Лабораторна робота 6. Використання і створення API

Завдання: Підключити зовнішній сервіс до свого рішення. Сервіс можна використовувати існуючий або створити власний. Приклади зовнішніх сервісів наведені в Додатку.

Вправа 6.1. Обрати зовнішній сервіс. і описати сценарій його використання. Створити в GitHub репозіторій з описом сценарію використання API зовнішнього застосування.

Вправа 6.2. Реалізувати на будь-якій мові програмування виклик API зовнішнього застосування і візуалізацію відповіді. Додати в GitHub код і опис прикладу застосування API.

Вправа 6.3. Реалізувати на будь-якій мові програмування http-сервер, організувати звертання до нього і отримати відповідь у браузері. Звертання має містити Ваш логін у Moodle. Відповідь має містити Ваші особисті дані (прізвище, ім'я, курс, група).

Теоретична частина

Приклад WEB-сервісу прогнозування погоди

Розглянемо доступ до зовнішнього API на прикладі WEB-сервісу прогнозування погоди *OpenWeatherMap*. Документація по його API знаходиться тут.

- Спершу треба отримати вільний логін тут. Через півгодини, максимум через дві API Key активується і можна користуватись сервісом. Можна подивитись значення API key на сторінці вашого логіну до сервісу https://home.openweathermap.org/api_keys .
- Виконайте один з прикладів виклику цього API. Наприклад, запит погоди по Id міста, де заданий параметр *appid*, який дорівнює API Key.

Id міста можна отримати з файлу city.list.json.gz по посиланню http://bulk.openweathermap.org/sample/ .

В підсумку запит

http://api.openweathermap.org/data/2.5/weather?id=703448&appid=4f670455dced9331d1cdf2d9e55ba4d1

генерує відповідь


```
{"coord":{"lon":30.52,"lat":50.43},"weather":[{"id":804,"main":"Clouds","description":"overcast
clouds","icon":"04d"}],"base":"stations","main":{"temp":276.58,"feels_like":269.53,"temp_min":276.15,"temp_max":277.15,"pressure":1002,"humidity":72},"visibility":10000,"wind":{"speed":7,"deg":280},"clouds":{"all":90},"dt":1580387376,"sys":{"type":1,"id":8903,"country":"UA","sunrise":1580362602,"sunset":1580395519},"timezone":7200,"id":703448,"name":"Kyiv","cod":200}
```


, яка візуалізується наступним чином:

Київ, поточна дата: Thu Jan 30 2020 р. Довгота: 30.52 Широта: 50.43

Температура: 3 градусів по Цельсію

Відчувається як: -4 градусів по Цельсію

Вітер: 7 м/сек

Таким чином, можна відіслати запит до API і подивитись відповідь у браузері. В html-документі з вбудованим кодом JavaScript виклик виглядає так:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Open Weather</title>
    <!-- Посилання на таблицю стилів для розділів документу -->
    <link href="https://fonts.googleapis.com/css?family=Faster+One" rel="styleshe
et">
    <link rel="stylesheet" href="ajax.css">
  </head>

  <body>
      <!--В цих розділах документу будуть дані про погоду-->
      <header>
      </header>
      <section>
      </section>

      <script type="text/javascript">
        var header = document.querySelector('header');
        var section = document.querySelector('section');
                          //Запит погоди для Києва (id=703448)
        var requestURL = 'http://api.openweathermap.org/data/2.5/weather?id=70344
8';
                          //До запиту додається власний API Key
        requestURL = requestURL+'&appid=4f670455dced9331d1cdf2d9e55ba4d1';
        var request = new XMLHttpRequest();
        request.open('GET', requestURL);
        request.responseType = 'json';
                          //Відправка запиту
        request.send();

        request.onload = function()
                          //Ця функція виконується після отримання відповіді
        {
          var cityweather_json = request.response;
          showWeather(cityweather_json);
        }
    function showWeather(jsonObj)
```

