ДИСТАНЦІЙНЕ ЗОНДУВАННЯ ЗЕМЛІ:

ОБРОБКА ТА АНАЛІЗ СУПУТНИКОВИХ ЗНІМКІВ НА ПЛАТФОРМІ GOOGLE EARTH ENGINE

С. М. БАБІЙЧУК
О. В. ГОРДІЄНКО
О. В. ТОМЧЕНКО
Л. І. ДАВИБІДА
Н. С. КОБЛЮК
С. Т. ПІКУЛЬ

Київ
2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
НАЦІОНАЛЬНИЙ ЦЕНТР «МАЛА АКАДЕМІЯ НАУК УКРАЇНИ»

**С. М. Бабійчук, О. В. Гордієнко, О. В. Томченко, Л. І. Давибіда,
Н. С. Коблюк, С. Т. Пікуль**

ДИСТАНЦІЙНЕ ЗОНДУВАННЯ ЗЕМЛІ: ОБРОБКА ТА АНАЛІЗ СУПУТНИКОВИХ ЗНІМКІВ НА ПЛАТФОРМІ GOOGLE EARTH ENGINE

Навчальний посібник

За редакцією
академіка НАН України
С. О. Довгого

Київ
Національний центр
«Мала академія наук України»
2023

УДК 528.8
Д48

Автори:

С. М. Бабійчук – завідувачка лабораторії «ГІС та ДЗЗ» НЦ «МАНУ», канд. пед. наук;
О. В. Гордієнко – молодший науковий співробітник Інституту телекомунікацій та глобального інформаційного простору НАН України, методист ІІ категорії лабораторії «ГІС та ДЗЗ» НЦ «МАНУ»;
О. В. Томченко – старша наукова співробітниця Державної установи «Науковий центр аерокосмічних досліджень Землі Інституту геологічних наук Національної академії наук України», методистка ІІ категорії лабораторії «ГІС та ДЗЗ» НЦ «МАНУ», канд. техн. наук;
Л. І. Давибіда – доцентка кафедри геотехногенної безпеки та геоінформатики Івано-Франківського національного технічного університету нафти і газу, методистка ІІ категорії лабораторії «ГІС та ДЗЗ» НЦ «МАНУ», канд. геол. наук;
Н. С. Коблюк – методистка ІІ категорії лабораторії «ГІС та ДЗЗ» НЦ «МАНУ»;
С. Т. Пікуль – методист ІІ категорії лабораторії «ГІС та ДЗЗ» НЦ «МАНУ»

Рецензенти:

Ю. І. Великодський – завідувач кафедри аерокосмічної геодезії та землеустрою Національного авіаційного університету, кандидат фізико-математичних наук, старший дослідник;
Р. Гілберт – доктор наук про Землю, ГІС-експерт Зволенського технічного університету (Словаччина)

Рекомендовано науково-методичною радою
Національного центру «Мала академія наук України»
(протокол № 4 від 27 вересня 2023 р.)

Д48 **Дистанційне** зондування Землі: обробка та аналіз супутникових знімків на платформі Google Earth Engine : навч. посіб. / С. М. Бабійчук, О. В. Гордієнко, О. В. Томченко та ін. ; за ред. С. О. Довгого. – Київ : Національний центр «Мала академія наук України», 2023. – 116 с.

ISBN 978-617-7945-58-0

Посібник є теоретичним компонентом третього рівня методики «Основи дистанційного зондування Землі», яку розробляє лабораторія «ГІС та ДЗЗ» НЦ «МАНУ». Він присвячений формуванню компетентностей з обробки та аналізу супутникових знімків у хмарній платформі компанії «Google» – Google Earth Engine.

Посібник знайомить читачів з основами мови програмування JavaScript і особливостями її використання на платформі Google Earth Engine. Також у ньому представлено інформацію про роботу з геоданими, зокрема: огляд колекцій даних у Google Earth Engine, їх фільтрування та візуалізацію, можливості використання функцій та створення вебзастосунків як форми представлення результатів дослідження.

Посібник може використовуватися науковцями, освітянами і керівниками секції «ГІС та ДЗЗ» системи МАНУ, а також усіма, хто прагне самостійно опанувати аналіз даних супутникового моніторингу Землі на платформі Google Earth Engine.

УДК 528.8

ISBN 978-617-7945-58-0

ЗМІСТ

Вступне слово 4

Перелік скорочень та умовних позначень 6

Розділ 1. Вступ до Google Earth Engine і мови програмування JavaScrip 7

1.1. Історія створення Google Earth Engine 8
1.2. Змінні в мові програмування JavaScript 11
1.3. Сторона сервера і сторона користувача 14
1.4. Типи даних, притаманні JavaScript 15

Розділ 2. Огляд колекції даних Google Earth Engine 20

2.1. Характеристики та типи колекцій даних 21
2.2. Растрові колекції даних 27
2.3. Векторні колекції даних 30

Розділ 3. Фільтрування колекцій 38

3.1. Фільтрування колекцій об’єктів 39
3.2. Фільтрування колекцій зображень 42
3.3. Сортування 46
3.4. Редуктори 47

Розділ 4. Функції в Google Earth Engine 53

4.1. Структура функції 54
4.2. Бібліотеки вбудованих функцій GEE 57
4.3. Створення функцій 62
4.4. Застосування функцій до колекцій даних 66

Розділ 5. Користувацький інтерфейс Google Earth Engine 71

5.1. Загальна характеристика модуля користувацького інтерфейсу 72
5.2. Основні компоненти модуля користувацького інтерфейсу 73
5.3. Огляд застосунків Earth Engine Apps 90
5.4. Публікація інтерактивного застосунку 93

Розділ 6. Оператори і цикли в JavaScript 97

6.1. Математичні оператори 98
6.2. Оператори порівняння 99
6.3. Логічні оператори 100
6.4. Умовні оператори if else і цикли for, while 100
6.5. Розширене використання операторів 105

Висновки 109

Список використаних джерел 110

Додатки 112

ВСТУПНЕ СЛОВО

Ласкаво просимо до світу Google Earth Engine і роботи з даними дистанційного зондування Землі у хмарних сервісах! Одним із напрямів, який на сьогодні найактивніше розвивається у світі інформаційних технологій, є збереження, обробка та аналіз великих обсягів даних (з англ. big data). Майже повсюдний доступ до швидкісного інтернету та зростаюча потреба в обчислювальних ресурсах підтримують цей розвиток, і хмарні сервіси стають важливим інструментом для розв'язання складних, комплексних і глобальних проблем. Наш посібник присвячений ознайомленню наукової та освітньої спільноти України з потужним хмарним сервісом обробки даних дистанційного зондування Землі – Google Earth Engine.

Ваше розуміння мови програмування, якою «спілкується» Google Earth Engine, і методів фільтрації й аналізу даних у Google Earth Engine буде ключовим для опанування цього посібника. Ми будемо використовувати мову програмування JavaScript, щоб детальніше розглянути функціональні можливості сервісу. Важливо також пам'ятати, що в Google Earth Engine є можливість використовувати інші мови програмування, як-от Python та R, взаємодіючи зі сторонніми застосунками.

Кількість даних, до яких тепер надає доступ цей хмарний сервіс, становить понад 50 петабайт (50 000 терабайт) інформації, зібраної за останні майже 40 років. Супутникові знімки місій NASA, Європейського космічного агентства, десятки комерційних місій – доступ до усіх цих даних можна отримати через Google Earth Engine. Цей хмарний сервіс працює на основі Google Cloud (хмара компанії «Google»), надаючи користувачеві доступ до високошвидкісних та високопродуктивних обчислень навіть з мобільного телефону. Оскільки це хмарна система, користувачеві не потрібна висока швидкість інтернету, бо всі обчислення відбуваються в хмарі, потужностями компанії «Google». Сервіс чудово інтегрований для збереження готових даних в обліковий запис Google Drive, Google Cloud Storage тощо.

Ми рекомендуємо опанування цієї книги після того, як ви ознайомитеся з попередніми частинами курсу «Основи дистанційного зондування Землі»: «Основи дистанційного зондування Землі: історія та практичне застосування» й «Основи дистанційного зондування Землі: аналіз космічних знімків у геоінформаційних системах». Це допоможе вам краще зрозуміти та легше освоїти матеріал посібника.

Сподіваємося, ви дізнаєтеся багато цікавого про можливості Google Earth Engine, і бажаємо вам надихнутися світом дистанційного зондування Землі, який відкриває цей хмарний сервіс.

Ми розробили для вас кольорові підказки: візуально ви зможете розпізнати рубрики, як-от: Визначення, Блок «Цікаво», Важливо!

Визначення	....
Блок «Цікаво»	....
Важливо!	*Важливий текст виділено червоним*

Також ми пояснюємо використання мови JavaScript у вигляді таблиці, яка складається з двох частин: власне скрипту (частина програмного коду) і текстового пояснення до нього. Кольором ми виділяємо різні типи даних (змінні, функції, дані тощо), аналогічно як їх підсвічує інтерфейс Google Earth Engine.

Зразок опису скрипту у вигляді таблиці

частина скрипту з Google Earth Engine	Роз'яснення, як працює скрипт
var date = ee.Date.fromYMD(2021,2,19);	ee.Date.fromYMD стандартна конструкція в GEE, де в дужках задається (рік, місяць, день);

Автори висловлюють подяку Федору Гонці за надані для оформлення видання ілюстрації.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface (прикладний програмний інтерфейс)

CSV – Comma-separated Values (значення, розділені комою)

FAO – The Food and Agriculture Organization of the United Nations (Продовольча та сільськогосподарська організація ООН)

GAUL – Global Administrative Unit Layers (глобальні шари адміністративних одиниць)

GEE – Google Earth Engine

JSON – JavaScript Object Notation (нотація об'єктів JavaScript)

LSIB – Large Scale International Boundary (міжнародний кордон великого масштабу)

NASA – National Aeronautics and Space Administration (Національне управління з аеронавтики і дослідження космічного простору)

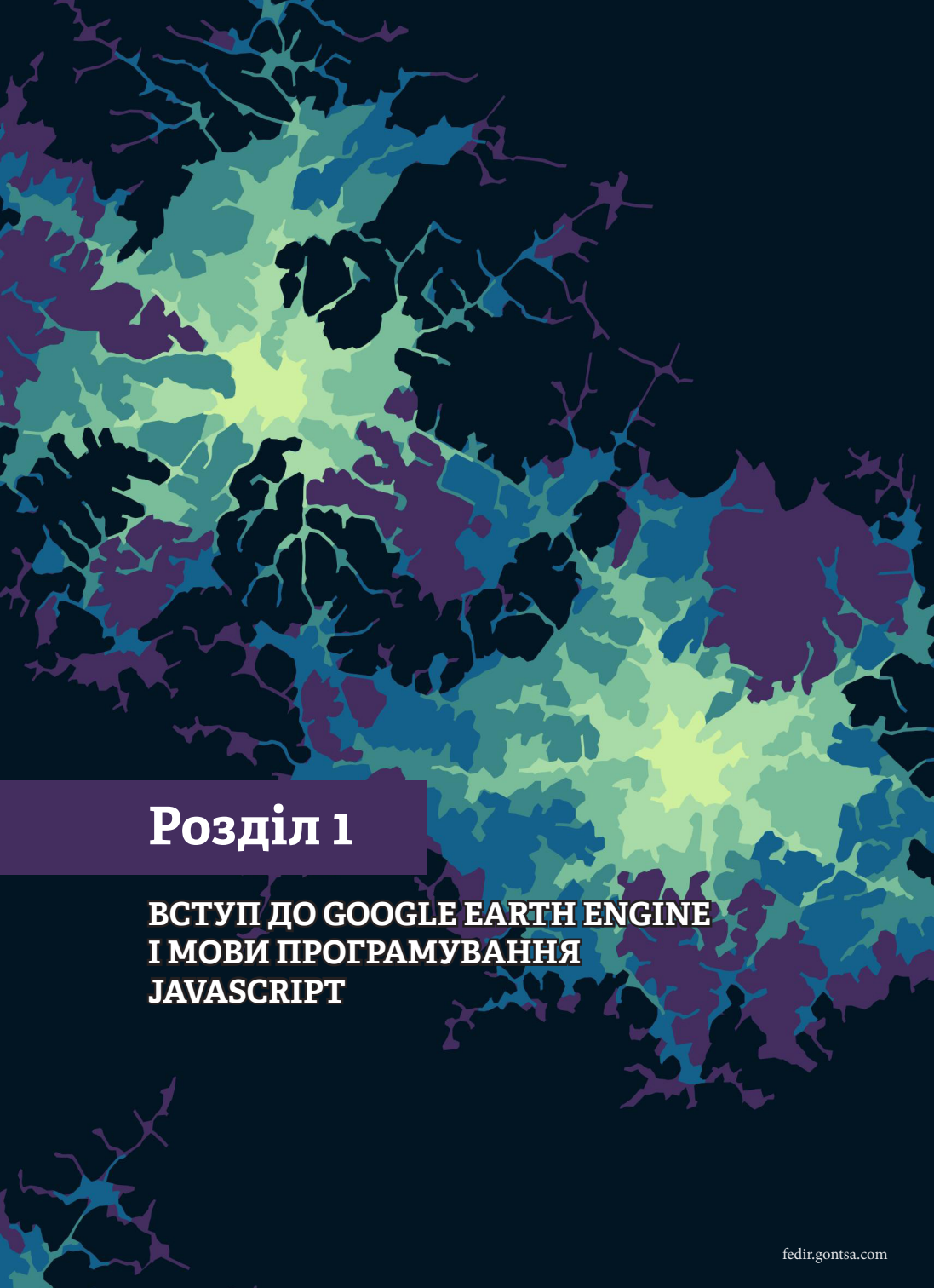
NDVI – Normalized Difference Vegetation Index (нормалізований диференційний вегетаційний індекс)

PNG – Portable Network Graphics (растровий формат збереження графічної інформації)

UI – User Interface (інтерфейс користувача)

URL – Uniform Resource Locator (уніфікований локатор ресурсів)

OSM – OpenStreetMap (відкрита вулична мапа)



Розділ 1

ВСТУП ДО GOOGLE EARTH ENGINE І МОВИ ПРОГРАМУВАННЯ JAVASCRIPT

1.1. Історія створення Google Earth Engine

Google Earth Engine (далі – GEE) – хмарна платформа для обробки даних супутникового моніторингу Землі та інших даних спостереження за Землею. Ця платформа надає доступ до великого сховища супутникових знімків і потужностей обчислювальних ресурсів Google [1].

У 2009 р. компанія «Google» отримала $10 млн фінансування від NASA для розробки GEE. Воно було надане через програму NASA «Applied Sciences Program» [2], яка спрямована на розробку інструментів і технологій для використання даних, отриманих із космосу, для моніторингу глобальних проблем, як-от: зміна клімату, катастрофи природного походження, забруднення довкілля та ін.

Фінансування від NASA дало змогу Google залучити висококваліфікованих фахівців та розробників, які спільно з експертами NASA створили платформу GEE. Ця платформа поєднала великі обсяги даних про земну поверхню, отриманих із супутників та інших джерел, з програмними інструментами для їх обробки та аналізу.

Крім того, співпраця з NASA дала змогу Google отримати доступ до даних, які раніше були недоступні для широкого загалу, зокрема про зміну клімату та океанографічних даних.

Цікаво

«До 1999 р., коли ми запустили Landsat 7, Landsat 4 і 5 експлуатувалися приватною фірмою, і вартість однієї сцени сягала 4400 доларів, що мало хто міг собі дозволити», – говорить James Irons (Джеймс Айронс), науковець проєкту Landsat Data Continuity Mission у Центрі космічних польотів ім. Годдарда, де програма Landsat функціонувала з моменту її заснування. «У 2008 р. вони прийняли рішення, яке я називаю "інституційно сміливим" – поширювати ці дані безкоштовно для тих, хто робить запит щодо них», – розповідає він.

Фахівці з обробки даних у Маунтін-В'ю, штат Каліфорнія, де розташований головний офіс інтернет-гіганта Google, не гаючи часу, скористалися перевагами нового ресурсу. Наприкінці 2010 р. Google представила Google Earth Engine – хмарну обчислювальну платформу для доступу та обробки зображень планети, отриманих із супутника Landsat за майже 40 років. З оцифруванням сховища інформації стало можливим наукове вивчення світових тенденцій з використанням даних Landsat [3].

Це дало можливість GEE зосередитися на дослідженні складних глобальних проблем, які вимагають знань із різних галузей науки та технологій [4].

Прототип GEE було побудовано на системі **CLASlite**, який розроблений в Університеті Карнегі-Меллон, і системі **Sistema de Alerta de Desmatamento** (SAD) компанії «Imazon». Прототип представили 2009 р. на конференції ООН зі зміни клімату COP15. Наступного року на цій самій конференції COP16 відбулося офіційне представлення платформи GEE. Науковці презентували карти водних ресурсів басейну річки Конго та лісів Мексики. На сьогодні в GEE доступно понад 80 наборів даних NASA. [5] (див. *рис. 1*).

