фігури.

1.2.15 Визначити для класу Square методи:

- length, який виводитиме у консоль суму довжин сторін геометричної фігури-чотирикутника;
- square, який виводитиме у консоль площу геометричної фігури-чотирикутника;
- info, який виводитиме у консоль (з відповідними описами) повну характеристику геометричної фігури-чотирикутника:
  - довжини всіх 4 сторін;
  - величини всіх 4 кутів;
  - суму довжин сторін;
  - площу.

1.2.16 Створити шляхом успадкування від класу Square клас Rectangle (прямокутник), із властивостями, які наведені у таблиці 3.8:

Таблиця 3.8 – Властивості класу Rectangle

Ім’я властивості	**Значення властивості**
a	ціле число (довжина прямокутника)
b	ціле число (ширина прямокутника)

1.2.17 Перевизначити для класу Rectangle успадковані:

- конструктор;
- статичний метод help;
- метод length;
- метод square;
- метод info.

1.2.18 Створити шляхом успадкування від класу Square клас Rhombus (ромб), із властивостями, які наведені у таблиці 3.9:

Таблиця 3.9 – Властивості класу Rhombus

Ім’я властивості	**Значення властивості**
a	ціле число (сторона ромба)
alpha	ціле число (тупий кут ромба у градусах)
beta	ціле число (гострий кут ромба у градусах)

1.2.19 Перевизначити для класу Rhombus успадковані:

- конструктор;
- статичний метод help;
- метод length;
- метод square;
- метод info.

1.2.20 Створити шляхом успадкування від класу Rectangle або Rhombus клас Parallelogram (паралелограм), із властивостями, які наведені у таблиці 3.10:

Таблиця 3.10 – Властивості класу Parallelogram

Ім’я властивості	**Значення властивості**
a	ціле число (довжина паралелограма)
b	ціле число (ширина паралелограма)
alpha	ціле число (тупий кут паралелограма у градусах)
beta	ціле число (гострий кут паралелограма у градусах)

1.2.21 Перевизначити для класу Parallelogram успадковані:

- конструктор;
- статичний метод help;
- метод length;
- метод square;
- метод info.

1.2.22 Для того з класів Rectangle або Rhombus, на основі якого НЕ був створений клас Parallelogram визначити ґеттери та сеттери для кожної з його властивостей.

1.2.23 Реалізувати виклик статичного методу help для кожного з класів Square, Rectangle, Rhombus та Parallelogram.

1.2.24 Створити по одному об’єкту для класів Square, Rectangle, Rhombus та Parallelogram з довільними у допустимих межах значеннями властивостей. Для кожного зі створених об’єктів реалізувати виклик методу info.

1.2.25 Створити функцію Triangular, яка приймає три цілочислові аргументи (a, b, c) і повертає у місце виклику об’єкт із властивостями, які наведені у таблиці 3.11:

Таблиця 3.11 – Властивості об’єкта, який повертається з Triangluar

Ім’я властивості	**Значення властивості**
a	ціле число (сторона трикутника)
b	ціле число (сторона трикутника)
c	ціле число (сторона трикутника)

У якості значень за замовчуванням використати значення сторін (3, 4, 5). При створенні об’єктів рекомендовано використати destructuring assignment.

1.2.26 За допомогою функції Triangular створити та вивести у консоль 3 об’єкти, один з яких буде містити значення за замовчуванням.

1.2.27 Створити функцію PiMultiplier, яка приймає у якості аргумента дійсне число і повертає у місце виклику функцію, яка повертає результат множення числа π на число, яке було задане аргументом функції PiMultiplier у момент створення функції-результату.

1.2.28 За допомогою функції PiMultiplier створити три функції, перша з яких множить число π на 2, друга – на $\frac{2}{3}$, а третя – ділить π на 2. Вивести у консоль результати роботи створених функцій.

1.2.29 Створити функцію Painter, яка приймає у якості аргумента рядок (колір) і повертає у місце виклику функцію, яка виводить у консоль рядок, який містить колір, заданий при створенні функцією Painter та вміст властивості type об’єкта, який їй передається у якості аргумента (якщо такої властивості немає, то функція повинна виводити рядок «No ‘type’ property occurred!»).