```
    //Виведення даних щодо погоди
    {
      var d = new Date();
      var sHead="Київ, поточна дата: "+ d.toDateString(d)+" р. ";
        //Доступ до даних отриманого JSON-
об'єкта спирається на його ієрархічну структуру
      sHead=sHead+"Довгота: "+jsonObj.coord.lon+" Широта: "+jsonObj.coord.lat;
      var myArticle = document.createElement('article');
      var myH1 = document.createElement('h1');
      myH1.textContent=sHead;
      myArticle.appendChild(myH1);
                          //Інформація про місто буде в заголовку секції
      section.appendChild(myArticle);
                          //Дані погоди будуть у звичайних рядках
      var myArticle = document.createElement('article');
      var myPara1 = document.createElement('p');
      var myPara2 = document.createElement('p');
      var myPara3 = document.createElement('p');
                          //Переводимо у градуси Цельсія
      var t1=Math.round(jsonObj.main.temp-273.16);
      var t2=Math.round(jsonObj.main.feels_like-273.16);
      myPara1.textContent = 'Температура: '+t1+" градусів по Цельсію";
      myPara2.textContent = 'Відчувається як: '+t2+" градусів по Цельсію";
      myPara3.textContent = 'Вітер: '+jsonObj.wind.speed+" м/сек";
                          //Збираємо що вийшло, і документ відображається
      myArticle.appendChild(myPara1);
      myArticle.appendChild(myPara2);
      myArticle.appendChild(myPara3);
      section.appendChild(myArticle);
    }
    //-->
    </script>
  </body>
</html>
```

Рисунок 1 – Виклик зовнішнього API сервісу погоди

Використання Postman для тестування API

Застосування Postman потребує скачування з Інтернет та інсталяції. Можна створити безкоштовний аккаунт. Під час інсталяції треба відмовитись від створення групи для аккаунту.

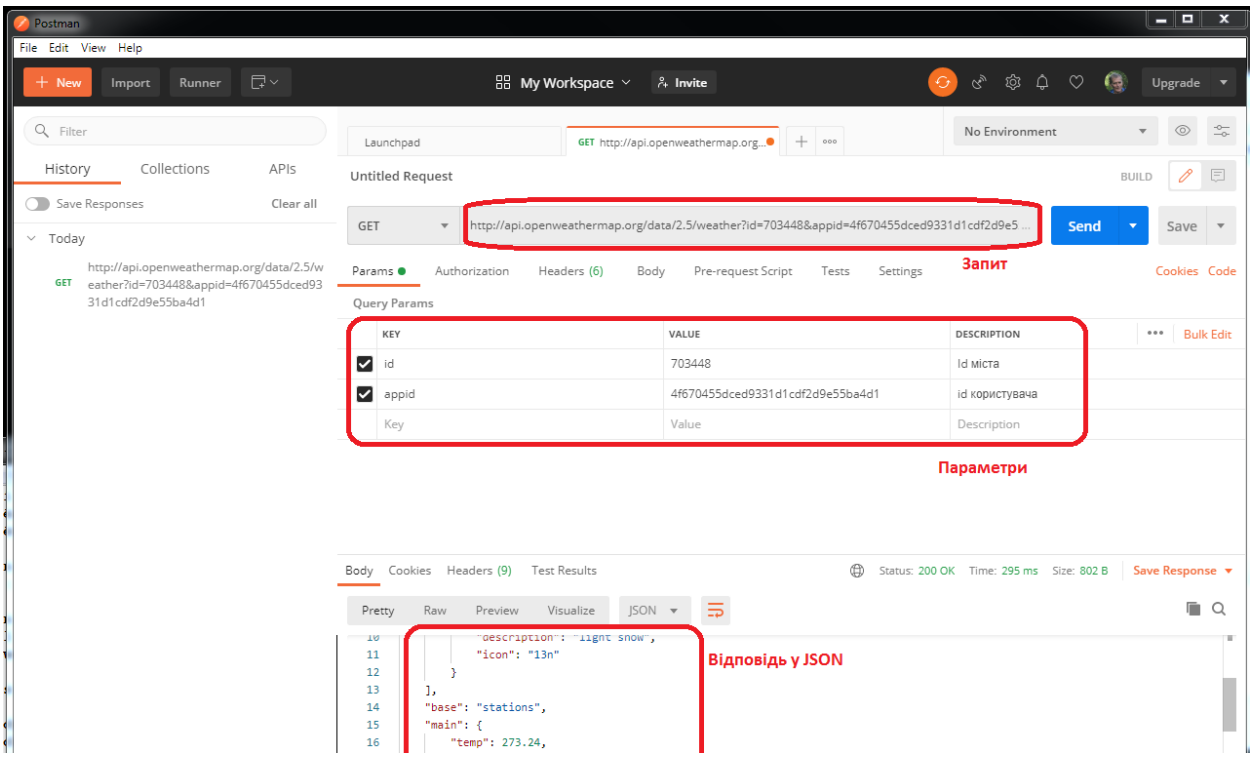


Рисунок 2 – Інтерфейс Postman

Найпростіший приклад побудови http-сервера засобами Node.js

Сервер створюється на локальному комп’ютері і доступ до нього здійснюється через порт 3000. Програмний файл httpServer_v1.js у Windows запускається з консолі Windows.