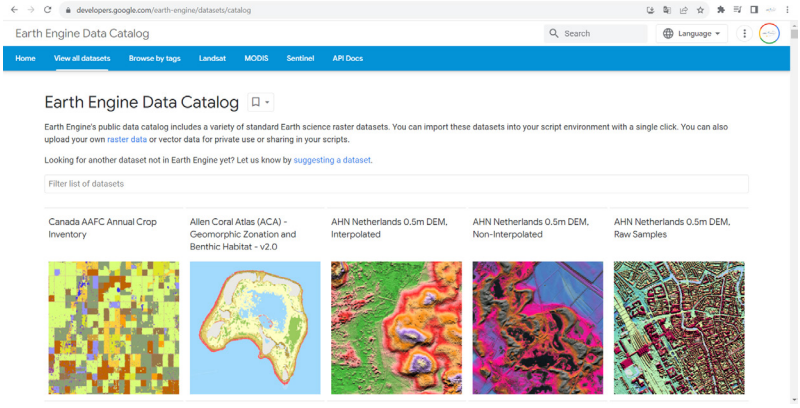


Рис. 1. Інтерфейс сайту Earth Engine з доступом до каталогу даних Earth Engine (https://developers.google.com/earth-engine/datasets/catalog)

Цікаво

Навесні 2013 р. у співпраці з журналом «Time» Google Earth Engine випустив веб-функцію «Timelapse», яка використовує знімки Landsat, щоб дати змогу користувачам переглядати анімацію будь-якої ділянки землі на планеті Земля – за винятком тих, що перебувають поблизу полюсів – у режимі реального часу з 1984 р. по 2012 р. Функція «Таймлапс» від Google виявилася популярною: лише за перший тиждень анімацію переглянули понад 3 млн глядачів. «Ви можете бачити дивовижні явища так чітко, ви можете бачити, як Лас-Вегас стрімко зростає, в той час як сусіднє озеро Мід зменшується. Ви можете побачити вирубку лісів в Амазонії, штучні острови, що проростають біля узбережжя Дубаю, льодовик Колумбія, що відступає на Алясці», – говорить Ребекка Мур, інженерна менеджерка програми «Google Earth Outreach» [6].

Партнерство з Google є частиною глобальної партнерської програми відділу наук про Землю NASA. Більше інформації про неї можна знайти на вебсайті Програми прикладних наук відділу наук про Землю [8]. Розглянемо інтерфейс хмарного сервісу GEE (див. *рис. 2*).

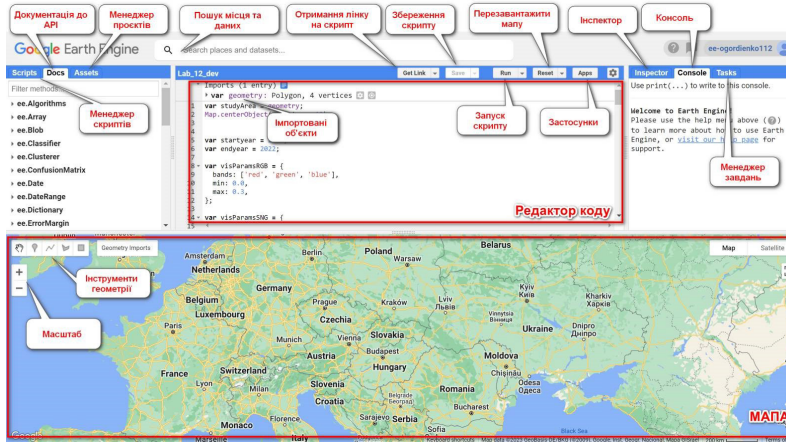


Рис. 2. Інтерфейс програми GEE

Map *(Мапа)* – займає найбільшу площу на сторінці, і саме на ній буде візуалізуватися результат.

Code Editor *(Редактор коду)* – головне вікно сервісу, в якому ви пишете свій скрипт.

Console *(Консоль)* – служить для виведення текстової інформації, результатів розрахунку, графіків тощо.

Tasks *(Менеджер завдань)* – у ньому можна бачити результати завантаження та вивантаження з GEE.

Inspector *(Інспектор)* – служить для отримання інформації за кліком, координати, значення в точці тощо.

Help *(Допомога)* – служить для зв'язку з компанією «Google» стосовно помилок у роботі програми.

Scripts manager *(Менеджер скриптів)* – у ньому зберігаються скрипти, які ви створювали, приклади скриптів від Google, а також доступ до папок інших користувачів.

Docs or API Documentation *(Документація до API)* – повна документація з описом стосовно всіх команд у GEE.

Reset *(Перезавантажити)* – перезавантажує вид на мапі і результати в консолі, повністю очищає їх від результатів попередніх розрахунків, не видаляючи скрипт.

Apps *(Застосунки)* – дає змогу створити вебзастосунок на основі вашого скрипту, що буде працювати як сторінка в інтернеті.

Assets manager *(Менеджер об'єктів)* – служить для завантаження своїх даних до середовища GEE.

Geometry tools *(Інструменти геометрії)* – служать для навігації по мапі, а також для створення своїх векторних даних.

Scale tools or Zoom *(Масштаб або зум)* – служить для наближення до території інтересу на мапі.

Search for data *(Пошук місця та даних)* – служить для пошуку даних у середовищі GEE та місця на мапі за назвою чи координатами.

Imports *(Імпорт)* – тут зберігаються об'єкти, які імпортувалися з вашого скрипту.

Get link (URL) to the script *(Отримання лінку на скрипт)* – дає змогу ділитися своїм скриптом із користувачами GEE.

Save the script *(Збереження скрипту)* – дає змогу зберігати скрипт.

Run the script *(Запуск скрипту)* – кнопка, яка виконує скрипт, написаний у редакторі коду.

1.2. Змінні в мові програмування JavaScript

Змінні є основними елементами програмування і використовуються для зберігання, обробки та маніпулювання даними в програмах. Уявіть, що змінні – це ящики-органайзери, які ви використовуєте для організації різних речей у своїй кімнаті. Кожен ящик може мати свою назву і зберігати різні предмети. Наприклад, у вас є кілька таких ящиків, один з них може називатися «Книги», і в ньому ви зберігаєте всі свої книги. Інший ящик може мати назву «Гаджети», і там зберігаються планшети, мобільні телефони, зарядні пристрої тощо. Вам може знадобитися якась інформація із цих ящиків, наприклад: вам потрібно прочитати книгу «Наша століття. Короткі нариси про довгу війну» Володимира В'ятровича. Ви будете її шукати в ящику «Книги».

Змінні в програмуванні подібні до цих ящиків, а дані – це предмети (книги, гаджети тощо) в цих ящиках. Ящики-змінні дають вам змогу зберігати й організовувати дані в програмі. Ви можете присвоїти змінній ім'я як назву ящика, і зберігати в ній дані так само, як у ящиках. Ви можете змінювати значення змінних так само, як перекладаєте речі між різними ящиками. Наприклад, змінна «кількістьКниг» може містити кількість, які у вас є. Ви можете збільшувати це число, якщо придбаєте нові книги, або зменшувати, якщо

подаруєте їх другові. Змінні допомагають програмістам працювати з даними, конкретизувати, які дані потрібні саме зараз, виконувати обчислення тощо.

Отже, змінна – це символічне ім'я, яке використовується для звернення до певного типу даних, до яких має доступ користувач.

Цікаво

Під час роботи над місією «Apollo 11», яка вивела людину на Місяць, перед ученими й інженерами NASA постало складне завдання обчислення оптимальної траєкторії для польоту. Вони мали врахувати взаємодію гравітації Землі і Місяця, обертання планет, швидкості ракети та багато інших факторів. Усі ці обчислення були дуже складними і вимагали точності до найменших деталей. Використання змінних допомогло зробити ці обчислення більш зрозумілими та керованими. Вчені створили математичні моделі, де вони використовували змінні для представлення різних параметрів та даних. Під час місії «Apollo 11» американська програмістка та інженерка Маргарет Гамільтон зробила вагомий внесок у розробку програмного забезпечення для модуля, який керував процесом приземлення на Місяць і повернення на Землю. Її методи та підходи до розробки програм були інноваційними для свого часу й використовувалися для забезпечення надійності та безпеки місії. Після успішного завершення місії «Apollo 11» Маргарет Гамільтон продовжила працювати над іншими космічними проєктами, включаючи програми Шаттлу та Міжнародної космічної станції. Маргарет Гамільтон стала символом внеску жінок у науку, технологію та інженерію. Вона також активно пропагує важливість програмування і STEM-освіти (наука, технологія, інженерія та математика) для молоді.

У цьому посібнику ми будемо спиратися на мову програмування **JavaScript**. Вона була розроблена Бренданом Айком у 1995 р. під час його роботи в компанії «Netscape». Спочатку мова задумувалася для створення інтерактивних елементів на вебсторінках та взаємодії з користувачем у браузері [7]. Нині ж вона використовується значно ширше, зокрема в розробленні застосунків для голосового управління, інтернету речей, блокчейну та багато ін.

Важливо відзначити, що **Java** і **JavaScript** – це дві різні мови програмування з різними призначеннями та характером використання.

Призначення та сфери застосування

Java: це мова програмування загального призначення, яка використовується для розробки різноманітних застосунків, включаючи великі програми, мобільні застосунки, серверні застосунки тощо. **Java** є статично типізованою мовою, а це означає, що типи змінних визначаються на етапі компіляції.

JavaScript: це скриптова мова програмування, яка переважно використовується для розробки вебзастосунків. Вона вбудовується безпосередньо у

вебсторінки й виконується в браузері. **JavaScript** є динамічно типізованою мовою, тобто типи змінних визначаються під час виконання програми.

Синтаксис та структура

Java: синтаксис **Java** більш формальний і строгий. Він вимагає від програміста вказувати типи змінних, оголошувати класи та методи, ініціалізувати змінні тощо.

JavaScript: синтаксис **JavaScript** менш строгий. Він дає змогу оголошувати змінні без вказівки типів і має більш вільний підхід до організації коду.

Парадигми програмування

Java: **Java** ґрунтується на об'єктно-орієнтованій парадигмі, де програма організована навколо об'єктів та класів.

JavaScript: **JavaScript** підтримує об'єктно-орієнтовану парадигму, але також має функціональні і прототипні можливості (що дає змогу користувачеві працювати з функціями як з даними, а прототипність – створювати спадкування й об'єктно-орієнтовану архітектуру коду).

Середовище виконання

Java: програми на **Java** виконуються на віртуальній машині **Java** (JVM), що дає змогу користувачеві працювати на різних операційних системах без зміни коду.

JavaScript: код **JavaScript** виконується безпосередньо в браузері користувача, що забезпечує динамічність вебсторінок.

Типи застосунків

Java: використовується для розробки різноманітних застосунків, включаючи стійкі та великі системи.

JavaScript: головно використовується для розробки вебзастосунків, як-от інтерактивні інтерфейси та взаємодія з користувачем.

Отже, **Java** і **JavaScript,** маючи схожі назви, використовуються для різних завдань і мають відмінності в синтаксисі та призначенні.

Змінні в **JavaScript** були задумані як засіб для зберігання та маніпулювання даними. Перші версії **JavaScript** не мали чіткого визначення типів даних, тому змінні могли зберігати дані будь-якого типу. Вони давали змогу зберігати числа, рядки, об'єкти тощо. Змінні в **JavaScript** оголошувалися за допомогою ключових слів **var**, **let** або **const**.

var є оригінальним ключовим словом для оголошення змінних, але з часом з'явилися більш сучасні ключові слова: **let** (для оголошення змінних із обмеженою областю видимості) та **const** (для оголошення константних змінних).

*Різниця між **let** та **var** у тому, як вони обробляють область видимості. var має глобальну або функціональну область видимості, тоді як let має блочну область видимості, що означає, що вона доступна лише в рамках блоку коду, в якому вона була оголошена.*

Щодо const, вона також використовується для зберігання даних, але з ключовою відмінністю – одного разу призначена константа не може бути переприсвоєна після того, як їй було присвоєно значення.

1.3. Сторона сервера і сторона користувача

Коли ми говоримо про програми або вебсайти, які працюють в інтернеті, можна виділити дві основні сторони: сторону сервера і сторону користувача. Ці дві сторони відіграють важливу роль у взаємодії та функціонуванні багатьох програм і вебсайтів.

Сторона сервера

Сторона сервера – це та частина програми, яка працює на сервері, центральному комп'ютері, який надає ресурси та інформацію користувачам. Сервер може зберігати бази даних, обробляти дані, відправляти запити до інших серверів та надавати інформацію користувачам через інтернет.

Якщо ви відкриваєте вебсайт із новинами, сервер обробляє запит і відправляє вам результат, який ви шукали, тобто сторінку з новинами.

Коли ви надсилаєте повідомлення в соціальній мережі, сервер зберігає це повідомлення в базі даних і відображає його іншим користувачам, яким ви це повідомлення адресували.

Сторона користувача

Сторона користувача – це та частина програми, яка виконується у веббраузері або на пристрої користувача. Вона відповідає за те, як ви бачите і взаємодієте з програмою.

Коли ви відкриваєте вебсайт новин і прокручуєте сторінку, це відбувається на стороні користувача.

Коли ви натискаєте на кнопку «Відправити» у формі для введення повідомлення, ця дія відбувається на вашому комп'ютері або пристрої.

У GEE деяка інформація, як-от відображення мап, графіків обробляється у вашому браузері. Проте розрахунки для відображення даних у вигляді мап і графіків відбуваються саме на стороні GEE. Це значно пришвидшує обробку даних для отримання результату.

```
var clientString = 'Я рядок';
print(typeof clientString);
```
// Приклад рядка на стороні клієнта

```
var serverString = ee.String('Я не рядок я об'єкт!');
print(typeof serverString);
```
// Приклад рядка (об'єкт) на стороні сервера

Взаємодія між сторонами

Сторона сервера і сторона користувача взаємодіють між собою, обмінюючись даними. Користувач може відправити запит до сервера, наприклад, запит на відображення певної сторінки або отримання даних. Сервер обробляє цей запит і надсилає назад відповідь, яку сторона користувача може побачити на своєму екрані.

Приклади:

Коли ви вводите адресу вебсайту у веббраузері, ваш браузер надсилає запит до сервера, який відповідає сторінкою вебсайту.

Коли ви робите покупку онлайн і вводите інформацію про платіж, дані відправляються на сервер для обробки.

Загальний підхід

Розробляючи програми або вебсайти, спеціалісти часто використовують JavaScript для програмування сторони користувача, а мови програмування Node.js, PHP, Ruby тощо – для програмування сторони сервера. Це дає змогу створити взаємодію між цими двома сторонами й забезпечити користувачам більші можливості та функціональність.

Отже, сторона сервера відповідає за обробку даних, зберігання інформації та надання відповідей користувачам, а сторона користувача – за відображення і взаємодію користувача з програмою чи вебсайтом.

1.4. Типи даних, притаманні GEE

GEE використовує спеціальний синтаксис програмування – **API** (Application Programming Interface) Earth Engine, який є розширенням стандартного JavaScript. Цей синтаксис дає змогу взаємодіяти з геоданими та виконувати операції з великими об'ємами даних на облікових серверах Google Earth Engine і є доступним на Python і JavaScript, що дає змогу легко використовувати потужність хмари Google для власного дослідження [8].

Загалом GEE розроблений так, щоб не залежати від мови програмування. Основна відмінність полягає в синтаксисі, який використовується для виклику функцій API. Як тільки ви зрозумієте синтаксис мов програмування, ваш код можна буде легко адаптувати, оскільки усі вони використовують однакові функції API [9].

Розгляньмо детальніше типи даних, які можна обробляти в GEE через мову програмування JavaScript. Це важливий аспект програмування, оскільки правильне розуміння типів даних допомагає ефективно працювати зі структурованою інформацією і дає змогу комп'ютеру розуміти, що означає значення чи запит користувача. Наприклад, якщо ви хочете додати два числа, то потрібно, щоб ці числа були збережені і відображалися у форматі чисел, бо ж якщо ви спробуєте додати число до рядка, то комп'ютер не «зрозуміє», що ви хочете зробити, і видасть помилку.

Тобто типи даних – це той формат інформації, яку «розуміє» ПК через мову програмування. Почнемо з типів даних у JavaScript. У цій мові програмування існує декілька типів даних:

Числа (number) – можуть бути натуральними числами (42) або з плаваючою крапкою (наприклад, 3.14).

Рядки (string) – це послідовності символів (наприклад, «Привіт, МАНУ!»).

Логічні значення (boolean) – використовуються для вираження логічних операцій, мають два можливих значення: true (істина) і false (неправда).

Об'єкти (object) – це структури даних, які можуть містити в собі інші значення, включаючи числа, рядки, логічні значення, об'єкти та функції (наприклад, { name: "JASU", country: Ukraine, scientific clubs: 72 }).

Функції (function) – це блоки коду, які можна викликати повторно.

Велике ціле число (BigInt) – використовується для зберігання дуже великих чисел, що перевищують межі стандартних 64-бітних чисел.

