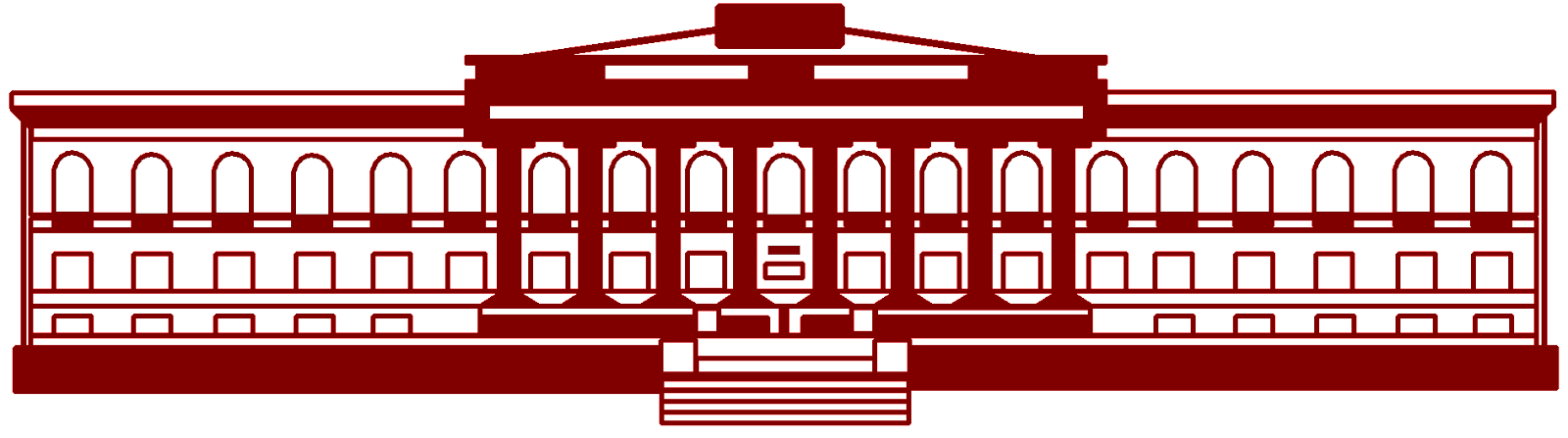
**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**імені ТАРАСА ШЕВЧЕНКА**

Факультет інформаційних технологій

**Кафедра прикладних інформаційних систем**



**Звіт**

до виконання лабораторної роботи №6

з дисципліни «**Веб-технології і програмування сайтів**»

на тему:

« Знайомство з підключенням та роботою сторонніх бібліотек у мові Java Script »

**Виконано:**

студ. групи ПП- 31, підгрупа 2

Шкандюк Анною Леонідівною

**Перевірено:**

Пирог М.В.