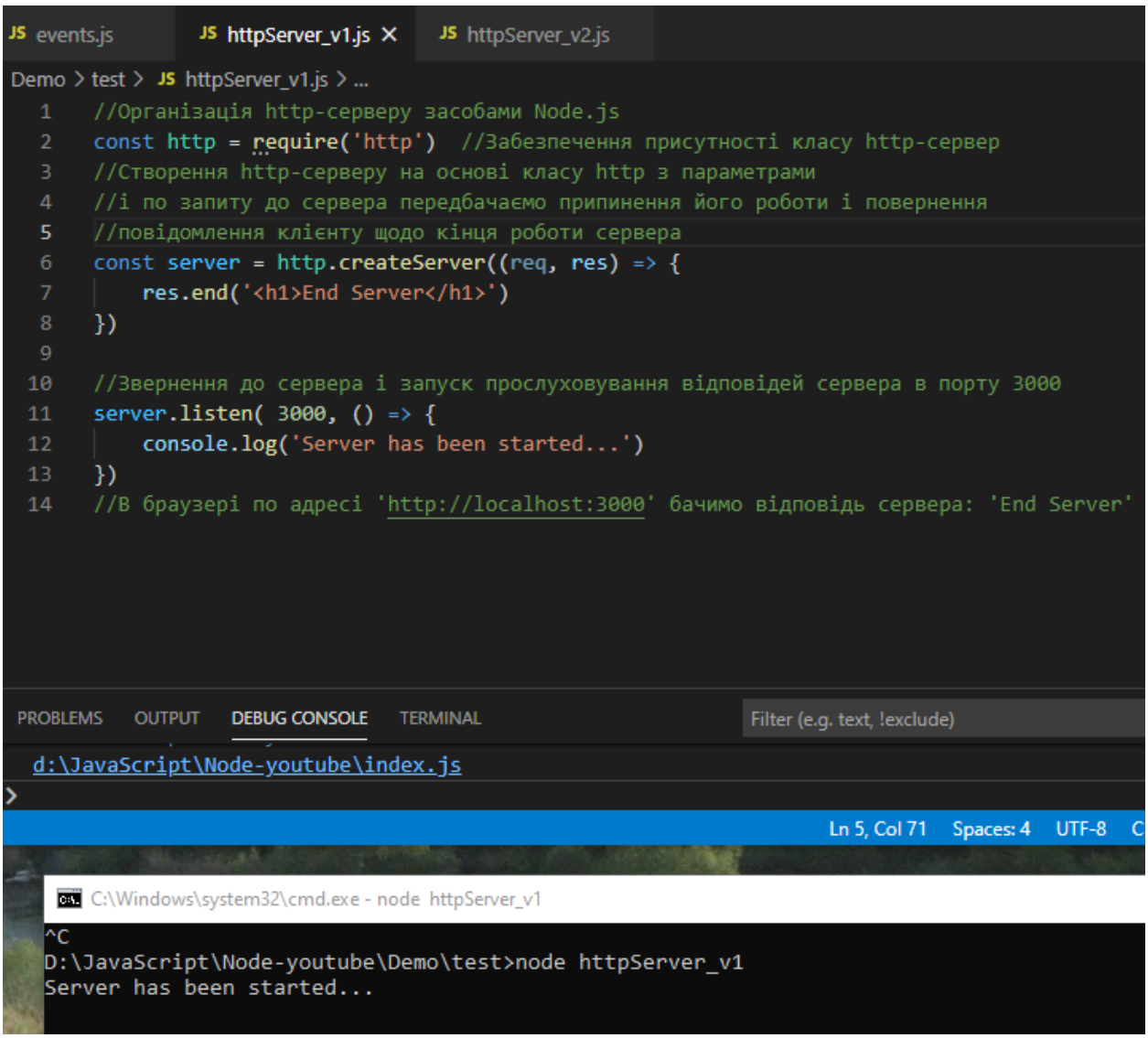


Рисунок 3 – Створення http-серверу і звернення до нього засобами Node.js

У браузері по адресі http://localhost:3000/ бачимо відформатовану згідно таблиці стилів відповідь сервера.

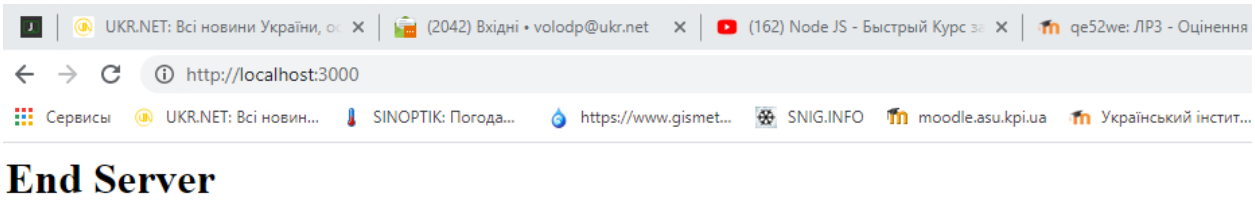


Рисунок 4 – Відповідь сервера в браузері

Визначення відповідей сервера за допомогою оператора if [2]

В даному прикладі формуємо заголовок розділу сторінки, який має сформуватись по результатах звернення до API.

За замовчуванням Node.js не має вбудованої системи маршрутизації. Зазвичай вона реалізується за допомогою спеціальних фреймворків типу Express. Однак якщо необхідно розмежувати найпростішу обробку пари-трійки маршрутів, то цілком можна використовувати для цього властивість url об'єкта Request. Наприклад:

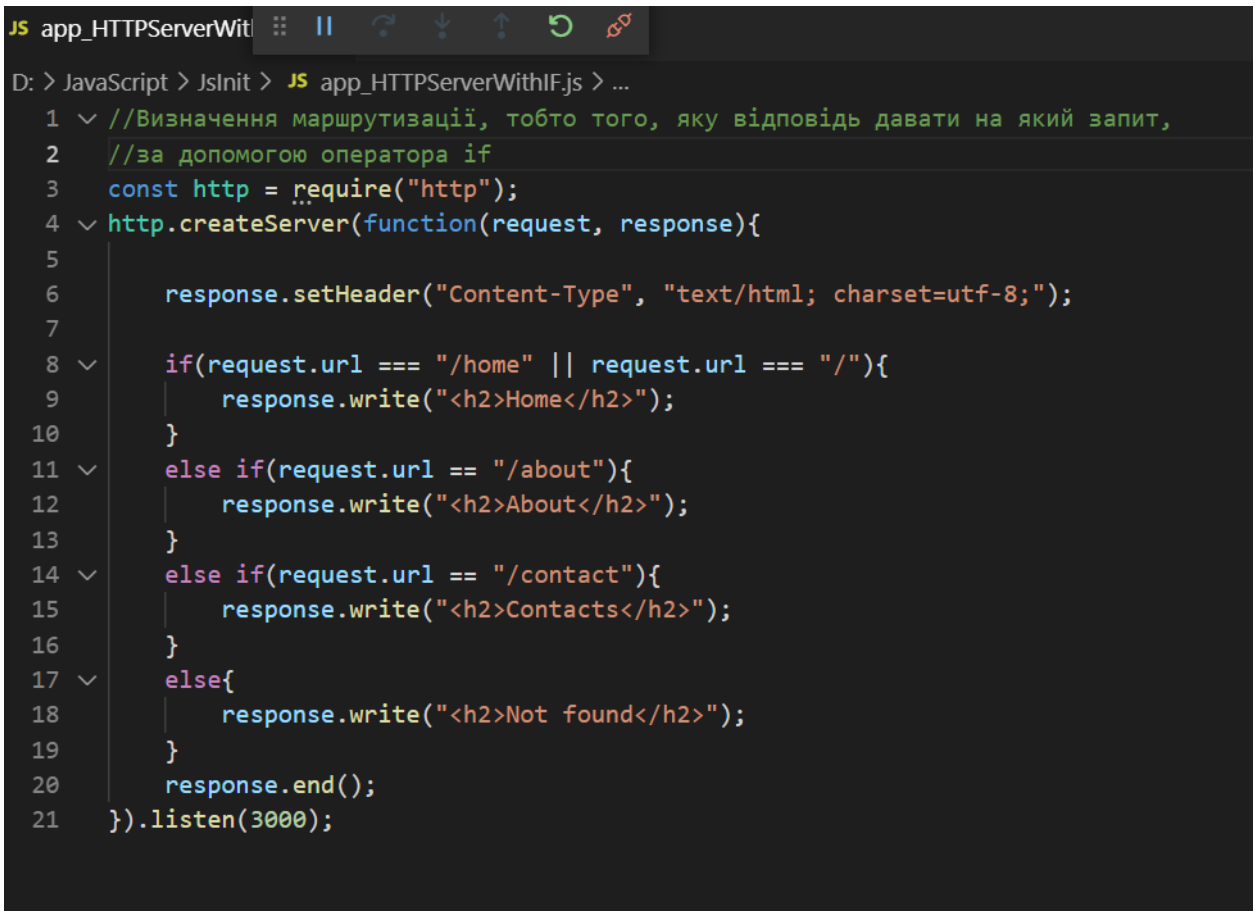


Рисунок 4 – Визначення маршрутизації запитів до HTTP-сервера за допомогою оператора if