1.2.30 За допомогою функції Painter створити функції PaintBlue, PaintRed та PaintYellow, які будуть «фарбувати» об’єкти у, відповідно, синій, червоний та жовтий кольори.

1.2.31 Продемонструвати роботу функцій PaintBlue, PaintRed та PaintYellow на кожному з об’єктів з таблиці 3.12:

Таблиця 3.12 – Тестові об’єкти для функцій PaintBlue, PaintRed та PaintYellow

Об’єкт 1		**Об’єкт 2**		**Об’єкт 3**	
Ім’я властивості	**Значення властивості**	**Ім’я властивості**	**Значення властивості**	**Ім’я властивості**	**Значення властивості**
maxSpeed	280	type	Truck	maxSpeed	180
type	Sportcar	avg speed	90	color	purple
color	magenta	load capacity	2400	isCar	true

1.2.32 Надати викладачеві посилання на зроблену веб-сторінку, розміщену на сервері GitHub Pages (посилання повинне бути у форматі: https://username.github.io/userrepo/userdirectory), а також, посилання на безпосередньо сам репозиторій (посилання повинне бути у форматі: https://github.com/username/userrepo).

1.2.33 Зробіть висновки щодо виконаної роботи.

1.3 Зміст звіту

1. Титульна сторінка та тема роботи.
2. Мета роботи.
3. Короткі теоретичні відомості (обсягом до 1 сторінки).
4. Хід роботи (результати виконаної роботи).
5. Висновки

1.4 Контрольні запитання

1.4.1 Як реалізовується успадкування у JS?

1.4.2 Які способи реалізації ООП є у JS?

1.4.3 Які способи реалізації успадкування у JS Ви знаєте?

1.4.4 Як задати ґеттер у JS?

1.4.5 Як задати сеттер у JS?

1.4.6 Що таке factory function у JS?

1.4.7 Які особливості написання конструкторів у JS?

Лабораторна робота №4

Робота із вводом користувача у JavaScript

***Мета роботи*:** Навчитись опрацьовувати ввід користувача у JavaScript.

1.1 Теоретичні відомості

DOM (Document Object Model) – об’єктна модель документа є програмним інтерфейсом для HTML-документів, який забезпечує структуроване представлення документа (його деревовидної структури) і визначає спосіб реалізації доступності структури документа для програми з метою зміни самої структури документа, його стилю та вмісту.

DOM забезпечує представлення документа у вигляді структурованої групи вузлів та об’єктів, які мають властивості та методи. Відповідно, вона по суті зв’язує веб-сторінки зі скриптами та серверними мовами програмування.

Основним об’єктом (відправною точкою) DOM є об’єкт document. Його можна знайти у переліку властивостей глобального об’єкта, наприклад для браузера:

window.document

У об’єкті document знаходиться увесь вміст веб-сторінки (включно зі службовою інформацією). У файлі JS-скрипта, як і у консолі браузера, доступ до об’єкта document можливий безпосередньо з глобальної області видимості без потреби звертання до глобального об’єкта (наприклад, window). У випадку HTML-документу, document є об’єктом «класу» HTMLDocument.

Для отримання доступу до елементів застосовують методи об’єкту document:

- getElementById(“id”);
- getElementsByTagName(“tag”);
- getElementsByClassName(“class”);
- querySelector(“CSS_Selector”);
- та ін.

Можливе, також, комбінування цих методів.

При застосуванні наведених методів слід пам’ятати, що скрипт, включений до HTML-документа у межах тега <head>, ще «не бачить» структурних елементів, у зв’язку з чим рекомендується включати скрипти у HTML-документи перед закриваючим тегом </body>.

Для отримання доступу до текстового вмісту елемента застосовують властивість textContent.

Для отримання доступу до HTML-коду всередині елемента застосовують властивість innerHTML. Текстовий рядок, переданий у якості значення цієї властивості буде пропущений через синтаксичний аналізатор (парсер (parser)) на предмет виявлення та вбудування наявного HTML-коду у сторінку.

1.2 Хід роботи

1.2.1 У створеному на GitHub репозиторії для курсу створити нову підпапку для завдання лабораторної.