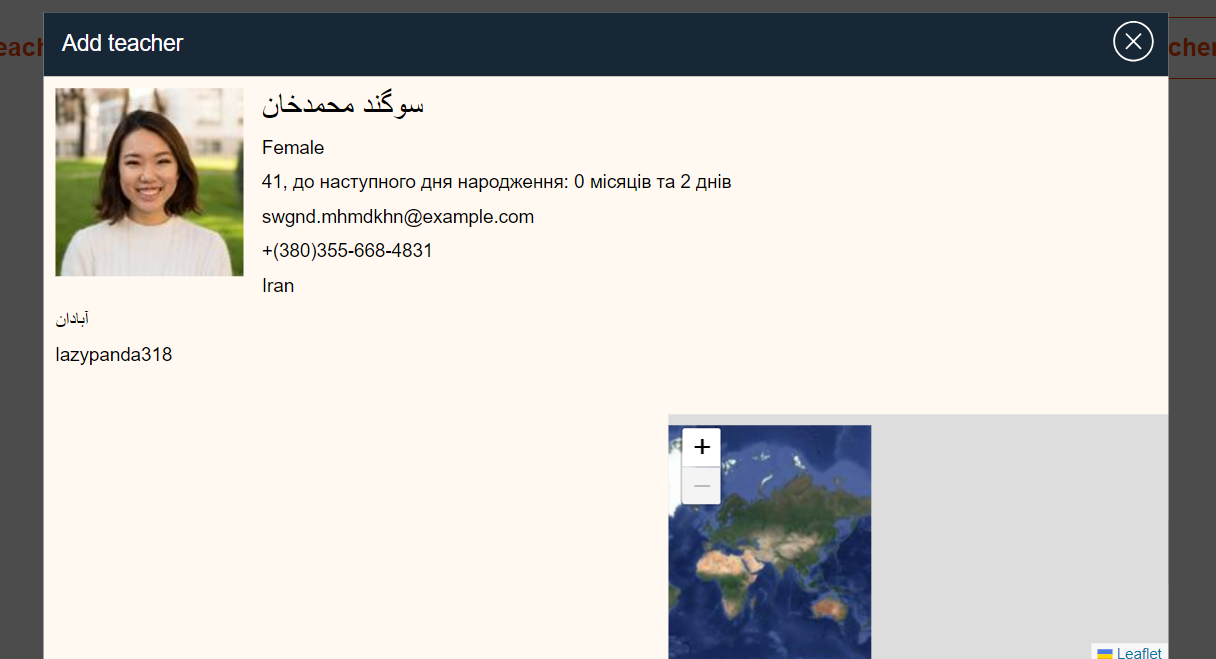
**Київ – 2023**

1. **Мета роботи:**

 Отримати навички роботи зі сторонніми бібліотеками.

1. **Завдання та їх виконання:**

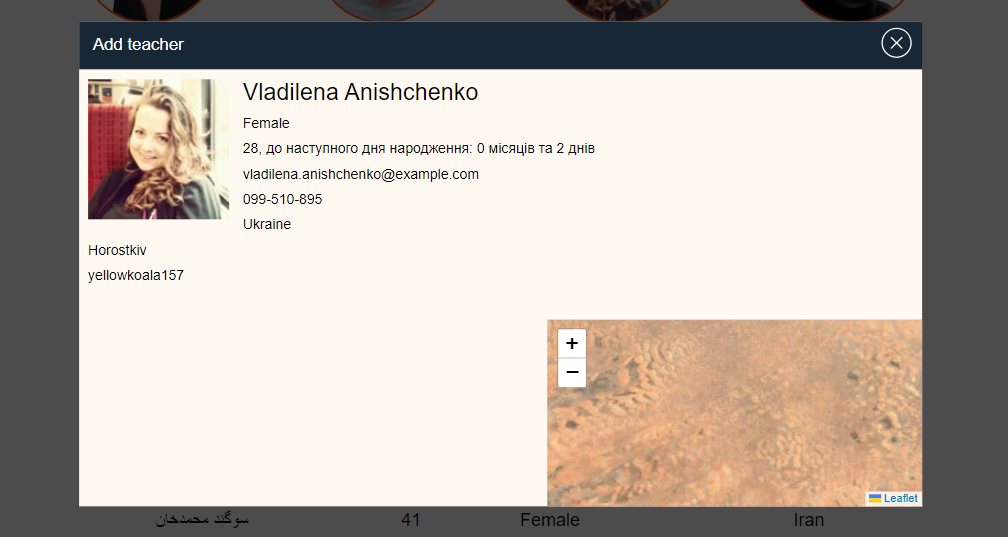
**Завдання 1**. Підключити до сторінки бібліотеку *leaflet*. Додати до картки викладача розташування, використовуючи координати надані в данних.



**Завдання 2.** Підключити до сторінки бібліотеку *chart.js*. Замінити таблиці зі статистикою на piechart.

**Завдання 3**.  Підключити до сторінки бібліотеку *lodash*. Використовуючи її відрефакторити функції (мапинг, валідація, сортинг, фільтерінг та пошук).

**Завдання 4**. Підключити до сторінки бібліотеку *day.js*. Додати до сторінки користувача, поле, яке буде показувати, скільки днів залишилось до наступного дня народження.