В даному випадку обробляються три маршрути. Якщо йде звернення до кореня сайту або за адресою localhost: 3000/home, то користувачеві виводиться рядок "Home". Якщо звернення йде за адресою localhost: 3000/about, то користувачеві в браузері відображається рядок About і так далі. Якщо запитана адреса не відповідає жодному маршруту, то виводиться заголовок "Not Found".

Використані джерела

1. Node JS - Быстрый Курс за 1 час (Все Включено!) /https://www.youtube.com/watch?v=3aGSqasVPsI&ab_channel=%D0%92%D0%BB%D0%B0%D0%B4%D0%B8%D0%BB%D0%B5%D0%BD%D0%9C%D0%B8%D0%BD%D0%B8%D0%BD
2. Руководство по Node.js. Сайт о программировании / https://metanit.com/web/nodejs/ (Последнее обновление: 27.01.2021)

Додаток 1

Приклади публічних безкоштовних API

Приклад сервісу

Використання Google API. Підключення google sheets як місце для зберігання інформації (https://www.youtube.com/watch?v=8yJrQk9ShPg)

https://developers.google.com/api-client-library

https://rapidapi.com/blog/access-global-weather-data-with-these-weather-apis/

https://dzone.com/articles/4-free-weather-providers-api-to-develop-weather-ap-1

https://openweathermap.org/current

https://pogrommist.ru/2018/11/openweathermap-poluchaem-prognoz-pogody-po-api/