Порожній (null) – значення, яке використовується для позначення відсутності даних.

Невизначений (undefined) – значення, що вказує на змінну, яка не має значення; його можна використовувати для перевірки того, чи було присвоєно значення змінній.

У GEE, як і в багатьох програмувальних середовищах, існує низка базових типів даних, спільних із JavaScript, зокрема: **number, string, boolean, object, function, null** та **undefined** (тобто усі, окрім великого цілого числа (**BigInt**)).

Розглянемо детальніше спільні із JavaScript типи даних на конкретних прикладах.

Числа (number) – це може бути висота поверхні, температура, відстань тощо.

```
var number = 42;
print(number)
var float = 3.14159;
console.log(temperature)
```

```
// Приклад цілого числа, виведення у консоль за допомогою print
// Приклад числа із плаваючою точкою, виведення у консоль за допомогою console.log
```

Рядки (string) – назва населеного пункту, опис об'єкта або будь-який текст.

```
var locationName = 'Mount Hoverla';
```

```
// Приклад рядка
```

Логічні значення (boolean) – вибір даних, які відповідають певним критеріям, як-от вибірка територій із певним кліматичним станом, рівнем опадів тощо.

```
var trueValue = true;
var falseValue = false;

print('trueValue:', trueValue);
print('falseValue:', falseValue);
```

```
У цьому прикладі в консоль виводяться два булеві оператори true (істина) і false (неправда).
```

Об'єкти (object) – можна створити об'єкт для збереження характеристик різних географічних об'єктів, які включають координати, назви, кількість населення та інші властивості.

```
var location = {
  name: 'Місто Київ',
  latitude: 50.4501,
  longitude: 30.5234
};

print(location);

print('Назва місця:', location.name);
print('Широта:', location.latitude);
print('Довгота:', location.longitude);
```

```
// Об'єкт location містить три значення.

// Виведення у консоль значення name
// Виведення у консоль значення latitude
// Виведення у консоль значення longitude
```

Функції (function) – це такий блок коду, який виконується, коли до нього звертаються.

```
function dodaj(a, b) {
  var suma = a + b;
  return suma;
}

var x = 5;
var y = 3;
var result = dodaj(x, y);
print('Результат додавання:', result);
```

```
// Визначаємо функцію, яка додає два числа
// Повертаємо результат

Викликаємо функцію та передаємо аргументи x і y
```

Порожній (null) – якщо дані про певний атрибут (кількість жителів у гірській місцевості) відсутні для деяких географічних об'єктів, їх значення може бути null.

```
var noValue = null;
var notDefined;
print(noValue);

print(notDefined);
```

```
// Змінна, що дорівнює null
// Змінна, що не задана
// Результат у консоль
```

Цікаво

BigInt. *BigInt не підтримується в середовищі Google Earth Engine, оскільки цей тип даних зазвичай використовується для дуже великих цілих чисел, що перевищують 2^53. У Google Earth Engine його замінив тип даних 64-розрядного числа (відомий також як Long), який пропонує більшу сумісність з іншими мовами програмування.*

Symbol. *Symbol не підтримується в середовищі Google Earth Engine. Її замінила функція FillSymbol, яка пропонує більше можливостей для форматування символів. Функція FillSymbol також використовується для створення символів, але вона має більше аргументів, ніж Symbol. Наприклад, FillSymbol дає вам змогу задавати колір заливки, прозорість і стиль границі.*

Крім того, GEE підтримує деякі спеціальні типи даних, яких нема у JavaScript, але які притаманні й ефективні для роботи в цьому хмарному сервісі, оскільки дають змогу працювати з геоданими (супутниковими знімками, векторними об'єктами, наборами геоданих тощо). Тут ми опишемо деякі типи таких даних:

Date – використовується для зберігання дат і часу.

DateRange – використовується для представлення діапазону дат, які можуть бути використані для вибору даних за певний період.

ee.Geometry.Point – використовується для зберігання точок на карті.

GeoPolygon – використовується для зберігання полігонів на карті.

GeoMultiPolygon – використовується для зберігання багатьох полігонів на карті.

Image – це тип даних, який представляє растрове зображення. Зображення може бути 2D або 3D.

ImageCollection – використовується для зберігання послідовності растрових зображень.

Feature – це тип даних, який представляє геопросторовий об'єкт. Об'єкт може бути точкою, лінією або полігоном.

FeatureCollection – використовується для зберігання геопросторових об'єктів (точок, ліній або полігонів).

Dictionary – тип даних для представлення словників (ключ-значення), що використовується для зберігання метаданих та додаткової інформації.

List – тип даних для представлення списків об'єктів (може містити об'єкти різних типів).

Array – використовується для представлення масивів даних, що можуть бути використані для обчислень та аналізу.

Розділ 2

ОГЛЯД КОЛЕКЦІЙ ДАНИХ GOOGLE EARTH ENGINE

2.1. Характеристики та типи колекцій даних

У GEE представлена велика кількість даних: усі вони зберігаються в колекціях даних із власною структурою та наповненням.

Колекції даних є ключовим елементом платформи GEE, вони дають змогу користувачам отримувати доступ до величезного обсягу геопросторової інформації і проводити аналіз та моделювання геопросторових процесів (див. *рис. 3*).

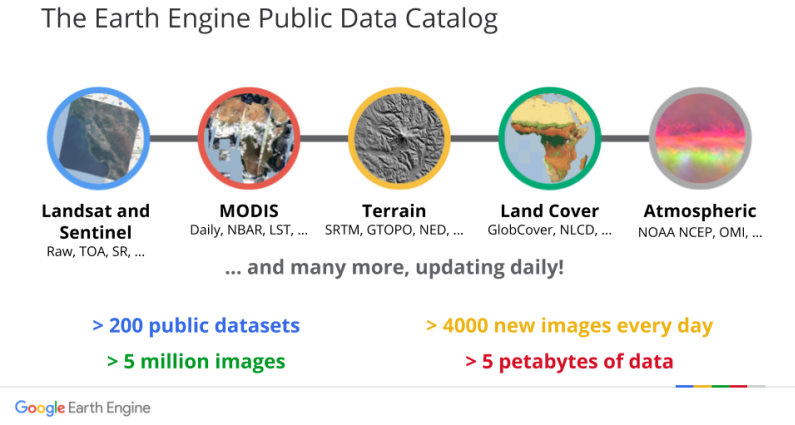


Рис. 3. Каталог GEE містить кілька петабайтів інформації у хмарі: супутникових зображень та наборів даних щодо опадів, щільності населення, топографії, земного покриву та клімату. *Джерело:* https://geohackweek.github.io/GoogleEarthEngine/01-introduction/

Колекції даних у GEE – це групи геопросторових даних, що зберігаються і є доступними для аналізу й обробки. Вони містять різні типи даних, які охоплюють широкий спектр геопросторової інформації. Наприклад, це можуть бути супутникові зображення, метеорологічні дані, географічні векторні дані, растрові шари тощо.

Колекції даних GEE організовані та структуровані за типами інформації. Кожна колекція може містити безліч об'єктів зображень, які представляють окремі знімки або растрові шари з різних дат, датчиків або регіонів. Зображення в колекції мають властивості, як-от: географічні координати, дати, значення пікселів та метадані. Велика кількість даних зберігається в колекціях, що дає змогу використовувати їх для проведення різних аналітичних операцій та досліджень.

Однією з ключових особливостей колекцій даних GEE є їх доступність та широкий спектр геопросторових даних, що включають дані від різних датчиків та джерел. Наприклад, доступні колекції можуть охоплювати зображення із супутників Landsat, Sentinel, MODIS, аерофотознімки, географічні адміністративні кордони, цифрові моделі рельєфу та багато ін. Це дає змогу виконувати різноманітні аналітичні завдання, використовуючи різні типи даних із різних джерел.

Структура колекцій даних GEE

Колекції даних GEE мають чітку структуру, що складається з колекцій та об'єктів зображень. Кожна колекція містить набір зображень, які представляють собою дані з певного датчика, області чи типу. Об'єкти зображень містять відомості про географічні координати, дату і час отримання зображення, значення пікселів та інші атрибути, що характеризують зображення.

Доступні операції для роботи з колекціями даних GEE

GEE надає велику кількість операцій для роботи з колекціями даних. Ці операції можна розділити на декілька категорій:

- фільтрація даних: застосування фільтрів до колекцій даних для вибору зображень з певною датою, місцезнаходженням та іншими критеріями;
- обрізання даних: вибір пікселів зображення за допомогою області інтересу (ROI – region of interest) та їх збереження як нової колекції даних;
- редукція даних: обчислення статистичних характеристик колекції даних, таких як середнє значення, медіана, стандартне відхилення та ін.;
- візуалізація даних: створення зображень та відео з колекцій даних для візуалізації результатів аналізу;
- класифікація даних: застосування алгоритмів машинного навчання для класифікації зображень за допомогою певних атрибутів та створення карт класифікації.

Використовуючи ***Code editor*** *(Редактор коду)*, ви можете писати команди, які надсилаються як об'єкт до Google для паралельної обробки в хмарі (на стороні сервера). Користувачі можуть візуалізувати результати від Google у своєму браузері (на стороні клієнта), включаючи такі об'єкти, як карти, діаграми або статистичні результати. Використовуючи один з API, ви можете фільтрувати величезні колекції зображень за датами і територією, що вас цікавлять, накладати алгоритми на колекції зображень, застосовувати алгоритми до окремих зображень або колекцій зображень, а також обчислювати сукупну статистику в часі і просторі без необхідності завантажувати щось на свій комп'ютер (див. *рис. 4*).

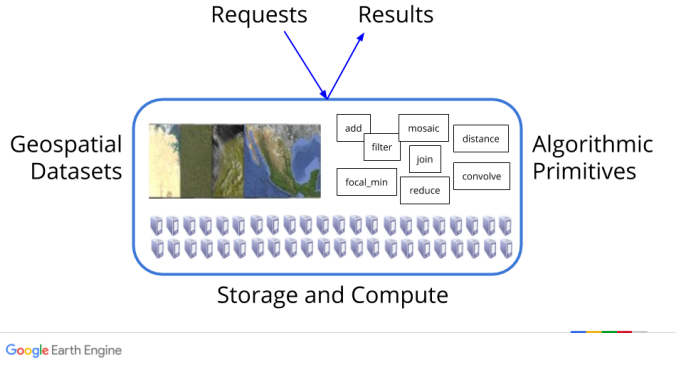


Рис. 4. Схематичне представлення роботи GEE, яке складається з формування запиту за допомогою алгоритмічних операцій. *Джерело:* https://geohackweek.github.io/GoogleEarthEngine/01-introduction/

У роботі Н. Горелика та ін. (2017) у таблиці 1 описано найбільш поширені набори даних. Ознайомтесь із ними для більш ширшого розкриття теми. На вебсайті GEE є загальні описи наборів даних, також ви можете переглядати набори даних безпосередньо через API GEE (див. *рис. 5*).

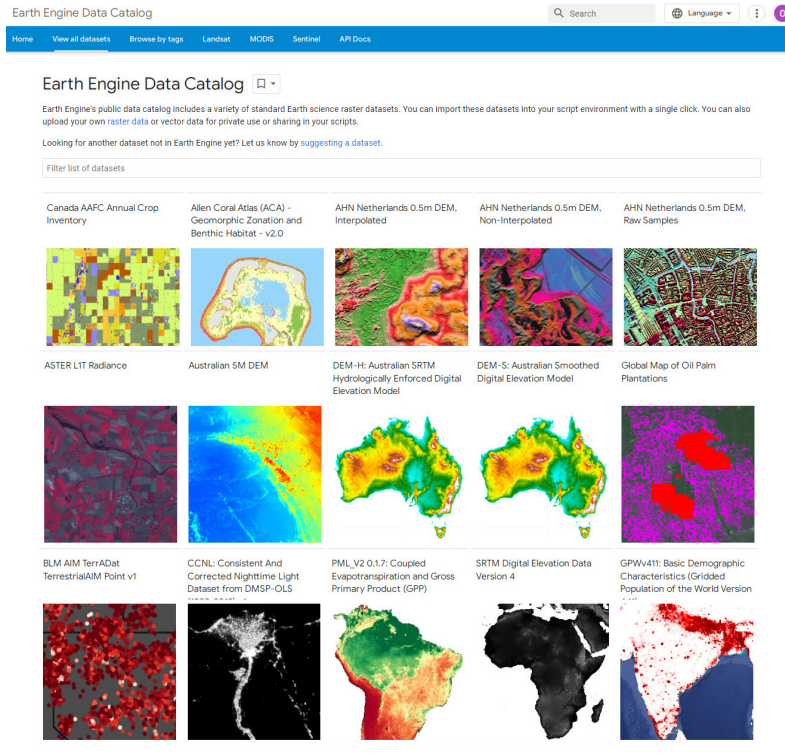


Рис. 5. Вигляд доступних тематичних даних у розділі Earth Engine Data Catalog на вебсайті GEE https://developers.google.com/earth-engine/datasets/catalog

Розглянемо приклад коду для отримання колекції супутникових зображень Landsat 8:

```
var collection = ee.ImageCollection('LANDSAT/LC08/C01/T1')
  .filterDate('2010-01-01', '2022-12-31')
  .filterBounds(geometry);

var firstImage = ee.Image(collection.first());
Map.addLayer(firstImage, {bands: ['B4', 'B3', 'B2'], min: 0, max: 3000}, 'Landsat-8');
```

```
// Вибір колекції Landsat 8

// Візуалізація першого зображення у колекції
```

Роль колекцій даних у GEE

Колекції даних у GEE є основною структурою для організації та роботи з геопросторовими даними. Розглянемо їх роль у GEE детальніше:

- Зберігання геопросторових даних: колекції даних GEE є місцем зберігання великої кількості геопросторових даних різних типів, як-от: супутникові зображення, кліматичні дані, картографічні дані та ін. Вони надають користувачам доступ до широкого спектра даних, що можуть бути використані для проведення аналізу та моделювання геопросторових процесів.
- Організація даних за датою та іншими атрибутами: колекції даних дають змогу організувати геопросторові дані за датою, регіоном, типом даних та іншими атрибутами. Це спрощує пошук та вибір даних, необхідних для аналізу, оскільки можна фільтрувати колекції за певними критеріями.
- Застосування функцій до колекцій: колекції даних GEE дають змогу застосовувати різноманітні функції та операції до всіх зображень у колекції одночасно. Наприклад, можна виконувати обчислення статистичних характеристик, візуалізувати дані, використовувати алгоритми класифікації та ін.
- Робота із часовими серіями даних: колекції даних у GEE особливо корисні для роботи із часовими серіями геопросторових даних. Вони дають змогу аналізувати зміни в даних протягом певного часу (за рік, за місяць, кожного понеділка протягом року тощо), проводити порівняння, виявляти тренди і залежності між різними змінними.
- Використання колекцій даних у скриптах та аналітичних завданнях: колекції даних у GEE є ключовим елементом під час написання скриптів та виконання аналітичних завдань. Вони дають змогу користувачам вибирати потрібні дані, застосовувати різні операції й алгоритми до колекцій, отримувати результати і візуалізувати їх.

Типи колекцій даних GEE

У GEE можна розділити колекції на два великих типи: векторні і растрові.

Векторні і растрові дані є двома основними форматами геопросторової інформації. Основна різниця між ними полягає в способі представлення та організації даних.

Векторні дані:

- векторні дані представляють географічні об'єкти за допомогою геометричних елементів, як-от точки, лінії або полігони;
- кожен геометричний елемент векторного шару має свої координати або набір координат, що визначають його місцезнаходження на карті;
- векторні дані описують об'єкти, такі як дороги, річки, кордони країн, будівлі та інші географічні об'єкти;
- векторні дані зазвичай мають більш точне геопросторове представлення, оскільки можуть зберігати деталі на рівні окремих геометричних елементів;
- завдяки геометричному характеру векторних даних можна виконувати точний аналіз геопросторових відносин, як-от перетини, злиття, попадання одного об'єкта в межі іншого.

Растрові дані:

- растрові дані представляють географічну інформацію у вигляді сітки пікселів, подібно до матриці;
- кожен піксель растрового зображення має свої значення, які відображають характеристики або атрибути цієї локації;
- растрові дані використовуються для представлення зображень, супутникових знімків, карт орографії, кліматичних даних тощо;
- растрові дані відображають географічні об'єкти шляхом значень пікселів, які можуть бути числовими даними, категоріями або кольорами;
- растрові дані зазвичай мають меншу точність геопросторового представлення порівняно з векторними даними, оскільки не можуть зберігати деталі на рівні окремих геометричних елементів;
- застосування растрових даних охоплює аналіз інтенсивності, класифікацію, виявлення змін у зображеннях та інші операції, які вимагають розгляду географічної інформації як набору значень пікселів.

Векторні дані і растрові дані мають свої переваги та недоліки у використанні, залежно від конкретного випадку. Обидва формати широко використовуються в геопросторовому аналізі та картографії, їх можна аналізувати як окремо, так і поєднувати в середовищі GEE (див. *рис. 6*).