1. **Висновки до роботи**

За результатами проробленої роботи можемо сказати, що отримали  навички роботи зі сторонніми бібліотеками.

1. **Контрольні запитання**
2. В чому полягає різниця між бібліотекою і фреймворком

Бібліотека (library) та фреймворк (framework) - це два терміни, які використовуються у програмуванні для опису різних підходів та інструментів для розробки програмного забезпечення. Вони мають різні ролі та характеристики:

Бібліотека (Library):

Бібліотека - це збірка функцій, класів та ресурсів, які можна використовувати для певних завдань або операцій у коді. Бібліотеки зазвичай зосереджуються на вирішенні конкретних завдань і надають інструменти для виконання конкретних операцій. Ви можите вибрати та використовувати лише ті частини бібліотеки, які вам потрібні.

Приклади бібліотек: jQuery (для роботи з DOM), lodash (функціональність JavaScript), requests (для здійснення HTTP-запитів у Python).

Фреймворк (Framework):

Фреймворк - це велика структура, яка надає базову архітектуру для вашого додатка або програми. Фреймворки надають вже готові структури, рішення та правила, які потрібно дотримуватися. Ви побудовуєте свій код відповідно до вимог фреймворку, який надає готові рішення для ряду задач.

Приклади фреймворків: Django (веб-фреймворк для Python), Ruby on Rails (веб-фреймворк для Ruby), Angular (фронтенд-фреймворк для JavaScript).

Основна різниця:

1. Керування потоком виконання: Бібліотеки зазвичай не визначають структуру програми та поток виконання. Ви вибираєте, коли та як використовувати функції бібліотеки. Фреймворки надають структуру та визначають потік виконання, вимагаючи вас слідувати певним правилам.

2. Роздільність відповідальностей: Бібліотеки зазвичай не обмежують вашу свободу щодо організації коду. Фреймворки зазвичай пропонують вам певну архітектурну модель та роздільність відповідальностей між компонентами.

3. Залежність: Бібліотеки можуть бути використані додатково до вже існуючого коду. Фреймворки можуть вимагати вас створити код, відповідний їхнім структурам та конвенціям.

1. На які параметри необхідно звертати увагу, коли обираєш JS бібліотеку

Обираючи JavaScript бібліотеку, важливо враховувати різні параметри, щоб забезпечити, що вона відповідає вашим потребам і розвитку проекту. Ось кілька ключових параметрів, на які варто звернути увагу:

1. Функціональність: Перш за все, переконайтеся, що бібліотека має необхідну функціональність для вашого проекту. Які завдання ви плануєте виконувати з її допомогою? Які функції вам потрібні?

2. Спільнота та підтримка: Велика та активна спільнота розробників є дуже важливим фактором. Чим більше розробників використовує бібліотеку, тим більше ресурсів, документації та підтримки ви знайдете.

3. Документація: Якість документації важлива для швидкого та ефективного вивчення бібліотеки. Впевніться, що є як офіційна документація, так і приклади використання.

4. Актуальність та релізи: Перевірте, коли останній раз випускали оновлення бібліотеки. Актуальна версія має більше шансів підтримувати останні технологічні та безпекові стандарти.

5. Розмір бібліотеки: Великі бібліотеки можуть впливати на завантаження та продуктивність вашого веб-додатка. Оберіть бібліотеку, яка надає тільки ті функції, які вам дійсно потрібні.

6. Залежності та конфлікти: Перевірте, які інші бібліотеки вимагає обрана вами бібліотека, і чи може це викликати конфлікти або перевантаження.

7. Проектні цілі та філософія: Важливо враховувати, чи відповідає філософія бібліотеки вашим проектним цілям. Наприклад, якщо ви працюєте над простим проектом, можливо, вам підійде легка бібліотека. Якщо ж ви створюєте складний односторінковий додаток, можливо, ви виберете більший фреймворк.

8. Тестування та якість коду: Перевірте, чи бібліотека має гарний стиль коду, тестові приклади та наявність валідної ліцензії.