https://gorest.co.in/

https://www.rainviewer.com/api.html

Ці приклади надані Романом Гармашем, Інфопульс

Примітка.

Оскільки web-сервіс OpenWeatherMap використаний у прикладі, в разі використання цього сервісу не треба використовувати доступ по Id міста, треба задіяти інші способи доступу до даних:

- по назві міста;
- по координатах;
- погода для кількох міст;
- погода всередині кола.

Так само в разі використання GIS File API (Додаток 2) слід використати виклики API, відмінні від використаних у прикладі.

Додаток 2

Приклад API для візуалізації картографічної інформації – GIS FILE

http://gisfile.com/map/

Матеріал люб’язно надано Сергієм Янчуком, ІС-82.

Посилання на репозиторій:

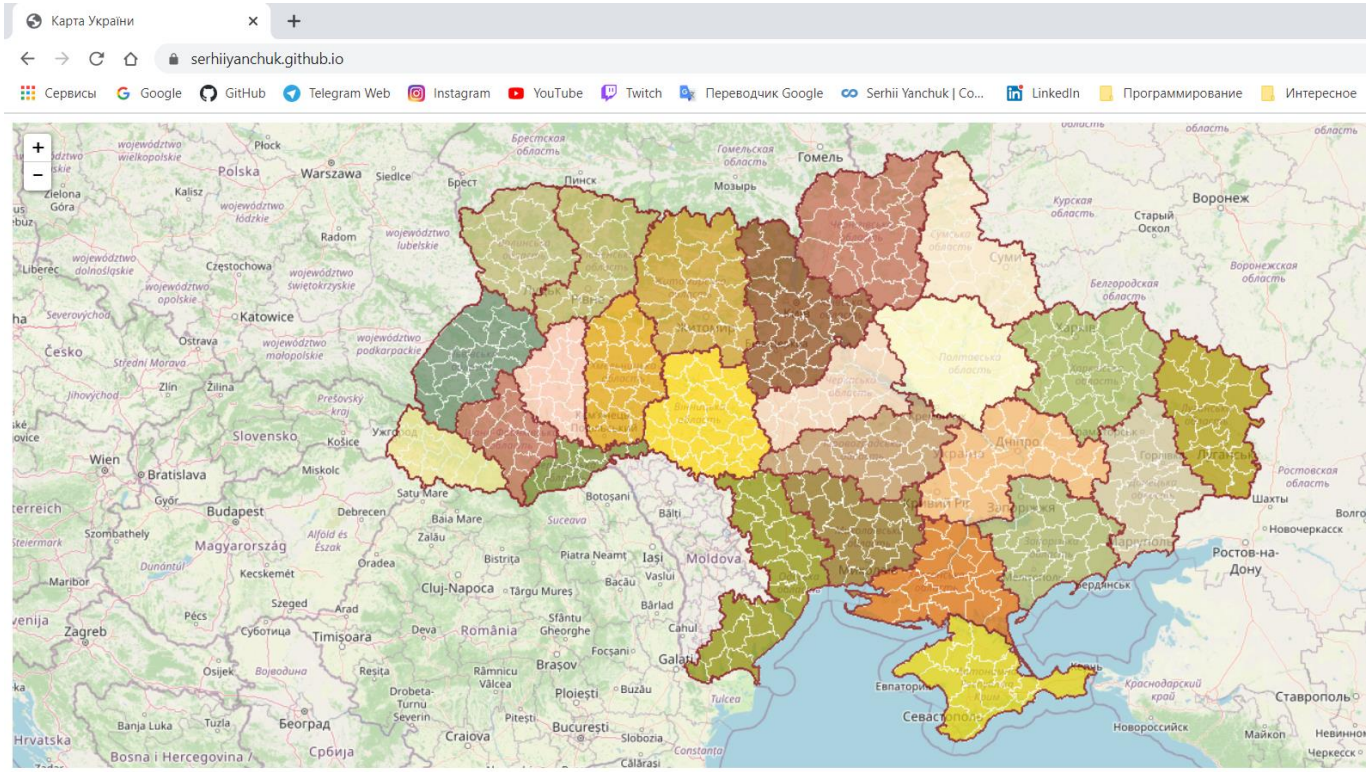
https://github.com/SerhiiYanchuk/SerhiiYanchuk.github.io

У даній лабораторній роботі використано GISFile API.

Посилання на сайт, де використовується API:

https://serhiiyanchuk.github.io/

Результат використання API: інтерактивна карта з особливо відміченими областями України.