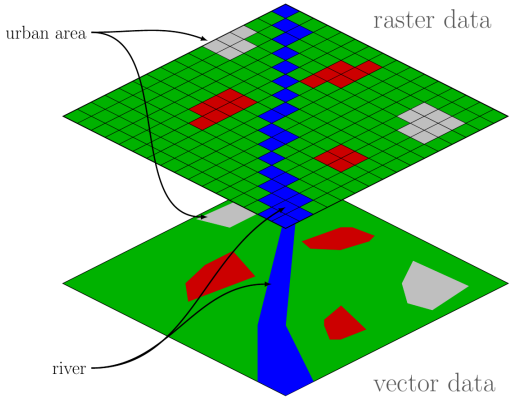


Рис. 6. Відображення растрових та векторних даних.
Джерело: Wegmann. 2010. https://commons.wikimedia.org/wiki/File:Raster_vector_tikz.png

2.2. Растрові колекції даних

Растрові колекції використовуються для роботи з геопросторовими растровими даними. Вони мають свою структуру і набір операцій, які дають змогу виконувати різноманітний аналіз та обробку растрових даних (див. *рис. 7*).

	Об'єднати	Зменшити	Композит та мозаїка	Перетворення	Графіки
	Join	Reduce	mosaic	reduceTo	Chart
Растрові дані					
Колекція зображень (Image Collection)	Так	Так	Ні	Так	Так
Зображення (Image)	Так	Ні	Так	Так	Так
Векторні дані					
Об'єкт, геометрія (Feature, Geometry)	Так	Так	Ні	Так	Так
Колекція об'єктів (Feature collection)	Так	Так	Ні	Так	Так

Рис. 7. Схема залежностей команд від типу даних (растрові і векторні колекції)

У GEE існують два основних типи даних для роботи з растровою інформацією: Image *(Зображення)* й ImageCollection *(Колекція зображень)*.

Image *(Зображення)*:

- Image у GEE представляє собою одне растрове зображення, яке може містити інформацію про різні канали, наприклад, кольорові канали (червоний, зелений, синій) або інші спектральні канали (залежно від технічних характеристик штучного супутника Землі кількість каналів знімка може варіюватися від трьох (Suomi NPP) до тринадцяти (Sentinel-2));
- зображення в GEE мають географічні координати і розмір, що відповідає кількості пікселів у горизонтальному та вертикальному напрямках;
- зображення може бути зчитане, оброблене, візуалізоване і використовуватися для аналізу та моделювання геопросторових процесів у GEE.

ImageCollection *(Колекція зображень)*:

- ImageCollection є набором растрових зображень, які групуються разом за темами;
- колекції зображень можуть містити багато зображень, які можуть бути отримані з різних дат, різних сенсорів або різних територій;
- колекції зображень використовуються для обробки й аналізу даних, що мають часову та просторову варіацію;
- вони можуть бути фільтровані за датою, територією, якістю зображення та іншими параметрами.

Різниця між Image та ImageCollection полягає в тому, що Image представляє одне конкретне растрове зображення, тоді як ImageCollection є набором багатьох растрових зображень, що можуть бути оброблені та проаналізовані як одне ціле.

Приклади використання Image й ImageCollection:

```
var singleImage = ee.Image('LANDSAT/LC08/C01/T1_SR/LC08_044034_20140318');
print(singleImage)

var imageCollection = ee.ImageCollection('LANDSAT/LC08/C01/T1_SR')
                  .filterDate('2019-01-01', '2019-12-31')
                  .filterBounds(geometry);
print(imageCollection)
```

```
// Зображення


// Колекція зображень
Задайте геометрію для змінної geometry
```

У прикладі вище singleImage є одним растровим зображенням, яке зчитується з набору даних Landsat 8. Із singleImage можна проводити різні операції аналізу або візуалізації.

ImageCollection є колекцією растрових зображень Landsat 8, що фільтруються за датою і територією. Ця колекція може використовуватися для аналізу різних зображень, які у ній представлені, з використанням таких методів, як фільтрація, обрізання, візуалізація тощо.

Отже, Image є окремим растровим зображенням, а ImageCollection – набором растрових зображень, які можна аналізувати як групу.

Ось приклад коду, який демонструє роль колекцій даних у GEE шляхом візуалізації зображень MODIS NDVI протягом певного періоду:

```
var collection =
ee.ImageCollection('MODIS/006/MOD13A2')
  .filterDate('2010-01-01', '2022-12-31')
.select('NDVI');

var meanNDVI = collection.reduce(ee.Reducer.mean());


Map.addLayer(meanNDVI, {min: -1, max: 1, palette: ['red', 'yellow', 'green']}, 'Mean NDVI')
```

```
// Вибір колекції MODIS NDVI


// Обчислення середнього NDVI для кожного зображення в колекції

// Візуалізація середнього NDVI
```

У цьому прикладі ми вибираємо колекцію зображень MODIS NDVI (Normalized Difference Vegetation Index) за певний період часу. Потім засто-

совуємо операцію reduce, використовуючи редуктор mean, для обчислення середнього NDVI для кожного пікселя у колекції. Нарешті, візуалізуємо середній NDVI, використовуючи палітру кольорів від червоного до зеленого, де високий NDVI відображається зеленим кольором, низький – червоним, а середні значення – жовтим кольором.

Цей приклад демонструє, як колекції даних у GEE дають змогу вибирати, обробляти та візуалізувати геопросторові дані, що відображають вегетаційний індекс NDVI протягом певного періоду.

Отже, один об'єкт Image в GEE є основним типом даних для роботи з растровими зображеннями. Він являє собою одне растрове зображення, яке може бути багатоканальним (містить інформацію про різні спектральні канали) або одноканальним (монохромне зображення).

Об'єднання двох зображень в одне за допомогою add():

```
var image1 = ee.Image('LANDSAT/LC08/C01/T1_TOA/LC08_044034_20140318');
var image2 = ee.Image('LANDSAT/LC08/C01/T1_TOA/LC08_044034_20140519');
var combinedImage = image1.add(image2);
Map.addLayer(combinedImage, {min: 0, max: 1}, 'Combined Image');
```

```
// Визначаємо два зображення та об'єднуємо їх за допомогою add()

// Додавання комбінованого зображення на мапу
```

2.3. Векторні колекції даних

Векторні дані – це тип геопросторових даних, що описують геометричні об'єкти у вигляді точок, ліній та полігонів. Вони використовуються для представлення реальних об'єктів, як-от: дороги, будівлі, річки, адміністративні межі, рельєф та багато інших елементів, які мають географічну прив'язку.

У GEE векторні дані, Geometry, Feature і FeatureCollection використовуються для роботи з геопросторовими об'єктами. Ось їхні визначення та різниця:

- Geometry *(Геометрія)*: використовується для представлення просторових форм векторних об'єктів. Вона може бути точкою, лінією, полігоном або набором цих елементів. Геометрія визначає просторові властивості об'єкта, як-от його розташування та форма.

- Feature *(Елемент)*: є основним компонентом векторних даних. Він являє собою окремий географічний об'єкт і має геометрію та атрибути. Геометрія елементу визначає його просторове положення, а атрибути містять різноманітні характеристики цього об'єкта (де описується додаткова інформація про нього). Наприклад, елемент може бути окремою будівлею, яка має геометрію полігона, й атрибути, як-от назва, тип, площа, кількість мешканців, кількість вікон тощо.
- Feature Collection *(Колекція елементів)*: є збіркою окремих елементів. Вона може містити різні типи геометрій та атрибути для кожного елемента. Колекція елементів зручна для організації й обробки геопросторових даних. Вона дає змогу працювати зі списком об'єктів, здійснювати операції фільтрації, злиття, вибору тощо.

Різниця між геометрією, елементом та колекцією елементів полягає у рівні організації та представлення геопросторових даних. Геометрія визначає форму об'єкта, елемент містить інформацію про цей об'єкт, а колекція елементів об'єднує багато елементів для легшої обробки та аналізу.

Основні компоненти векторних даних охоплюють:

- точки: використовуються для представлення одиночних місць на мапі, таких як адреси, географічні координати, ваш будинок, геологічні аномалії тощо;
- лінії: використовуються для представлення відрізків, шляхів, річок, доріг, залізниць та інших географічних об'єктів, які мають протяжність та напрямок;
- полігони: використовуються для представлення замкнутих областей, як-от: округи, адміністративні межі, лісові масиви, озера, гірські хребти та інші географічні об'єкти.

Кожен геометричний об'єкт векторних даних має географічні координати, що визначають його місцезнаходження на Землі. Крім того, векторні дані можуть мати атрибути, які містять додаткову інформацію про об'єкти. Наприклад, атрибути можуть містити назву, тип, площу, довжину, населення, висоту тощо. Векторні дані використовуються для різних цілей, включаючи аналіз географічних властивостей, створення карт, планування та прийняття рішень, визначення зон впливу, моделювання географічних процесів та багато ін. Вони є важливими джерелами інформації для геопросторового аналізу та розробки рішень.

У GEE векторні дані представлені у вигляді векторних колекцій, які містять набір геометрій з атрибутами. Це дає змогу робити складні запити, фільтрувати дані, виконувати геопросторовий аналіз та візуалізувати результати. Векторні колекції в GEE є важливим елементом для аналізу геопросторових даних. Вони представляють собою набір геометрій, які можуть бути використані для визначення територій інтересу, а також для аналізу та взаємодії з растровими даними.

Ось кілька ключових ознак та можливостей векторних колекцій у GEE:

- Представлення геометрій: векторні колекції можуть містити різні типи геометрій, як-от точки, лінії та полігони. Це дає змогу визначати і досліджувати лише території інтересу, лінії руху або адміністративні межі для аналізу та взаємодії з растровими даними.
- Метадані й атрибути: кожен об'єкт у векторній колекції може мати метадані і додаткові атрибути, як-от ідентифікатори, назви, дати, числові значення тощо. Ці атрибути можуть бути використані для фільтрації, класифікації та іншого аналізу даних.
- Операції над векторними колекціями: GEE надає набір операцій для роботи з векторними колекціями, як-от фільтрація, об'єднання, розбиття, візуалізація тощо. Ці операції дають змогу виконувати складні завдання, пов'язані з аналізом та обробкою векторних даних.
- Взаємодія з растровими даними: векторні колекції можуть бути використані для взаємодії з растровими даними. Наприклад, можна використовувати векторні полігони для обмеження областей аналізу растрових зображень або для обчислення статистики в межах певної території.
- Інтеграція із зовнішніми джерелами: векторні колекції можуть бути імпортовані із зовнішніх джерел даних, як-от файли форматів GeoJSON, Shapefile, KML тощо. Це дає змогу користувачам використовувати свої дані для аналізу та моделювання в GEE.

Приклад коду для створення та візуалізації векторної колекції:

```
var countriesUSDOS  = ee.FeatureCollection("USDOS/LSIB_SIMPLE/2017");                    // Створення змінної з країнами
var UkraineUSDOS = countriesUSDOS.filter(ee.Filter.eq('country_na', 'Ukraine'));        // Фільтрування датасету для відображення України
var styleParams_1 = {                                                                   // Параметри візуалізації
  fillColor: 'FF000000',
  color: 'FF0000',
  width: 1.0,
};
var coloredUSDOS = UkraineUSDOS.style(styleParams_1);                                   // Застосування параметрів візуалізації
Map.centerObject(UkraineUSDOS,5)
Map.addLayer(coloredUSDOS,{},'Ukraine Red',true);                                       // Додавання шару до мапи
```

У цьому прикладі ми звертаємося до наявної векторної колекції USDOS, вибираємо за атрибутом Україну і задаємо візуалізацію відображених даних.

Векторні колекції в GEE підтримують різноманітні операції, які дають змогу здійснювати обробку та аналіз геопросторових даних. Ось декілька прикладів операцій, які можна виконувати з векторними колекціями в GEE:

- Фільтрація: можна фільтрувати векторну колекцію за допомогою різних критеріїв, таких як географічні координати, атрибути (наприклад, значення атрибута або його діапазон), ідентифікатори тощо. Фільтрація дає змогу вибрати лише певні об'єкти, що відповідають заданим умовам.

Приклад коду для фільтрації векторної колекції за атрибутом:

```
var filteredCollection = vectorCollection.filter(ee.Filter.greaterThan('population', 1000000));   // Фільтрація векторної колекції за атрибутом "population"
```

- Злиття: злиття декількох векторних колекцій в одну дає змогу об'єднати їх геометрії та атрибути в одну колекцію. Це корисно, коли потрібно об'єднати окремі шари даних або додати нові атрибути до наявної колекції.

Приклад коду для злиття двох векторних колекцій:

```
var mergedCollection = geometry.
merge(geometry2);
Map.addLayer(mergedCollection)
print(mergedCollection)
```

```
// Злиття двох векторних колекцій.
У редакторі коду необхідно створити
два об'єкти з відповідними назвами
і задати їм тип Feature Collection
// Додавання шару на мапу
```

- Перетин та об'єднання: можна виконувати операції перетину та об'єднання між векторними колекціями. Перетин повертає тільки об'єкти, які перетинаються з обох колекцій, тоді як об'єднання поєднує всі об'єкти з обох колекцій.

Приклад коду для перетину двох векторних колекцій:

NOTE не працюватиме без налаштувань типу даних.

```
var polygonIntersection = geometry.
intersection(geometry2, 1);

Map.addLayer(geometry,
        {'color': 'black'},
        'Geometry [black]: polygon');
Map.addLayer(geometry2,
        {'color': 'blue'},
        'Parameter [blue]: inputGeom');
Map.addLayer(polygonIntersection,
        {'color': 'red'},
        'Result [red]: polygon.intersection');
```

```
// Створення двох об'єктів із
типом даних Feature і нанесення
їх на мапу так, щоб вони пере-
тиналися.
```

Це лише кілька прикладів операцій, які можна виконувати з векторними колекціями в GEE. Цей хмарний сервіс надає багато інших можливостей для роботи з векторними даними, як-от знаходження перетину, обчислення відстані, класифікація тощо.

У GEE ви маєте можливість підключити власні дані для аналізу та обробки. Це дає змогу використовувати власні географічні дані разом із великим асортиментом доступних даних у GEE. Існує кілька способів підключення власних даних.

Завантаження даних із локального джерела: ви можете завантажити ваші географічні дані з локального комп'ютера безпосередньо в GEE. Для цього скористайтеся інструментом ***Upload*** *(Завантажити)* в редакторі коду або використовуйте API для завантаження даних.

Приклад коду для завантаження файлу з локального комп'ютера: перейдіть до ***Assets*** *(Менеджер об'єктів)* і виберіть ***Image Upload*** *(Завантаження зображення)* для того, щоб завантажити растрові дані до платформи GEE (див. *рис. 8–13*).

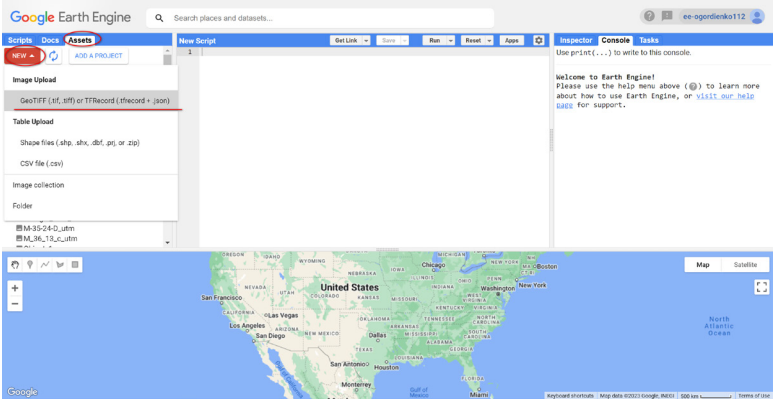


Рис. 8. Вибір растрових даних для завантаження в робоче середовище

У випадному вікні виберіть необхідні файли.

Важливо! *Ідентифікатор для цього файлу може містити лише літери, цифри, дефіси та підкреслення.*

Натисніть UPLOAD *(Завантажити)*.

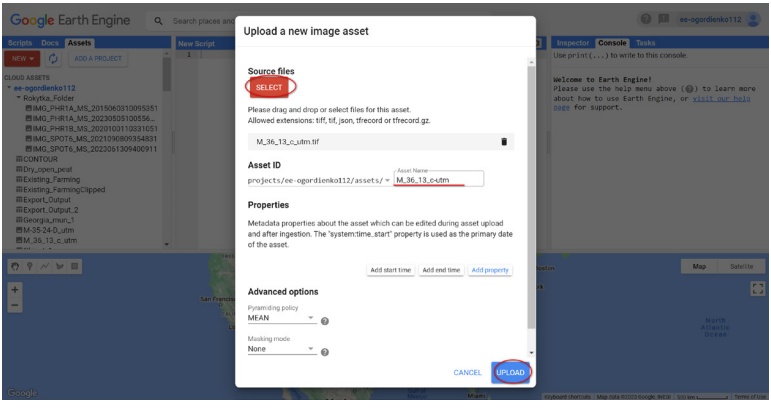


Рис. 9. Вибір необхідних растрових даних

Прогрес з’явиться в ***Tasks*** *(Менеджер завдань)*.