1.2.2 Створити браузерну гру-клікер, у якій користувачеві необхідно здійснити клацання по графічному примітиву (квадрат, коло) за певний проміжок часу, після чого примітив повинен переміститися за випадковими координатами і відлік часу на клацання починається заново. У випадку вичерпання часу на клацання користувач повинен отримати повідомлення про завершення гри із кількістю очок, які він набрав за вдалі клацання. Гра повинна надавати три рівні складності, які визначатимуть час на одне клацання, розмір примітиву та розкид координат переміщення. Передбачити наявність видимого таймера для відліку часу на клацання та можливість вибрати принаймні 3 різні кольорові варіанти оформлення графічного примітиву.

1.2.3 Надати викладачеві посилання на зроблену веб-сторінку, розміщену на сервері GitHub Pages (посилання повинне бути у форматі: https://username.github.io/userrepo/userdirectory), а також, посилання на

безпосередньо сам репозиторій (посилання повинне бути у форматі: https://github.com/username/userrepo).

1.2.4 Зробити висновки щодо виконаної роботи та методів і особливостей роботи із вводом користувача у JS.

1.3 Зміст звіту

1. Титульна сторінка та тема роботи.
2. Мета роботи.
3. Короткі теоретичні відомості (обсягом до 1 сторінки).
4. Хід роботи (результати виконаної роботи).
5. Висновки

1.4 Контрольні запитання

1.4.1 Що таке DOM?

1.4.2 Як називається глобальний об’єкт у DOM?

1.4.3 Як за допомогою JavaScript отримати доступ до вузлів DOM?

1.4.4 Що таке обробник події?

1.4.5 Які способи прив’язування обробників подій до вузлів DOM Вам відомі? Які переваги і недоліки кожного з них?

1.4.6 Коли спрацьовує подія DOMContentLoaded?

Лабораторна робота №5

Застосування JavaScript для реалізації серверної взаємодії на основі Ajax

***Мета роботи*:** Навчитись застосовувати JavaScript для реалізації серверної взаємодії на основі Ajax.

1.1 Теоретичні відомості

Традиційний процес роботи Веб полягає у відправці HTTP-запиту (наприклад, по натисканню кнопки) на сервер і отримання нової сторінки у відповідь, навіть якщо її вміст на 99% ідентичний сторінці, з якої відправлено запит.

Процес роботи Ajax відрізняється тим, що HTTP-запит відправляється JS, а у відповідь приходить невелика порція даних, яка динамічно (без перезавантаження сторінки) включається у поточну сторінку.

JSON (JavaScript Object Notation) – це легкий (lightweight) формат обміну даними, який забезпечує просте текстуальне подання даних.

З точки зору синтаксису, JSON є підмножиною JS синтаксису літерала об’єкта з наступними уточненнями:

- ідентифікатори властивостей повинні братись у подвійні лапки;
- рядкові літерали повинні братись у подвійні лапки.

Решта синтаксичних правил JSON аналогічні правилам JS синтаксису літерала об’єкта.

1.2 Хід роботи

1.2.1 У створеному на GitHub репозиторії для курсу створити нову підпапку для завдання лабораторної.

1.2.2 Створити односторінковий веб-застосунок, який здійснюватиме запит на сервер за інформацією, яка міститься у форматі JSON, щодо

відображення переліку умовних товарів чи послуг деяких категорій шляхом динамічного завантаження оновленого контенту сторінки.

На головній сторінці у довільному вигляді мають міститися посилання на категорії каталогу (за їх назвами, які отримуються із відповідного JSON-файлу) умовних товарів чи послуг. Кількість категорій у каталозі має бути рівна 3-4, кожна з яких містить по 4-6 позицій. Після переліку категорій каталогу повинне знаходитись посилання на категорію “Specials”, при переході за яким користувачу відображатиметься вміст випадкової категорії каталогу. Для реалізації випадкового вибору категорії можна скористатись функцією Math.random().

У верхній частині сторінки повинні знаходитись посилання:

- «Home» або «Додому», яке посилатиметься на саму сторінку (фактично, вестиме до її оновлення);
- «Catalog» або «Каталог», яке завантажуватиме каталог (без оновлення сторінки).