```
<html>

<head>

    <title>Карта України</title>

    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
    <script src="https://gisfile.com/api/1.0/?map=ukraine&type=tile&width=100%&he
ight=360&id=map" type="text/javascript" charset="utf-8"></script>



</head>

<body>

    <div id="map" style="width: 100%; height: 600px"></div>

</body>

</html>
```

GISFile JavaScript API представляє собою набір JavaScript-компонентів, призначених для відображення інтерактивних карт на веб-сторінках і в додатках. API компоненти реалізовані у вигляді класів, функцій, статичних об'єктів і інтерфейсів. JavaScript API доступні для використання відразу після їх завантаження.

Підключення API проводиться за допомогою завантаження зовнішнього JavaScript-файлу, що містить компоненти API. Завантаження JavaScript-файлу і створення контейнера для відображення карти можуть бути виконані автоматично, якщо код JavaScript вставити в тіло документа. ID контейнера створюється автоматично, якщо він не вказаний в рядку src.

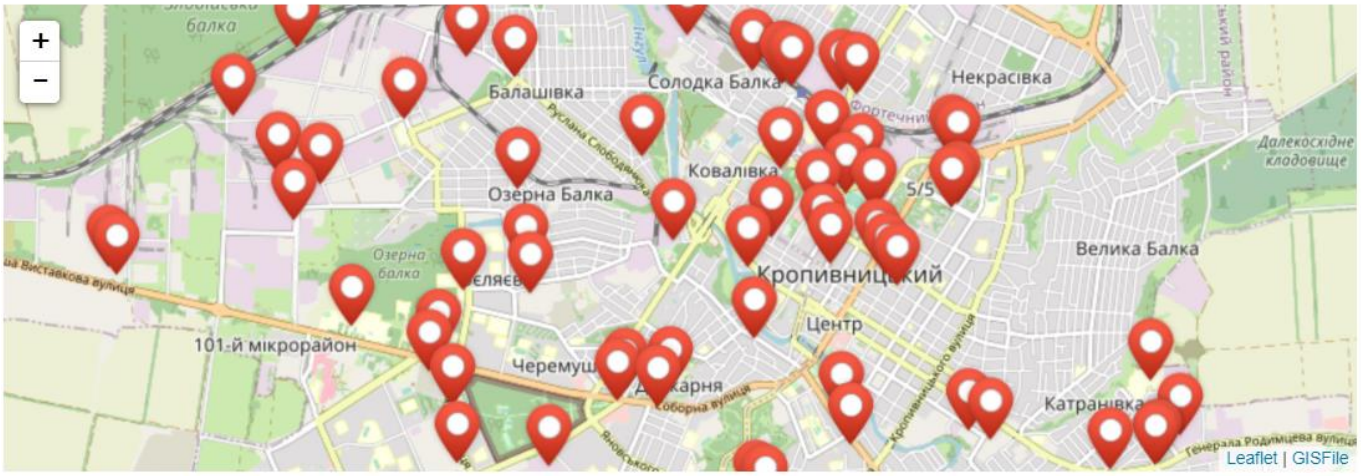
```
<script src="https://gisfile.com/api/1.0/?map=ukraine&type=tile&width=100%&height
=360&id=map" type="text/javascript" charset="utf-8"></script>
```

Основним компонентом API є карта, яка може бути розміщена в будь-якому HTML-елементі і має прямокутну форму. Параметрами карти є область показу, вид карти і відображення. При створенні карти можна вказати параметри відображення (центр карти і масштаб). Якщо вони не вказані, відображення виконується по об'єктах карти. Коли JavaScript розташований в тілі документа, необхідно вказати ширину width і висоту height, для автоматично створюваного контейнера карти.

API надає можливість відображення трьох видів карт: мітка (icon), шар (layer) і проект (map). Шар і проект мають підтип відображення (type): об'єкти (Overlay) і тайли (TileLayer і TileImage). Контейнер карти може містити один або кілька накладених один на одного шарів. Існує можливість комбінованого відображення власних і підключених шарів або проекту карти. Більш складні комбінації відображення реалізуються за допомогою GISFile API, JavaScript і бібліотеки LeafLet.

Відображення одного шару на мапі найчастіше використовується для відображення набору міток (точок). Формат іконки для міток можна задати за допомогою параметра icon = <Ім'я>, <Ім'я> - найменування іконки (приклади: icon-red, icon-red.png або http://gisfile.com/css/icons/icon-red. png)