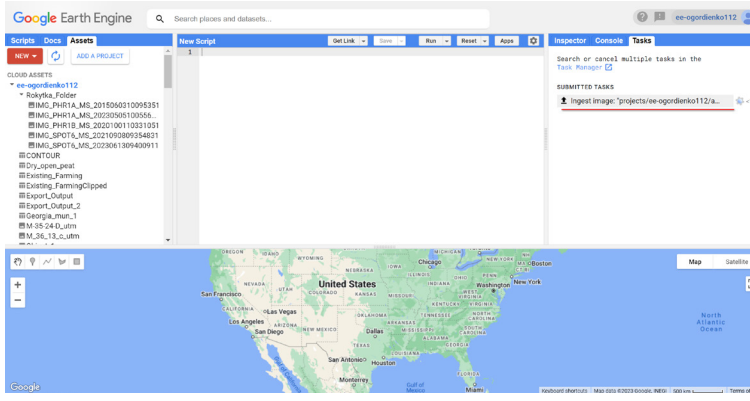


Рис. 10. Відображення завантаження растра в менеджері завдань

Для завантаження векторних даних виберіть ***Table Upload*** *(Завантаження таблиць)* у меню.

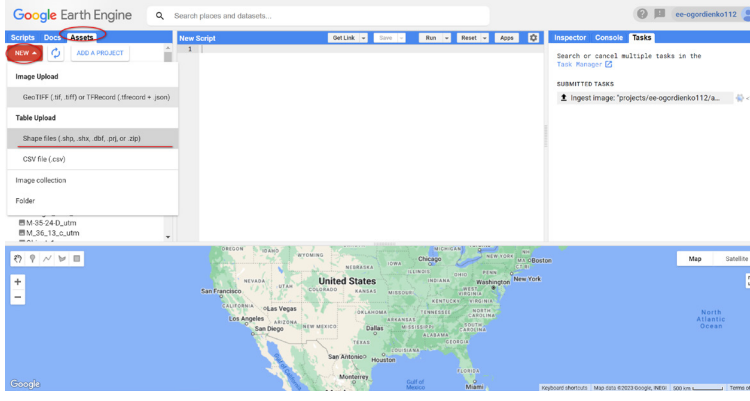


Рис. 11. Вибір векторних даних для завантаження в робоче середовище

У вікні, що випадає, виберіть необхідні файли.

Важливо! *Ідентифікатор для цього файлу може містити лише літери, цифри, дефіси та підкреслення.*

Важливо! *Завантажуючи Shape-файл, ви можете скористатися архівом або завантажити всі необхідні складові файлу.*

Натисніть UPLOAD *(Завантажити)*.

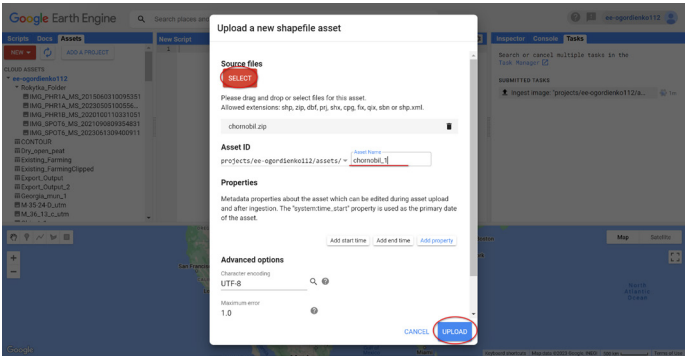


Рис. 12. Вибір необхідних векторних даних

Прогрес з'явиться в ***Tasks*** *(Менеджер завдань)*.

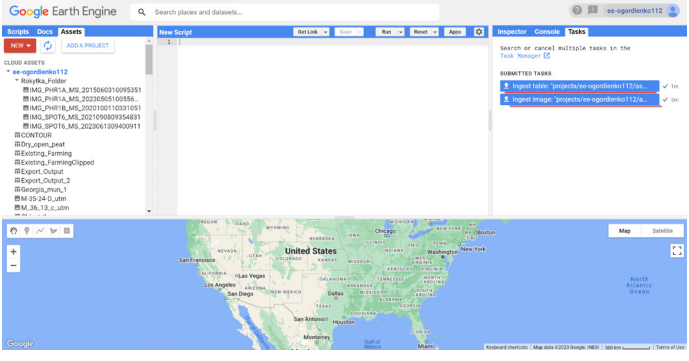


Рис. 13. Відображення завантаження вектора в менеджері завдань

Процес ідентичний для векторних і растрових даних.

GEE надає широкий спектр можливостей для роботи з геопросторовими даними, включаючи векторні і растрові колекції. Колекції даних є ключовим елементом GEE, вони дозволяють доступ до великого обсягу геопросторової інформації для аналізу та моделювання.

Розділ 3

ФІЛЬТРУВАННЯ КОЛЕКЦІЙ

3.1. Фільтрування колекцій об'єктів

Колекції даних – це набір геопросторових даних, які можуть охоплювати різні типи інформації, наприклад: зображення з космосу (супутникові знімки), картографічні дані, кліматичні дані та ін. **FeatureCollection** *(Векторні колекції)* дають змогу представляти географічні об'єкти за допомогою геометричних елементів, таких як точки, лінії або полігони. Вони забезпечують точний аналіз геопросторових відносин і дають змогу виконувати операції злиття, фільтрації та ін. **ImageCollection** *(Растрові колекції)* використовуються для представлення географічної інформації у вигляді сітки пікселів, дають змогу виконувати операції аналізу і класифікації зображень, створювати карти висот, кліматичних даних та іншої географічної інформації. Загальною метою використання колекцій даних у GEE є аналіз геопросторової інформації та отримання нових знань про наше середовище. Це потужний інструмент, який дає змогу досліджувати та моделювати різноманітні географічні процеси для розв'язання певних проблем і робити важливі висновки.

Фільтрування колекцій у GEE допомагає звузити коло даних, які ви хочете проаналізувати або відобразити. Існує кілька способів фільтрування колекцій в Earth Engine, зокрема часова та просторова фільтрація, а також фільтрація на основі властивостей (атрибути, метадані тощо).

Часова фільтрація дає змогу вибрати дані на основі певного часового діапазону. Наприклад, ви можете відфільтрувати колекцію знімків Sentinel-2, щоби показати дані лише за певний місяць або рік. Для фільтрації за певним діапазоном дат використовують функцію filterDate().

Просторова фільтрація дає змогу вибрати дані на основі певного географічного регіону. Ви можете використовувати функцію filterBounds() для фільтрації за певною областю, що вас цікавить. Ця функція приймає як аргумент геометричний об'єкт, наприклад: точку, лінію, полігон або обмежувальну рамку.

Фільтрація на основі властивостей дає змогу відбирати дані на основі значень певних властивостей, пов'язаних із даними. Наприклад, ви можете відфільтрувати колекцію зображень, щоби показати тільки дані з певним відсотком хмарності або супутниковим датчиком. Ви можете використовувати функцію filter() для фільтрації за значеннями властивостей.

На додаток до цих базових методів фільтрації ви також можете комбінувати їх для створення більш складних, специфічних фільтрів. Наприклад, можна відфільтрувати колекцію зображень Landsat так, щоб показати дані лише з

певного регіону та часового діапазону, комбінуючи функції filterBounds() та filterDate().

Загалом фільтрування колекцій у GEE дає змогу зосередитися на даних, які є найбільш релевантними для аналізу, що полегшує пошук корисної інформації та інсайтів.

Фільтрація колекцій векторних об'єктів (**FeatureCollection**) у GEE дає змогу вибрати підмножину об'єктів, які відповідають певним критеріям. Колекції об'єктів часто є великими і містять багато інформації, тому фільтрація необхідна для виділення та аналізу об'єктів, які є найбільш релевантними для конкретного аналізу.

Існує кілька методів фільтрації колекцій об'єктів у Earth Engine. Найпоширеніші методи базуються на значеннях атрибутів, просторовому розташуванні та метаданих. Конкретна функція, що використовується для фільтрації, залежить від типу необхідного фільтра. Розглянемо декілька прикладів фільтрування об'єктів.

Для того щоб виконати фільтрацію за атрибутами, використовуйте ee.FeatureCollection.filter() для відбору об'єктів на основі значень певних атрибутів, як-от землекористування або щільність населення. Нижче цей тип фільтрації розглянуто на прикладі виділення території Харківської області з колекції Глобальних шарів адміністративних одиниць (Global Administrative Unit Layers, GAUL) [10]. Ця колекція збирає і поширює найбільш повну доступну інформацію про адміністративні одиниці для всіх країн світу. Перед тим як починати фільтрацію колекції об'єктів за атрибутами, слід ознайомитися з таблицею атрибутів, щоб дізнатися їх імена, які будуть застосовуватися під час створення фільтра. Її завжди можна знайти на сторінці з описом у вкладці ***Table Schema*** *(Структура атрибутивної таблиці)* (див. *рис. 14).*

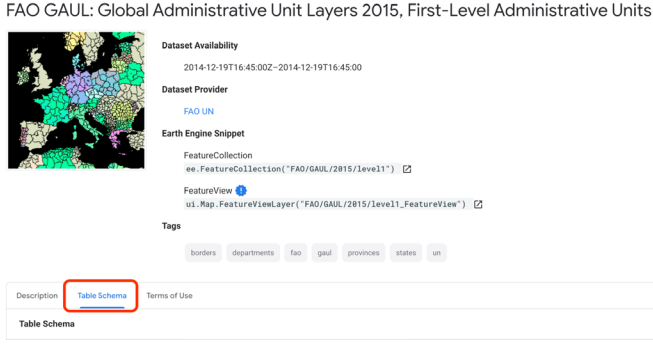


Рис. 14. Доступ до таблиці атрибутів колекції об'єктів

```
var targetRegionName = "Kharkivs'ka"
var boundaries =
ee.FeatureCollection('FAO/
GAUL/2015/level1')
.filter(ee.Filter.eq('ADM1_NAME',
targetRegionName));
print(boundaries);
Map.addLayer(boundaries);
```

```
// Визначення назви області
// Додавання колекції об'єктів з кордонами країн та областей до проєкту

// Фільтрування колекції за атрибутом "ADM1_NAME" (назва адміністративного регіону)
// Відображення результатів фільтрування
```

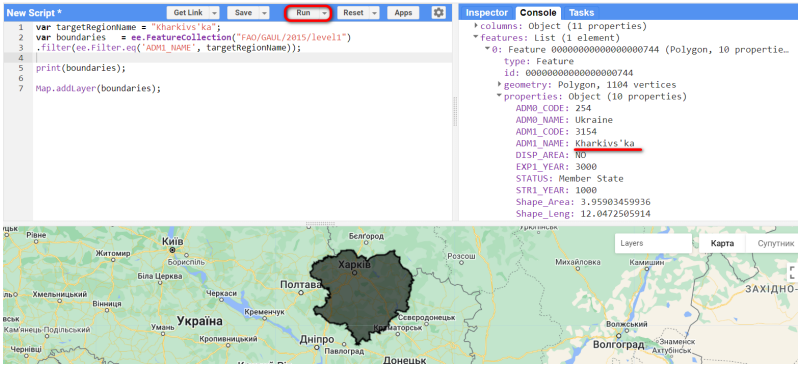


Рис. 15. Результат фільтрування колекції об'єктів за атрибутом

Також можна довільно комбінувати розглянуті фільтри для створення більш складних фільтрів, які відповідають конкретним вимогам. Наприклад, ви можете відфільтрувати колекцію даних так, щоб вибрати області України лише із площею, що перевищує певне значення *(див. рис. 16)*.

```
var boundaries =
ee.FeatureCollection("FAO/
GAUL/2015/level1")
var minArea = 3;

var UkraineRegions = boundaries.
filter(ee.Filter.and(
  ee.Filter.eq('ADM0_NAME', 'Ukraine'),
ee.Filter.gt('Shape_Area', minArea)
));
print(UkraineRegions);
Map.addLayer(UkraineRegions);
```

```
// Додавання колекції об'єктів з кордонами країн та областей до проєкту

// Вибір значення площі для фільтрації (у 10⁴ км²)
// Вибір областей України


// Фільтр за площею, більший, ніж задане значення
// Відображення результатів фільтрування
```

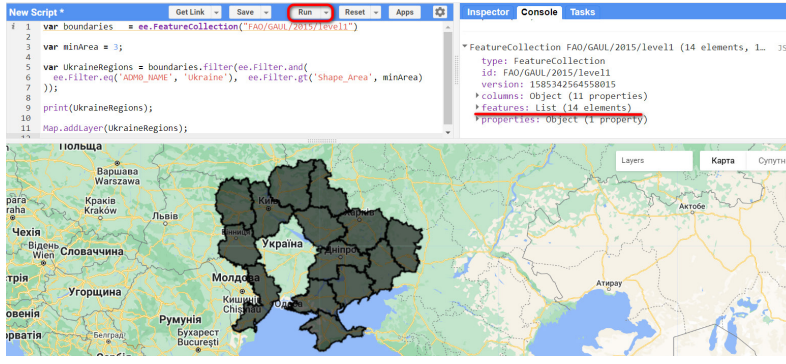


Рис. 16. Результат фільтрування колекції об'єктів за допомогою комбінованого фільтра

Після того як ви застосували фільтри до колекції об'єктів, ви можете виконувати різні операції з відфільтрованою колекцією, наприклад, обчислювати зведену статистику, застосовувати алгоритми або візуалізувати об'єкти. Відфільтровану колекцію можна використовувати як вхідні дані для інших функцій Earth Engine, як-от ee.FeatureCollection.reduce() або ee.FeatureCollection.map(), для подальшої обробки об'єктів.

3.2. Фільтрування колекцій зображень

Фільтрація колекцій зображень у GEE (**ImageCollection**) – це потужний інструмент для вибору певних зображень або підмножин зображень із більшої колекції. Колекції зображень часто бувають великими (за об'ємом інформації) і містять багато зображень, тому фільтрація необхідна для аналізу зображень, які є найбільш релевантними для конкретного випадку.

Розглянемо фільтрування колекцій на прикладі архіву супутникових даних із Sentinel-2 з рівнем обробки L2A (дані, що пройшли атмосферну корекцію). Ці дані доступні з 2017 р., а це означає, що в цьому архіві за замовчуванням колекція зображень містить усі зображення, доступні з 2017 р. і дотепер по всьому світу. Тепер ми хочемо переглянути метадані зображень. Для дослідження зображень в Earth Engine передбачена функція друку print(), яка виводить метадані зображень.

```
var Imagery =
ee.ImageCollection("COPERNICUS/
S2_SR_HARMONIZED")
print(Imagery)
```

```
// Додавання колекції зображень
Sentinel-2 до проєкту

// Виведення інформації про колек-
цію зображень Sentinel-2 у консоль
```

Функція друку print() повертає максимум 5000 елементів за один раз. Якщо ваша колекція зображень містить більше 5000 зображень, ви не зможете їх вивести в консоль. Earth Engine видає помилку (див. *рис. 17*).

Рис. 17. Помилка при відображенні інформації про колекцію зображень

Ми можемо відфільтрувати колекцію зображень супутника, щоб зменшити кількість знімків у нашій колекції. Адже архів даних Sentinel-2 L2A містить дані з усього світу, починаючи з 2017 р. і до сьогодні. Ми не можемо опрацювати дані для всього світу за весь час (це дуже великий об'єм інформації). Нас радше буде цікавити певна місцевість чи регіон. Отже, ми можемо зосередитися на цьому конкретному місці і відфільтрувати колекцію знімків, щоб отримати зображення для цієї конкретної місцевості.

Фільтр також зменшує час обчислень, оскільки ми збираємо лише обмежену кількість зображень, необхідних для нашої роботи.

Існує кілька методів фільтрації колекцій зображень в Earth Engine. Найпоширеніші методи базуються на часі, просторі та метаданих. Конкретна функція, що використовується для фільтрації, залежить від типу необхідного фільтра. Розглянемо декілька практичних прикладів.