1. Для чого призначені бібліотеки *leaflet, chart.js, lodash, day.js.*

Бібліотеки `leaflet`, `chart.js`, `lodash` та `day.js` призначені для різних цілей та завдань у розробці програмного забезпечення. Ось їхнє призначення:

1. Leaflet:

Leaflet - це бібліотека для роботи з інтерактивними картами на веб-сайтах. Вона дозволяє легко створювати та взаємодіяти з картами, додавати шари, маркери, полігони, полілінії, інтерактивні відомості та інше. Leaflet підтримує підключення до різних картографічних джерел (наприклад, OpenStreetMap) та має велику спільноту розробників.

2. Chart.js:

Chart.js - це бібліотека для створення інтерактивних діаграм, графіків та графіків на веб-сайтах. Вона дозволяє легко візуалізувати дані в різних форматах, таких як лінійні графіки, стовпчасті діаграми, кругові діаграми тощо. Chart.js надає багато налаштувань для налаштування вигляду та поведінки графіків.

3. Lodash:

Lodash - це бібліотека, яка надає набір корисних утиліт для роботи з даними та різними операціями в JavaScript. Вона надає функції для роботи з рядками, масивами, об'єктами, функціональним програмуванням, маніпулюванням даними та іншими операціями. Lodash допомагає зменшити кількість коду та спростити роботу з даними.

4. Day.js:

Day.js - це легка та невелика бібліотека для роботи з датами та часом в JavaScript. Вона надає простий та зручний спосіб для парсингу, форматування, відображення різних форматів дати та часу, а також для виконання математичних операцій над ними. Day.js може бути використана для роботи зі звичайними датами, часовими розривами, відображенням різних годинних поясів тощо.

1. Які основні проблеми вирішує module bundler

Module bundler (бандлер модулів) - це інструмент, який вирішує декілька проблем, пов'язаних з організацією та управлінням JavaScript-кодом у проектах, особливо тих, які використовують модульну структуру коду. Основні проблеми, які вирішуються за допомогою бандлера модулів:

1. Розділення на модулі: Багато сучасних веб-додатків складаються з великої кількості файлів, які мають різні функції та компоненти. Бандлери модулів допомагають розділити код на окремі модулі, що полегшує організацію та підтримку проекту.

2. Залежності між модулями: Коли код поділено на модулі, виникає питання залежностей між ними. Бандлери допомагають визначити ці залежності та зберігають потрібний порядок завантаження модулів, щоб уникнути конфліктів.

3. Зменшення кількості запитів: Без бандлера модулів кожен модуль завантажується окремо, що може призвести до багатьох запитів до сервера. Бандлери дозволяють об'єднувати модулі в один або кілька бандлів, зменшуючи кількість запитів.

4. Оптимізація розміру коду: Бандлери можуть виконувати різні оптимізації для зменшення розміру коду, такі як зведення зайвих проміжних змінних, видалення не використовуваних частин коду тощо.

5. Робота з різними типами файлів: Бандлери підтримують роботу з різними типами файлів, такими як JavaScript, CSS, зображення, шрифти тощо. Вони можуть об'єднувати та обробляти ці файли, забезпечуючи їхню правильну підключеність та використання.

6. Управління версіями: Бандлери можуть допомагати управляти версіями модулів та бібліотек, забезпечуючи їхню актуальність та сумісність.

7. Забезпечення сумісності: Бандлери можуть автоматично вирішувати питання сумісності між різними версіями модулів, що допомагає уникнути конфліктів.

Загалом, бандлери модулів допомагають зробити розробку веб-додатків більш організованою, ефективною та легшою для управління.

1. Назвіть основні характеристики Webpack

Webpack - це один з найпопулярніших бандлерів модулів для розробки веб-додатків. Він має багато характеристик, які полегшують організацію та оптимізацію роботи з JavaScript-кодом та іншими ресурсами. Основні характеристики Webpack включають:

1. Модульна система: Webpack дозволяє використовувати модульну систему JavaScript (CommonJS, ES6, AMD та ін.) для розділення коду на окремі модулі. Це спрощує організацію та підтримку проекту.