```
<script src="http://gisfile.com/api/1.0/?layer=shelter&icon=icon-red&lat=48.52934&lon=32.24693&z=11&width=100%&height=500" type="text/javascript" charset="utf-8"></script>
```



Для використання алгоритму MarkerCluster угруповання міток, необхідно додати параметр marker.

```
<script src="http://gisfile.com/api/1.0/?layer=spcities&marker&width=100%&height=500" type="text/javascript" charset
="utf-8"></script>
```

Параметри завантаження API

Параметри завантаження дозволяють максимально спростити відображення карти на Веб-сторінках та в додатках. Використовуючи всього один рядок коду для завантаження JavaScript з параметрами можна вирішити безліч стандартних завдань. Рядок коду можна отримати в дизайнера карт всього в кілька кліків мишки. Розглянемо докладніше параметри завантаження JavaScript API.

Параметр	Описание	Формат	Пример
map	отображает указанный проект карты	map=<идентификатор проекта карты>	map=ukraine
layer	отображает указанный слой на карте	layer=<идентификатор слоя>	layer=mp
type	вид отображения проекты карты и слоя в виде гео-объектов (shape) и тайлов (tile)	type=<вид отображения> (используется с layer или map)	type=tile (по умолчанию type=shape)
marker	группировка отображения объектов	marker (без дополнительных значений, используется с layer)	marker
icon	отображает метку на карте и позволяет указать наименование иконки по заданным координатам lat и lon	icon или icon=<наименование иконки или полный путь к ней> (если параметры icon, layer или map не заданы, метка не отображается)	layer=spshels&icon=icon-red, icon, icon=icon-green
name	наименование объекта	name=<текст> (отображается при нажатии на метку)	name=Hello world
xname	наименование объекта в Hex формате	xname= (применяется вместо name, если кодировка не UTF-8)	xname=d09fd180d0b8d0b2d0b ...
note	описание объекта	note=<текст> (отображается при нажатии на метку)	note=Welcome! GISFile
xnote	описание объекта в Hex формате	xnote= (применяется вместо name, если используются специальные символы)	xnote=d09fd180d0b8d0b2d0b5 ...
lat	широта	lat=<число>	lat=49.03787
lon	долгота	lon=<число>	lon=2.28516
z	масштаб карты	z=<число от 0 до 19> (по умолчанию 6)	z=5
s	отображение панели поиска объектов	s=<значение 0 или 1> (по умолчанию 1)	s=0