Для виконання **часової фільтрації** використовуйте ee.ImageCollection.filterDate(), щоб вибрати зображення в межах певного діапазону дат, наприклад, всі зображення, отримані в певний місяць або рік.

```
var Imagery =
ee.ImageCollection('COPERNICUS/
S2_SR_HARMONIZED');
var Imagery2022 = Imagery.
filterDate('2022-01-01', '2022-01-02')
  .filter(ee.Filter.lt('CLOUD_
COVERAGE_ASSESSMENT',1));
print(Imagery2022)
```

```
// Додавання колекції зображень
Sentinel-2 до проєкту

// Фільтрація колекції зображень
Sentinel-2 за датами
```

Ми також можемо використовувати функцію ee.Filter.date замість filterDate.

```
var Imagery =
ee.ImageCollection("COPERNICUS/
S2_SR_HARMONIZED");

var Imagery2022 = Imagery.filter(ee.
Filter.date('2022-01-01', '2022-12-31'));
```

```
// Додавання колекції зображень
Sentinel-2 до проєкту

// Фільтрація колекції зображень
Sentinel-2 за датами
```

Просторова фільтрація здійснюється за допомогою ee.ImageCollection.filterBounds() для відбору зображень, які перекривають певну територію інтересу, наприклад, досліджувану територію або певну особливість (тільки гірську місцевість) на поверхні Землі.

```
var boundaries =
ee.FeatureCollection('FAO/
GAUL/2015/level1');
var Kharkivska = boundaries.
filter(ee.Filter.eq('ADM1_NAME',
"Kharkivs'ka"));
var Imagery =
ee.ImageCollection('COPERNICUS/
S2_SR_HARMONIZED');
var Kharkivska_image = Imagery.
filterBounds(Kharkivska)
.filter(ee.Filter.lt('CLOUD_
COVERAGE_ASSESSMENT',1));
print(Kharkivska_image)
```

```
// Додавання колекції об'єктів з
кордонами країн та областей до
проєкту

// Фільтрація колекції об'єктів за
атрибутами (обираємо Харківську
область)

// Додавання колекції зображень
Sentinel-2 до проєкту

// Фільтрація колекції зображень
Sentinel-2 за областю інтересів
```

Для **фільтрації за метаданими** використовують ee.ImageCollection.filter() для відбору зображень на основі певних властивостей метаданих, як-от тип сенсора або відсоток хмарності.

```
var Imagery =
ee.ImageCollection("COPERNICUS/
S2_SR_HARMONIZED");

var Imagery_cloudless = Imagery.
filter(ee.Filter.lt('CLOUDY_PIXEL_
PERCENTAGE', 20));
```

```
// Додавання колекції зображень
Sentinel-2 до проєкту

// Фільтрація колекції зображень
Sentinel-2 за метаданими (відсоток
хмарності знімка менший за 20%)
```

Також можна комбінувати фільтри для створення більш складних фільтрів, які відповідають певним вимогам, аби підібрати знімки конкретно під ваше дослідження. Наприклад, ви можете відфільтрувати колекцію зображень, щоб вибрати лише безхмарні знімки, отримані протягом певного періоду часу в певному місці.

У наведеному нижче прикладі коду *(див. рис. 18)* ви побачите, що можна «зв'язати» команди фільтрації, які потім виконуються зліва направо. Нижче ми з'єднаємо фільтри в ланцюжок у тому самому порядку, який ми описали вище. Зверніть увагу, що це дає ту саму колекцію зображень і того самого розміру, що і в разі застосування фільтрів по одному (по черзі).

```
var boundaries  =
ee.FeatureCollection("FAO/GAUL/2015/
level1");

var Kharkivska = boundaries.filter(ee.Filter.
eq("ADM1_NAME", "Kharkivs'ka"));

var Imagery =
ee.ImageCollection('COPERNICUS/S2_SR')
  .filterDate('2022-01-01','2022-12-31')
  .filterBounds(Kharkivska)
  .filter(ee.Filter.lt('CLOUDY_PIXEL_
  PERCENTAGE', 20));

print(Imagery);
Map.addLayer(Imagery);
```

```
// Додавання колекції
об'єктів з кордонами країн та
областей до проєкту

// Фільтрація колекції об'єктів
за атрибутами (обираємо
Харківську область)

// Фільтрація колекції зобра-
жень Sentinel-2 за складним
фільтром (дата, область
інтересів і відсоток хмарності
зображення)

// Відображення результатів
фільтрування
```

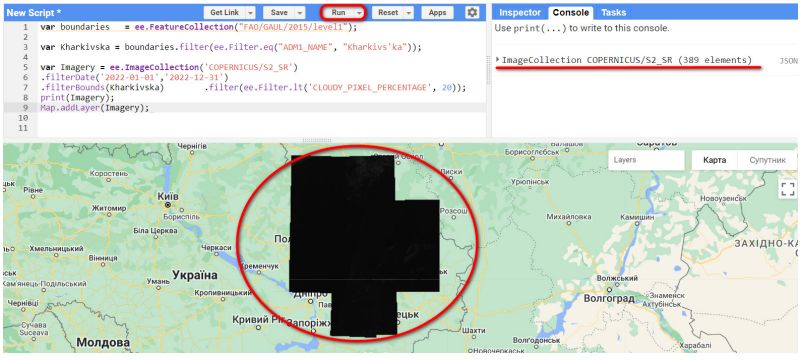


Рис. 18. Результат фільтрування колекції зображень за допомогою комбінованого фільтра

Цікаво

Після застосування фільтрів до колекції зображень ви можете виконувати різні операції з відфільтрованою колекцією, наприклад: обчислювати зведену статистику, застосовувати алгоритми або візуалізувати зображення. Відфільтровану колекцію можна використовувати як вхідні дані для інших функцій Earth Engine, як-от ee.ImageCollection.reduce() *або* ee.ImageCollection.map()*, для подальшої обробки зображень.*

3.3. Сортування

Крім розглянутих у попередніх підрозділах способів фільтрування в GEE існує ще один метод впорядкування даних – **сортування**, який реалізується за допомогою функції .sort(). Він може бути корисним, наприклад, якщо потрібно розташувати зображення в певному порядку, відсортувавши колекцію за заданою властивістю.

Як правило, щоб вирішувати практичні задачі опрацювання колекцій даних, поєднують методи фільтрування і сортування для більш ефективної роботи з інформацією. Наприклад, відомо, що хмарний покрив на зображеннях є серйозною перешкодою для аналізу даних дистанційного зондування Землі. Часто потрібно знайти зображення з найменшим відсотком хмарності або вибрати зображення з певним допустимим рівнем хмарності. Розглянемо приклад вирішення такої задачі в GEE за допомогою функцій фільтрування і сортування (див. *рис. 19*):

```
var col = ee.ImageCollection('LANDSAT/LC08/C02/T1_TOA')
  .filterBounds(ee.Geometry.Point(35, 50))
  .filterDate('2022-07-01', '2022-09-01');
print('Колекція', col);
var colCldSortAsc = col.sort('CLOUD_COVER');
print('Відсортовані в порядку зростання відсотка хмарності зображення:', colCldSortAsc);
var visParams = {
  bands: ['B4', 'B3', 'B2'],
  min: 0.01,
  max: 0.25
};
Map.setCenter(35, 50, 6);

Map.addLayer(colCldSortAsc.first(), visParams, 'Знімок із найнижчим значенням хмарності');
```

```
// Колекція зображень Landsat 8 TOA (відбір зображень для заданого часового періоду і певної точки інтересу за допомогою фільтрування)
// Відображення списку зображень
Сортування зображень у порядку зростання відсотка хмарності зображення
Відображення відсортованого списку зображень
// Задання параметрів візуалізації шару


// Центрування і масштабування карти
// Знаходження і відображення зображення з найменшим рівнем хмарності
```

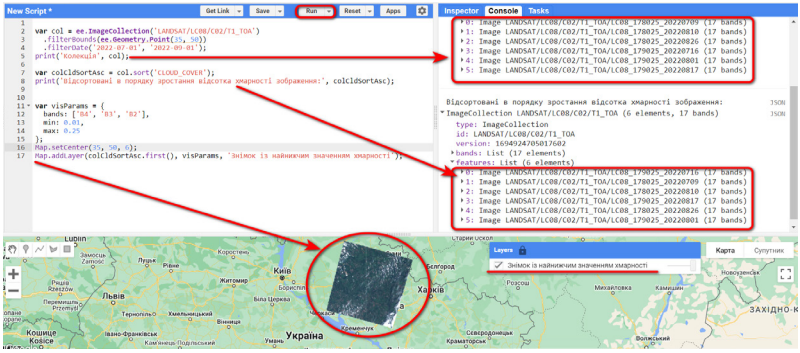


Рис. 19. Результат застосування методів фільтрування і сортування колекції зображень

У розглянутому прикладі відібрано супутникові знімки колекції Landsat 8 TOA для періоду липень – серпень 2022 р. із прив'язкою до заданої точки інтересу із заданими координатами (35 град. сх. д., 50 град. пн. ш.). Створена колекція знімків містить шість зображень, які були відсортовані за відсотком хмарності (атрибут CLOUD_COVER). Зображення з найнижчим показником хмарності відображене у вигляді шару на карті.

> **Цікаво**
>
> *Як і у випадку з фільтрами, можна також застосовувати сортування за різними властивостями, використовуючи різні методи sort() для однієї і тієї самої колекції, якщо це необхідно. Важливо зауважити, що сортування відбувається послідовно, і перше сортування визначає основний порядок сортування.*

3.4. Редуктори

Редуктори в GEE – це інструменти, які допомагають поєднувати й узагальнювати інформацію з різних джерел, таких як зображення, карти і таблиці даних. Їх слід розглядати як набір правил, які визначають, як поєднувати й узагальнювати дані в процесі аналізу. Ці правила можуть бути простими, як-от пошук найменшого або найбільшого числа в групі, або складними – створення діаграми або пошук тенденцій у даних.

Ці правила можна використовувати в процесі аналізу даних у різний спосіб, враховуючи аспекти часу (наприклад, розглядаючи зміни в серії зобра-

жень), простору (наприклад, вивчаючи конкретну територію на карті), або певних атрибутів даних (наприклад, зосереджуючись на важливих якостях чи характеристиках). Можна сказати, що редуктор надає єдиний набір інструментів для опрацювання всього обсягу доступної інформації. Коли ми використовуємо редуктор, він приймає дані на вході і дає нам єдиний результат на виході (див. *рис. 20*).

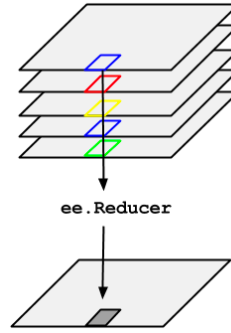


Рис. 20. Застосування редуктора до колекції зображень у Google Earth Engine

Якщо ми використовуємо редуктор на багатоканальному зображенні (яке схоже на кольоровий рисунок із багатьма шарами), то GEE автоматично застосовує редуктор до кожного шару окремо. Отже, якщо в нас було три шари на вхідному зображенні, ми отримаємо три шари на вихідному зображенні, і на кожному з них буде показано отриманий результат опрацювання даних відповідного шару вхідного зображення.

Слід ураховувати, що деякі редуктори потребують більше одного набору даних для роботи, наприклад для виконання складних обчислень, як-от лінійна регресія. У таких випадках GEE не повторює автоматично обчислення для кожного шару зображення. Нам потрібно надати йому правильні дані в певному порядку. Крім того, деякі редуктори є особливими, оскільки вони повертають більше одного результату. Наприклад, вони можуть визначити найменше і найбільше значення в наших даних або створити гістограму (діаграму, що показує, як розподілені дані) чи список значень. Отже, коли ми використовуємо ці спеціальні редуктори, ми отримуємо результат, що складається з кількох частин.

До редукторів, які найбільш часто використовуються у GEE, належать:

- ee.Reducer.mean(): обчислює середнє значення;
- ee.Reducer.median(): знаходить медіану;
- ee.Reducer.min(): знаходить мінімальне значення;
- ee.Reducer.max(): знаходить максимальне значення;
- ee.Reducer.sum(): обчислює суму значень;
- ee.Reducer.frequencyHistogram(): обчислює гістограму частот;
- ee.Reducer.toList(): перетворює значення пікселів на список;
- ee.Reducer.stdDev(): обчислює стандартне відхилення;
- ee.Reducer.variance(): обчислює дисперсію.

Ці редуктори можна використовувати з функціями reduceRegion() і reduceRegions() для зональної статистики, а також із функціями reduce() і reduceToImage() для агрегації для заданого часового проміжку.

Розглянемо декілька прикладів практичного застосування різних редукторів для роботи з колекціями геопросторових даних.

Застосування редуктора ee.Reducer.mean() для визначення зональної статистики (середнього значення) для пікселів у межах області інтересу (див. *рис. 21*):

```
var roi = ee.Geometry.Polygon(                    // Визначення області інтересу
  [[[33.02, 50.05],                               (ROI) як геометричного об'єкта
    [33.09, 50.05],
    [33.09, 50.07],
    [33.02, 50.07]]]);
var image = ee.Image('LANDSAT/LC08/               // Завантаження одного зобра-
C02/T1_TOA/LC08_179025_20220716')                 ження Landsat 8 за певну дату та
  .clip(roi);                                     область.
                                                  // Обмеження зображення областю
var meanResult = image.                           інтересу.
reduceRegion({                                    // Обчислення середнього значення
  reducer: ee.Reducer.mean(),                     для всіх каналів у вказаній області.
  geometry: roi,
  scale: 30,                                      // Масштаб
  maxPixels: 1e9                                  // Максимальна кількість пікселів
});                                               для обробки.
print('Середні значення для всіх                 // Виведення результату обчислен-
каналів зображення в межах області                ня середнього значення.
інтересу:', meanResult);
Map.addLayer(image);                              // Додавання зображення на карту.
```

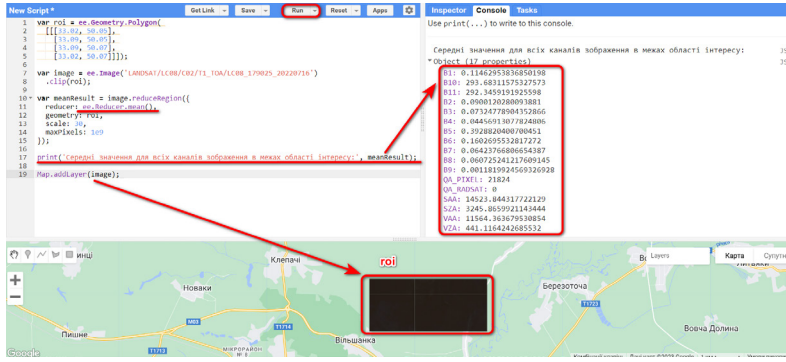


Рис. 21. Результат застосування редуктора ee.Reducer.mean() до колекції зображень для визначення зональної статистики

У розглянутому вище прикладі створений код виконує операції з обробки та аналізу зображення Landsat 8 для заданої області інтересу за допомогою редуктора й виводить середні значення для всіх каналів на карту і в консоль.

Застосування редуктора ee.Reducer.sum() для визначення сумарного значення для пікселів у межах області інтересу протягом визначеного часового проміжку (див. *рис. 22*):

```
var roi = ee.Geometry.Polygon(
  [[[33.02, 54.05],
    [36.09, 54.05],
    [36.09, 50.07],
    [33.02, 50.07]]]);
var precipCollection =
ee.ImageCollection("NASA/GPM_L3/
IMERG_V06")
          .select('precipitationCal')
          .filterDate('2022-01-01', '2022-01-31')
          .filterBounds(roi);

print(precipCollection, 'Дані спостережень
за кількістю опадів');
var annualPrecip = precipCollection.
reduce(ee.Reducer.sum());
print(annualPrecip);

var annualPrecip2 = precipCollection.sum();
var annualPrecip3 = annualPrecip2.clip(roi);
var precipPal = ['white','blue']

Map.addLayer(annualPrecip3, {min: 60,
max: 150, palette: precipPal}, ‹Сумарна
кількість опадів›);
```

```
// Визначення області інтересу
(ROI) як геометричного об'єкта

// Завантаження щоденних да-
них опадів за січень 2022 року
для області інтересу
// Вибір каналу опадів
// Задання часового проміжку
// Обмеження зображення об-
ластю інтересу
// Виведення ImageCollection та
його імені для перевірки
// Обчислення сумарної кілько-
сті опадів для всіх днів у січні

// Обчислення сумарної кілько-
сті опадів і обрізка за новою зо-
ною інтересу
// Визначення палітри кольорів
для візуалізації результату
// Додавання зображення на
карту
```