2. Збірка бандлів: Webpack об'єднує ваші модулі та їх залежності в один або кілька бандлів. Це допомагає зменшити кількість запитів до сервера та поліпшити продуктивність.

3. Лоадери: Webpack дозволяє використовувати лоадери для обробки різних типів файлів (наприклад, JavaScript, CSS, зображення) перед їх збіркою в бандл. Це дозволяє використовувати різні технології та ресурси в проекті.

4. Плагіни: Webpack підтримує систему плагінів, яка дозволяє вам виконувати різні операції під час збірки бандлів. Це може бути оптимізація, згортання, зміна змінних середовища тощо.

5. Code Splitting: Webpack підтримує розділення коду (code splitting), що дозволяє завантажувати лише ту частину коду, яка потрібна на даній сторінці. Це поліпшує швидкість завантаження та продуктивність.

6. Hot Module Replacement (HMR): HMR - це функція, яка дозволяє змінювати код на живому сайті без перезавантаження сторінки. Це допомагає прискорити процес розробки та перевірки змін.

7. Розширені конфігурації: Webpack надає гнучку систему конфігурацій, яка дозволяє налаштовувати різні аспекти збірки, оптимізації та роботи з ресурсами.

8. Підтримка ES6+: Webpack дозволяє використовувати сучасні можливості JavaScript, такі як ES6+ синтаксис, і транспілювати їх у більш старі версії JavaScript для підтримки різних браузерів.

9. Управління зображеннями та шрифтами: Webpack дозволяє оптимізувати, компресувати та управляти зображеннями, а також підключати шрифти в проекті.

10. Режим розробки та продакшн: Webpack надає можливість налаштовувати різні режими роботи - для розробки (development) та для випуску (production), залежно від потреб проекту.

Загалом, Webpack дозволяє створювати, оптимізовувати та керувати процесом збірки веб-додатків, полегшуючи роботу розробників та підвищуючи продуктивність проектів.

1. В чому різниця і які задачі виконує Webpack loader і Webpack plugin

Webpack Loaders та Plugins - це ключові компоненти системи збірки Webpack, але вони виконують різні завдання у процесі обробки та збірки ресурсів. Ось їхні різниці та задачі:

Webpack Loaders:

Loaders - це обробники, які використовуються Webpack для обробки різних типів файлів під час збірки бандлу. Вони зчитують дані з файлів та виконують різні дії над цими даними, такі як трансформація коду, компіляція препроцесорів (наприклад, Sass або Less), оптимізація зображень тощо. Основні завдання Loaders:

1. Трансформація коду: Loaders можуть перетворювати код з одного формату в інший, наприклад, транслювати сучасний JavaScript (ES6+) в більш старий формат для підтримки різних браузерів.

2. Обробка CSS та препроцесорів: Loaders можуть обробляти CSS-файли, додавати префікси, оптимізовувати, компілювати Sass або Less у CSS тощо.

3. Оптимізація зображень: Loaders можуть оптимізувати зображення, стискаючи їх розмір, а також перетворювати їх у формати, які підтримують різні браузери.

Webpack Plugins:

Plugins - це додаткові інструменти, які використовуються для розширення можливостей Webpack під час збірки. Вони дозволяють виконувати складні операції, оптимізацію, генерацію файлів та інші дії. Основні завдання Plugins:

1. Генерація HTML: Багато проектів потребують HTML-файли з посиланнями на збіркові бандли. Плагіни можуть автоматично генерувати HTML-файли з посиланнями на бандли.

2. Оптимізація: Плагіни можуть виконувати оптимізації коду, зображень та інших ресурсів для поліпшення продуктивності.

3. Мініфікація: Плагіни можуть здійснювати мініфікацію (стиснення) JavaScript, CSS та HTML коду для зменшення розміру бандлу.

4. Генерація файлів: Плагіни можуть генерувати додаткові файли, такі як manifest файл для відстеження версій.

5. Запуск зовнішніх інструментів: Плагіни можуть викликати зовнішні інструменти, такі як лінтери або тестові середовища.