Кожна категорія у JSON-файлі зі списком категорій повинна містити поля: id (унікальний номер), name (назва категорії), shortname (скорочена або службова назва), notes (примітки до категорії, можуть бути або порожні, або містити текст-заповнювач (Lorem ipsum)).

Для кожного товару або послуги повинне відображатись невелике (до 200×200 пікселів) зображення, найменування, опис та ціна у довільній валюті. Ці відомості завантажуються із JSON-файлу відповідної категорії (такі файли можуть бути названі, наприклад, за скороченою назвою категорії), який для кожного товару чи послуги містить такі поля: id (унікальний номер), name (назва товару чи послуги), shortname (скорочена або службова назва товару чи послуги), description (опис товару чи послуги, який може містити текст-заповнювач (Lorem ipsum)), price (ціна у довільній валюті).

При завантаженні вмісту категорії, перед переліком товарів чи послуг має бути розміщений заголовок, який міститиме назву категорії (її

можна отримувати, наприклад із JSON-файлу категорії, попередньо включивши її у нього у довільній формі).

Зображення товарів чи послуг можна згенерувати, наприклад, за допомогою ресурсу https://place-hold.it/.

Для виконання завдання **забороняється** використовувати функціональні front-end фреймворки (Angular, React, Vue.js і т.д.), проте **можна** застосовувати front-end фреймворки для стилізації (Bootstrap, Tailwind CSS і т.д.).

Візуальне оформлення та укладання елементів сторінки (кількість колонок і т.д.) – довільні за умови дотримання усіх попередніх вимог.

Усі маніпуляції з контентом повинні відбуватись у межах **ОДНІЄЇ** сторінки (index.html).

1.2.3 Надати викладачеві посилання на зроблену веб-сторінку, розміщену на сервері GitHub Pages (посилання повинне бути у форматі: https://username.github.io/userrepo/userdirectory), а також, посилання на безпосередньо сам репозиторій (посилання повинне бути у форматі: https://github.com/username/userrepo).

1.2.4 Зробити висновки щодо виконаної роботи та особливостей роботи Ajax при серверній взаємодії.

1.3 Зміст звіту

1. Титульна сторінка та тема роботи.
2. Мета роботи.
3. Короткі теоретичні відомості (обсягом до 1 сторінки).
4. Хід роботи (результати виконаної роботи).
5. Висновки

1.4 Контрольні запитання

1.4.1 За що відповідає JavaScript у трійці HTML, CSS, JavaScript?

1.4.2 На що вказує вказівник this, розташований всередині функції, яка не є конструктором у JavaScript?

1.4.3 На що вказує вказівник this, розташований всередині літерала об’єкта у JavaScript?

1.4.4 Чи можу масив у JavaScript містити різнотипні елементи? Чому?

1.4.5 Що таке замикання (closure) у JavaScript?

1.4.6 Як визначається IIFE у JavaScript?

1.4.7 Як працює http-запит?

1.4.8 Поясніть механізм роботи Ajax.

1.4.9 Що таке JSON та чи є він тотожним літералу об’єкта у JavaScript?

Рекомендовані джерела

1. Head First. Програмування на JavaScript (пер. з англ.) / Е. Фрімен, Е. Робсон. – Фабула, 2022. – 672 с.
2. MDN Web Docs [Електронний ресурс] / Режим доступу: https://developer.mozilla.org/en-US/
3. ECMA-262 - Ecma International (актуальний стандарт мови JavaScript) [Електронний ресурс] / Режим доступу: https://ecma-international.org/publications-and-standards/standards/ecma-262/
4. Сучасний підручник з JavaScript [Електронний ресурс] / Режим доступу: https://uk.javascript.info/
5. Гринчак О. В., Моцик Р. В. Веб-технології та дизайн // Вісник Хмельницького національного університету. – 2021. – № 1(293). – С. 22-26. https://doi.org/10.31891/2307-5732-2021-293-1-22-26
6. Веб-технології та веб-дизайн : навч. посібник / О. Г. Трофименко, О. Б. Козін, О. В. Задерейко, О. Є. Плачінда. – Одеса : Фенікс, 2019. – 284 с.
7. JavaScript Підручник. W3Schools українською [Електронний ресурс] / Режим доступу: https://w3schoolsua.github.io/js/