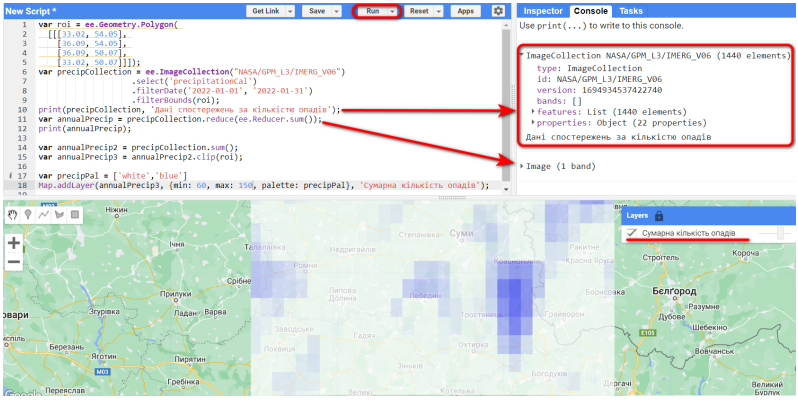
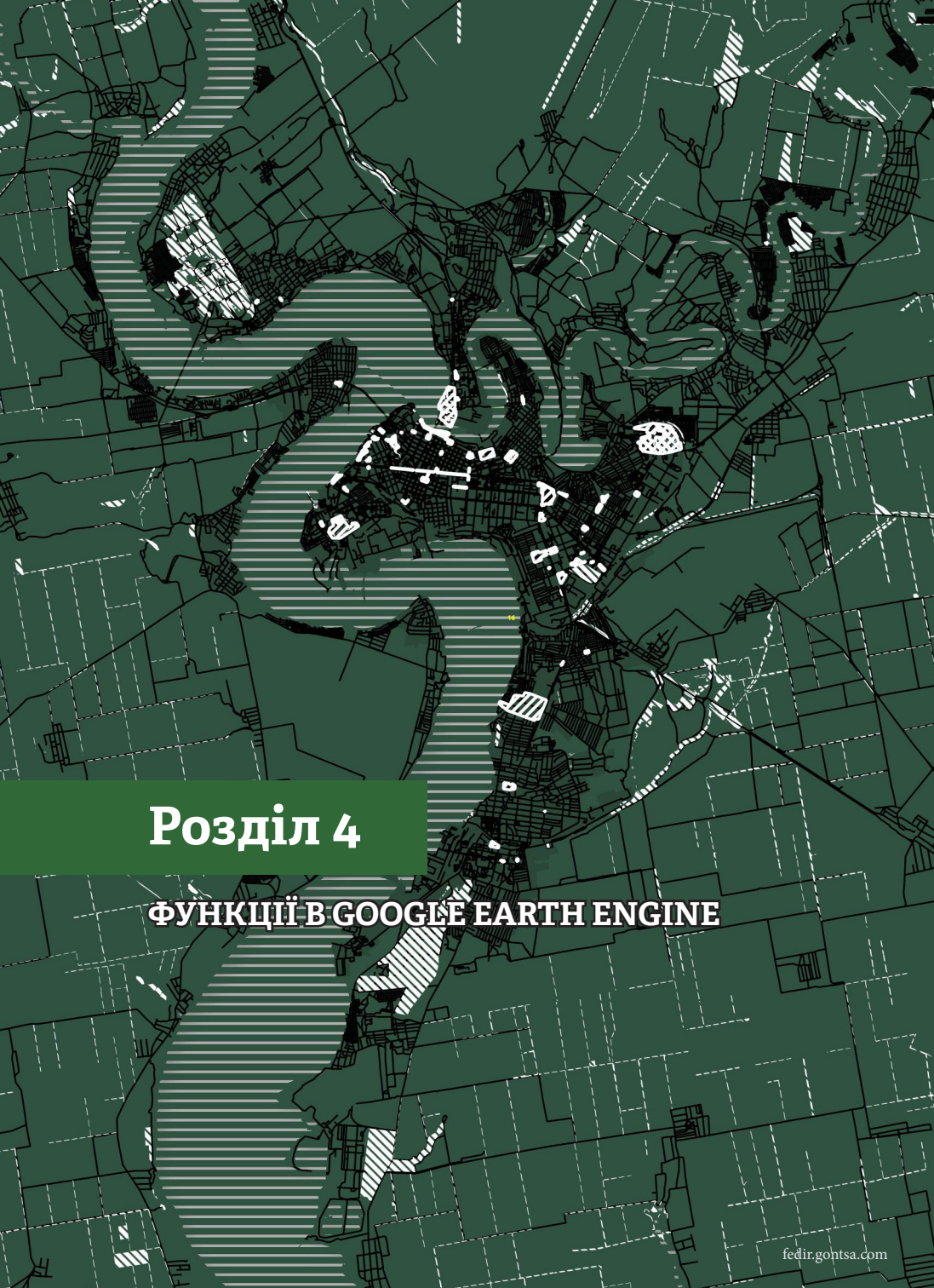


Рис. 22. Результат застосування редуктора ee.Reducer.sum() до колекції зображень для часової агрегації даних

Запропонований вище код завантажує дані колекції NASA/GPM_L3/IMERG_V06, яка містить інформацію про добові дані опадів (кількість опадів у мм), і за допомогою редуктора обчислює сумарну кількість опадів січня 2022 р. для кожного пікселя в межах заданої області інтересу.

Отже, в GEE редуктори – це функції, які використовуються для агрегації даних, визначення зональної статистики чи оцінки мінливості протягом періоду часу досліджуваних параметрів колекцій растрових зображень або векторних об'єктів. Редуктори допомагають аналізувати й виокремлювати корисну інформацію з великих обсягів геопросторових даних, доступних на платформі GEE.



Розділ 4

ФУНКЦІЇ В GOOGLE EARTH ENGINE

4.1. Структура функції

Однією з ключових особливостей GEE є можливість використання функцій для обробки та аналізу даних. Функції в GEE є важливою частиною платформи і дуже потужним інструментом, що дає змогу користувачам результативно використовувати й аналізувати великі обсяги геопросторових даних, підтримувати і створювати більш ефективний код, автоматизувати виконання стандартних операцій, які часто використовуються, як-от обчислення середнього значення даних або визначення вегетаційного індексу NDVI для серії растрових зображень.

Функції в програмуванні можна порівняти з функціями в математиці. Математична функція приймає вхідні дані (аргументи) і повертає результат. Наприклад, функція f(x) = x^2 приймає вхідний аргумент x і повертає квадрат цього аргументу. Однак функції в GEE працюють з геоданими, такими як супутникові знімки, дані дистанційного зондування і дані про землекористування.

Функції мають свої суворі і водночас гнучкі правила складання. У GEE вже існує власна бібліотека команд і функцій, також користувачі можуть ділитися своїми функціями з іншими користувачами для більшої зручності роботи. Ви самі можете створювати такі функції для використання в GEE.

Функція – це набір інструкцій, який може виконуватися при виклику. Функції можуть приймати аргументи (або вхідні дані), обробляти їх і повертати результат. Вони допомагають організовувати код у відокремлені блоки (або модулі), які можуть бути повторно використані під час розробки, що спрощує створення та підтримку програм.

Основою роботи в середовищі GEE є мова програмування JavaScript. Для кращого розуміння принципів створення і роботи функцій у Google Earth Engine рекомендуємо ознайомитися з основами, синтаксисом JavaScript, а також базовими принципами її використання в середовищі GEE за покликанням: https://developers.google.com/earth-engine/tutorials/tutorial_js_01.

Базова структура функції охоплює такі складові:

- *functionName* – ім'я функції, яке задає користувач-розробник; ім'я функції має бути унікальним і відображати призначення функції;

- **Function** – ключове слово, що вказує на початок оголошення функції;
- *argument1, argument2, ...* – параметри функції (аргументи), які приймаються як вхідні дані; можна вказати будь-яку кількість аргументів, розділяючи їх комами;
- *{}* – фігурні дужки визначають тіло функції, де вписано код, що містить набір інструкцій, які будуть виконуватися під час виклику функції;
- **return** – оператор return використовується для повернення результату роботи функції; результат може бути чим завгодно: значенням, об'єктом, зображенням, списком тощо;
- // – коментарі, які можуть бути додані для пояснення коду функції. Коментарі не впливають на роботу функції і використовуються для того, щоби пояснити код іншим розробникам або самому авторові.

Приклад відображення базової структури функції у реакторі коду GEE:

```
function functionName (argument1, argument2)
{


  return result;
}
```

```
// Оголошення функції та задання її імені й аргументів для отримання вхідних даних та виконання обчислень.
// Тіло функції – код, який виконується при виклику функції.
// Повернення результату функції
```

Функція – це така частина коду, яка виконується лише тоді, коли до неї звернутися (викликати). Для цього необхідно в скрипті записати її ім'я і задати відповідні аргументи всередині дужок, завершуючи оператор крапкою із комою. Результат роботи функції буде збережений у змінній або використаний якось інакше у скрипті.

```
functionName(argument1, argument2);
var result = functionName(argument1, argument2);
```

```
// Виклик оголошеної функції;
// Збереження оголошеної функції у змінну
```

Цікаво

Під час створення коду загальноприйнятою практикою є оголошення й визначення усіх функцій на початку скрипту з подальшим їх викликом. Це не обов'язково, але такий підхід робить код легшим для читання і розуміння.

Можливості використання функцій для обробки та аналізу даних у GEE не обмежуються створенням функції з базовою структурою. Для роботи зі специфічними об'єктами GEE (зображеннями (Image), колекціями зображень (ImageCollection), векторами (Feature), геометричними фігурами (Geometry), фільтрами (Filter) та ін.) можуть використовуватися **методи**.

Метод – це функція, збережена як властивість в об'єкті. Методи використовуються для зміни або отримання інформації з об'єктів і зазвичай прив'язані до конкретного об'єкта.

Методи в GEE – це функції, які викликаються на об'єктах, як-от зображення, колекції, геометричні об'єкти тощо. Вони використовуються для взаємодії з об'єктами і виконання операцій, які стосуються цих об'єктів. Наприклад, ee.Image.select() використовується для вибору певних каналів у зображенні або Map.addLayer() – для додавання певного шару у вікно карти.

Синтаксис методу в GEE стандартний і відповідає синтаксису багатьох об'єктно-орієнтованих мов програмування. Наведемо загальну структуру синтаксису методу:

- Object – об'єкт, до якого викликають метод;
- methodName – назва методу, який викликають;
- *argument1, argument2, ...* – аргументи, які передаються в метод для обробки.

object.methodName(*argument1, argument2*);	// Виклик методу;

Основна різниця між функціями і методами у GEE полягає в тому, що функції виконують глобальні операції або дії в скрипті, тоді як методи викликаються на об'єктах і взаємодіють з ними. В обох випадках це інструкції, які виконуються у вашому скрипті для обробки та аналізу геопросторових даних у GEE.

Крім того, у GEE існують **функції високого порядку**, зокрема reduce(), map(), evaluate(), filter() тощо, які можуть приймати інші функції як аргументи. Це робить код більш гнучким і дає змогу виконувати різні операції з обробки даних та аналізу.

На основі поєднання різних функцій та інших програмних конструкцій для вирішення певних геоінформаційних завдань, наприклад класифікації земельного покриття або аналізу часових рядів, у GEE можна створювати алгоритми.

Алгоритм – це послідовність кроків або операцій, які виконуються для розв'язання конкретної задачі або завдання. Алгоритми можуть охоплювати виклики функцій, операції обробки даних, умовні конструкції, цикли тощо.

Більш детально приклади використання різних функцій, методів і алгоритмів у GEE розглянемо у наступних підрозділах.

4.2. Бібліотеки вбудованих функцій GEE

Google Earth Engine має велику бібліотеку вбудованих функцій, які можна використовувати для маніпулювання геопросторовими даними. Залежно від операцій, які вони виконують, вбудовані функції можна розділити на кілька категорій, зокрема:

1. **Операції із зображеннями:**
- обчислення індексів вегетації, таких як NDVI та EVI;
- фільтрація, обрізка та маскування зображень;
- згладжування та фільтрація зображень.
2. **Робота з растрами та регіонами:**
- обчислення статистики (середнє, мінімальне, максимальне значення тощо) для регіонів;
- класифікація растрів і аналіз земного покриву.
3. **Робота з векторними даними:**
- вибір об'єктів на основі умови;
- обчислення властивостей об'єктів.
4. **Часовий аналіз:**
- обчислення середніх значень для даних на різні дати;
- агрегація часових рядів даних.
5. **Аналіз супутникових знімків:**
- обробка та аналіз даних супутникових систем, як-от Landsat, Sentinel тощо;
- визначення областей зміни покриття Землі.
6. **Робота з числами та масивами даних:**
- математичні операції, статистика, фільтрація масивів чисел.
7. **Візуалізація та виведення даних:**
- візуалізація зображень, растрових і векторних даних на мапі;
- створення графіків і діаграм для аналізу даних.
8. **Геометричні операції та обробка геоданих:**
- обчислення площі, довжини, відстаней між точками тощо;
- обробка геометричних форм та геоданих.

Кожна категорія містить багато функцій, методів та алгоритмів, які можна використовувати для обробки та аналізу геоданих на платформі GEE. До-

кладну інформацію про них (призначення, синтаксис, аргументи тощо) можна знайти в офіційній документації за покликанням: https://developers.google.com/earth-engine/guides або безпосередньо у вкладці ***Docs*** редактора коду GEE (https://code.earthengine.google.com/) (див. *рис. 23*).

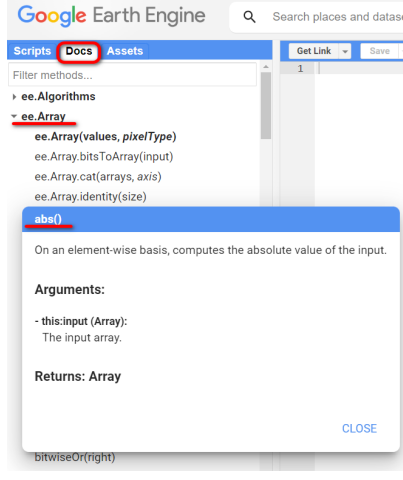


Рис. 23. Перегляд інформації про одну із вбудованих функцій GEE – *abs()*

Для використання вбудованих функцій та методів у GEE необхідно написати код, який викликатиме їх *у* вашому скрипті. Наприклад:

```
var collection =
ee.ImageCollection('LANDSAT/LC08/C01/
T1_TOA')
  .filterBounds(ee.Geometry.Point(30.5238,
50.4501))
  .filterDate('2020-01-01', '2020-12-31');

var image = collection.first();


var ndvi = image.normalizedDifference(['B5',
'B4']);
var ndviVis = {min: -1, max: 1, palette: ['red',
'yellow', 'green']};

Map.addLayer(ndvi, ndviVis, 'NDVI Image');

Map.setCenter(30.5238, 50.4501, 10);


print('NDVI значення для зображення:',
ndvi);
```

```
// Імпорт колекцій даних
Landsat
// Київ (довгота, широта)
// Встановлення бажаного
часового періоду
// Вибір одного зображення з
колекції (наприклад, перше
доступне)
// Виклик вбудованих функцій
та методів:
// Обчислення NDVI
// Визначення параметрів візуа-
лізації NDVI


// Додавання зображення на
мапу
// Встановлення центру карти у
Києві (за допомогою коорди-
нат) та її масштабування
// Виведення значення NDVI у
консоль
```

Цей код демонструє приклади використання вбудованих функцій і методів, як-от: normalizedDifference(), Map.addLayer(), Map.setCenter(), first() та print(). Виконання коду дає змогу створити карту значень NDVI для Києва на основі одного із супутникових зображень Landsat і візуалізувати отримані результати (див. *рис. 24*).

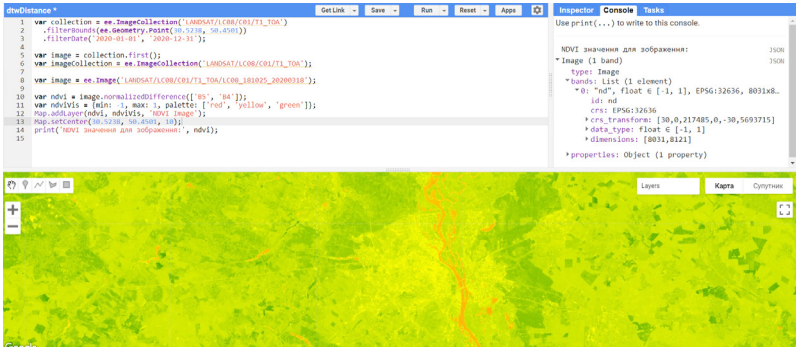


Рис. 24. Результат застосування скрипту із вбудованими функціями GEE

Також існує багато різних зовнішніх бібліотек функцій, які створені спільнотою користувачів GEE. Ці бібліотеки містять функції для обробки та аналізу геопросторових даних для конкретних завдань. Наприклад, деякі популярні бібліотеки містять інструменти для роботи з Landsat, Sentinel, моделями рослинного покриву тощо.

Імпорт таких бібліотек дає змогу використовувати готові функції і методи, що спрощує аналіз та обробку даних у GEE. Однак пам'ятайте, що перед використанням бібліотеки варто ознайомитися з документацією і прикладами коду, які надаються разом з бібліотекою, щоб зрозуміти, як користуватися її функціями. Можна знайти опис і приклади застосування таких бібліотек в офіційній документації за покликанням https://developers.google.com/earth-engine/ і в різноманітних блогах та форумах спільноти користувачів GEE.

Для використання сторонніх бібліотек спочатку потрібно імпортувати їх у скрипт за допомогою ключового слова require. Для вбудованих функцій цього не потрібно, оскільки вони вже доступні в GEE.

Розглянемо приклад імпорту й використання функцій зовнішньої бібліотеки **OpenEarthEngineLibrary**, документацію якої можна знайти за покликанням: https://www.open-geocomputing.org/OpenEarthEngineLibrary/#.

Щоб скористатися цією бібліотекою, користувачеві просто необхідно виконати її імпорт у свій код, а потім скористатися доступними функціями. Для отримання переліку використаних функцій бібліотеки і супутньої інформації про них у кінці коду слід використати метод *oeel.refs()*.

```
var oeel=require('users/OEEL/lib:loadAllSF');
print('List of functions used',oeel.refs());
```

```
// Імпорт бібліотеки OpenEarthEngineLibrary
// Виведення списку використаних функцій
```

Нижче наведено приклад використання функції цієї бібліотеки *oeel.Algorithms.Sentinel2.cloudfree*(maxCloud, S2Collection), яка дає змогу створити колекцію безхмарних зображень на основі даних, отриманих за допомогою супутника Sentinel-2 (див. *рис. 25–26*).

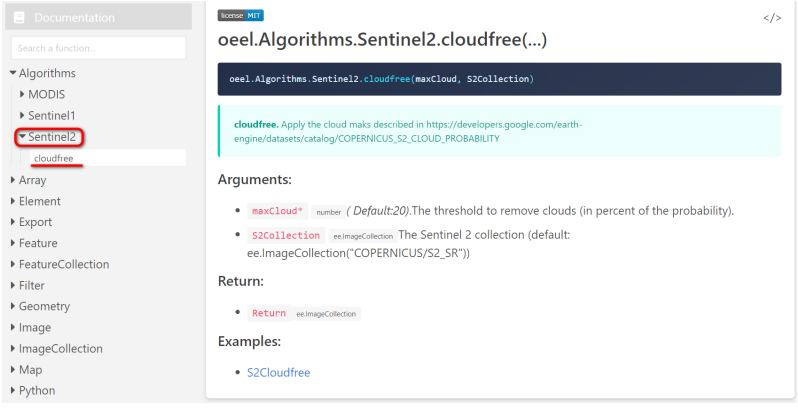


Рис. 25. Перегляд доступної інформації про функцію *oeel.Algorithms.Sentinel2.cloudfree* на сайті https://www.open-geocomputing.org

var oeel=require('users/OEEL/lib:loadAll'); Map.centerObject(**ee.Geometry.Point**([-30.2988129359949543, 50.759976124941396]),10)	// Імпорт бібліотеки // Центрування карти навколо точки із заданими координатами і масштабуванням
var S2Cloudfree=oeel.Algorithms.Sentinel2.cloudfree()	// Створення змінної S2Cloudfree, яка представляє собою результат виклику функції oeel.Algorithms.Sentinel2.cloudfree()
Map.addLayer(S2Cloudfree.filterBounds(Map.getBounds(**true**)) .filter(ee.Filter.calendarRange(170, 220)) .median(),{bands:['B4','B3','B2'],min:0, max:3000})	// Додати на карту зображення, яке оброблене функцією S2Cloudfree, відфільтрувавши його за зоною видимості карти, діапазоном календарних днів і медіанним значенням
print('list of functions used',oeel.refs())	// Відображення списку використаних функцій бібліотеки у консолі

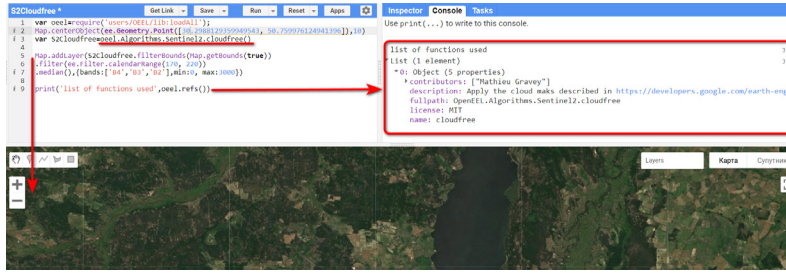


Рис. 26. Результат застосування функції *oeel.Algorithms.Sentinel2.cloudfree*

Загалом бібліотеки вбудованих функцій у GEE роблять платформу потужним інструментом для виконання різноманітних завдань, пов'язаних із геоспросторовою аналітикою та дослідженнями. Також користувачі можуть доповнювати бібліотеки вбудованих функцій GEE або розробляти власні бібліотеки для виконання специфічних завдань або для організації коду в більш зручний спосіб.

4.3. Створення функцій

Як уже було зазначено, користувачі в GEE можуть створювати за допомогою мови програмування JavaScript власні функції, які виконують конкретні специфічні завдання, недоступні в бібліотеках, що вже існують. Це дає змогу користувачам розширювати можливості платформи, адаптувати її під вирішення власних завдань.

Розглянемо кілька простих прикладів створення функцій у середовищі GEE (див. *рис. 27–30*).

Обчислення суми двох значень:

```
var addNumbers = function(a, b) {
  return a + b;
};

var result = addNumbers(5, 3);
print("Результат додавання:", result);
```

```
// Функція для обчислення суми двох значень


// Виклик функції
```

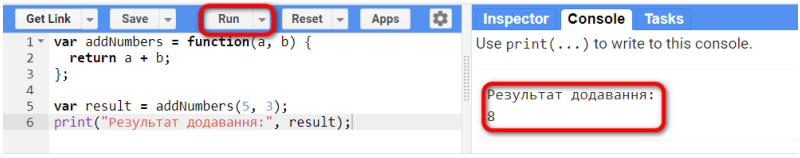


Рис. 27. Результат застосування функції *addNumbers*

Обчислення площі прямокутника:

```
var calculateRectangleArea =
function(length, width) {
  return length * width;
};
var area = calculateRectangleArea(10, 8);
print("Площа прямокутника:", area);
```

```
// Функція для обчислення площі прямокутника

// Виклик функції
```

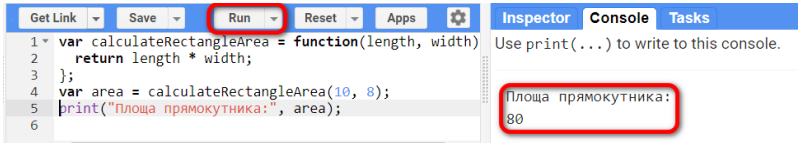


Рис. 28. Результат застосування функції *calculateRectangleArea*

Визначення найменшого значення у списку:

```
var findMinValue = function(numbers) {
  var minValue = numbers[0];
  for (var i = 1; i < numbers.length; i++) {
    if (numbers[i] < minValue) {
      minValue = numbers[i];
    }
  }
  return minValue;
};

var numbersList = [10, 5, 20, 8, 15];
var min = findMinValue(numbersList);
print("Найменше значення:", min);
```

```
// Функція для знаходження найменшого значення у списку

// Виклик функції
```

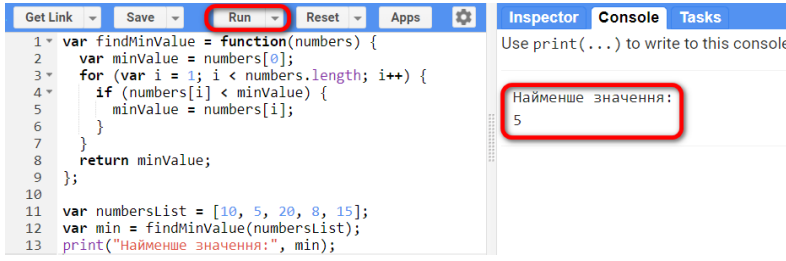


Рис. 29. Результат застосування функції *findMinValue*

Обчислення середнього значення NDVI для знімка:

```
var calculateAverageNDVI =
function(image) {
  var ndvi = image.
normalizedDifference(['B5', 'B4']);
  var avg = ndvi.reduceRegion({
              reducer: ee.Reducer.mean(),
              geometry: image.geometry(),
              bestEffort: true,
              scale: 30
  });
  return avg.get('nd');
};
```

```
// Оголошення функції для обчислення середнього NDVI
```

```
var landsatImage =
ee.Image('LANDSAT/LC08/C01/T1_TOA/
LC08_123456789');
var averageNDVI = calculateAverageND
VI(landsatImage);
print("Середній NDVI:", averageNDVI);
```

```
// Завантаження зображення Landsat

// Після того як функцію оголошено, ви можете викликати її, передавши необхідні аргументи
```

Рис. 30. Результат застосування функції *calculateAverageNDVI*

В останньому прикладі ми створили функцію *calculateAverageNDVI*, яка приймає зображення як параметр (аргумент). У тілі функції ми обчислюємо NDVI для зображення, розраховуємо середнє значення за допомогою методу reduceRegion і повертаємо результат.

Цікаво

У GEE ви можете використовувати як локальні, так і глобальні змінні у своїх функціях. Локальні змінні – це змінні, які оголошуються і використовуються в межах конкретної функції. Вони мають обмежений обсяг видимості і доступні лише всередині функції, де вони були оголошені. Це дає змогу ізолювати дані й обчислення від інших частин коду. Глобальні змінні – це змінні, які оголошуються на верхньому рівні вашого скрипту або програми і доступні для всіх функцій і частин коду в межах цього скрипту. Вони мають більший обсяг видимості і можуть використовуватися для передачі даних між функціями. Зверніть увагу, що глобальні змінні можуть бути корисними для передачі об'єктів геометрії, регіонів чи інших даних між функціями. Однак важливо бути обережними, оскільки використання глобальних змінних може призвести до неочікуваних результатів, якщо вони змінюються в різних частинах коду.

Для кращого розуміння відмінностей між локальними і глобальними змінними розглянемо простий приклад їх використання (див. *рис. 31*):

```
function calculate_Sum (in_value1,
in_value2) {
var Sum = ee.Number(in_value1).add(ee.
Number(in_value2));
            print("Локальна змінна, Sum, = ",
Sum);
            return Sum;
}
var Sum_test = calculate_Sum(75, 82);

print("Глобальна змінна, Sum_test, =
",Sum_test);
```

```
// Оголошення функції обчис-
лення суми

// Створення локальної змінної
Sum і присвоєння їй результату
функції

// Виведення значення локаль-
ної змінної в консоль

// Створення глобальної змінної
Sum_test і присвоєння їй
результату функції
// Виведення значення глобаль-
ної змінної в консоль
```

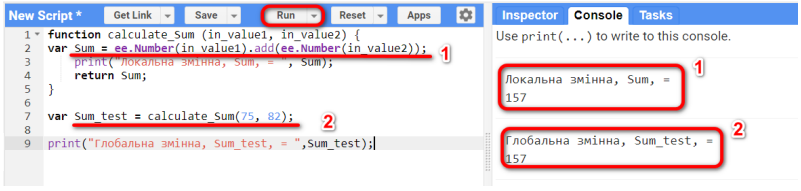


Рис. 31. Приклад застосування локальної і глобальної змінних

Створення функцій у GEE є доволі гнучким процесом. Вибір підходу до створення функції залежить від конкретних потреб робочого процесу і стилю програмування розробника. Важливо розуміти, що немає єдиного правильного способу створення функцій. Головне – це зробити код читабельним і ефективним для автора та інших розробників і користувачів.

4.4. Застосування функцій до колекцій даних

У GEE функції часто застосовуються до колекцій даних, як-от зображення або векторні дані, для виконання обчислень або операцій на всіх елементах цієї колекції. Це дає змогу ефективно обробляти й аналізувати великі масиви геоданих.

Загальний алгоритм для застосування функцій до колекцій даних у GEE:

1. Визначення функції: спочатку потрібно визначити функцію, яку буде застосовано до кожного елемента колекції. Функція повинна прийма-

ти елемент колекції (наприклад, зображення) як вхідний параметр і повертати результат.

2. Застосування функції до колекції: тут доцільно використовувати методи колекцій GEE (чи функції високого порядку), такі як map(), filter(), reduce(), щоб застосувати функцію до всіх елементів колекції. Найчастіше використовується метод map(), який застосовує функцію до кожного елемента колекції і повертає нову колекцію з результатами.
3. Фільтрація даних (за потреби): якщо потрібно, можна застосувати фільтрацію до колекції перед або після застосування функції. Наприклад, можна використовувати метод filter() для вибору певних даних за критеріями.
4. Виведення результатів або аналіз: залежно від задачі можна вивести результати або виконати подальший аналіз даних.

GEE також підтримує функції високого порядку, такі як map(), reduce(), filter(), які можуть приймати інші функції як параметри і можуть бути корисними для роботи з різними колекціями геоданих.

Основні функції високого порядку в GEE:

- map(): дає змогу застосовувати іншу функцію до кожного елемента колекції даних (наприклад, зображення) та створювати нову колекцію з результатами;
- reduce(): уможливлює зведення даних у колекції до одного значення з використанням іншої функції. Наприклад, можна використовувати reduce() для обчислення середнього значення або інших статистик для всіх елементів колекції;
- filter(): дає змогу фільтрувати елементи колекції, використовуючи іншу функцію, яка має повертати логічний результат (true або false). Можна використовувати filter() для вибору зображень, які відповідають певним критеріям.

Розглянемо декілька типових прикладів застосування функцій до колекцій даних.

Щоб застосувати функцію до кожного зображення в колекції ImageCollection, використовується imageCollection.map(). Єдиним аргументом map() є функція, яка приймає один параметр: ee.Image. Наприклад, наведений нижче код додає канал із міткою часу до кожного зображення колекції (див. *рис. 32*).

```
var collection =                                    // Завантаження колекції Landsat 8
ee.ImageCollection('LANDSAT/LC08/                   (тільки знімки 2021 року)
C02/T1_TOA')
  .filterDate('2021', '2022')
  .filter(ee.Filter.eq('WRS_PATH', 44))
  .filter(ee.Filter.eq('WRS_ROW', 34));
var addTime = function(image) {                     // Ця функція додає канал, що
  return image.addBands(image.                      відображає мітку часу отримання
getNumber('system:time_start'));                    зображення
};

print(collection.map(addTime));                     / Відобразити функцію на колекцію і
                                                    вивести результат у консоль
```

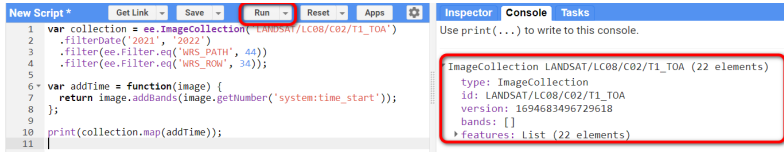


Рис. 32. Результат застосування функції *collection.map(addTime)*

Зверніть увагу, що в попередньо визначеній функції метод getNumber() використовується для створення нового зображення із числовим значенням властивості часу, коли було зафіксоване це зображення. Зазвичай це числове значення, що представляє час у мілісекундах із 1 січня 1970 р. (згідно з епохою UNIX). Наявність відомостей про часовий діапазон корисна для моделювання змін і створення композитів.

Створена функція *collection.map(addTime)* дещо обмежена в операціях, які вона може виконувати. Зокрема, вона не може змінювати глобальні змінні, виводити інформацію на друк, використовувати оператори JavaScript '*if*' або '*for*'. Однак для розширення можливостей функції можна додатково використовувати ee.Algorithms.If() для виконання умовних операцій. Наприклад, створимо за допомогою цього алгоритму функцію, яка повертатиме зображення, отримані при висоті Сонця над горизонтом 40 градусів і вище, а в іншому випадку (висота Сонця до 40 градусів) – виводитиме порожнє (нульове) зображення (див. *рис. 33*):

```
var collection =
ee.ImageCollection('LANDSAT/LC08/
C02/T1_TOA')
  .filterDate('2021', '2022')
  .filter(ee.Filter.eq('WRS_PATH', 44))
  .filter(ee.Filter.eq('WRS_ROW', 34));

var conditional = function(image) {
  return ee.Algorithms.If(ee.
Number(image.get('SUN_
ELEVATION')).gt(40),
                    image,
                    ee.Image(0));
};

print('Розгорніть, щоб побачити ре-
зультати', collection.map(conditional));
```

```
// Завантажити колекцію Landsat 8 –
тільки зображення 2021 року

// Ця функція використовує умовний
оператор для повернення зображен-
ня, якщо висота Сонця > 40 градусів.
В іншому випадку повертається
«нульове зображення».

// Відобразити функцію на колекцію і
вивести результат
```

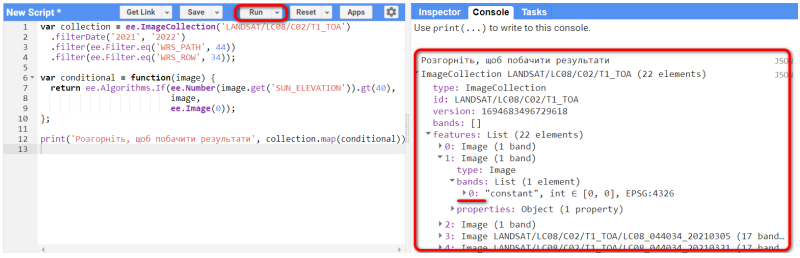


Рис. 33. Результат застосування функції *collection.map(conditional)*

Перегляньте список зображень у результуючій колекції ImageCollection і зверніть увагу, що 7 із 22 зображень виведені як «нульові зображення». Тобто, якщо умова, яка обчислюється алгоритмом If(), є істинною, результат містить зображення з початкової колекції, а якщо хибною – властивості bands присвоюється значення 0.

Важливо! *Хоча цей приклад демонструє можливості створення умовної функції на стороні сервера, уникайте використання If() взагалі і використовуйте замість неї фільтри для вищої продуктивності коду.*

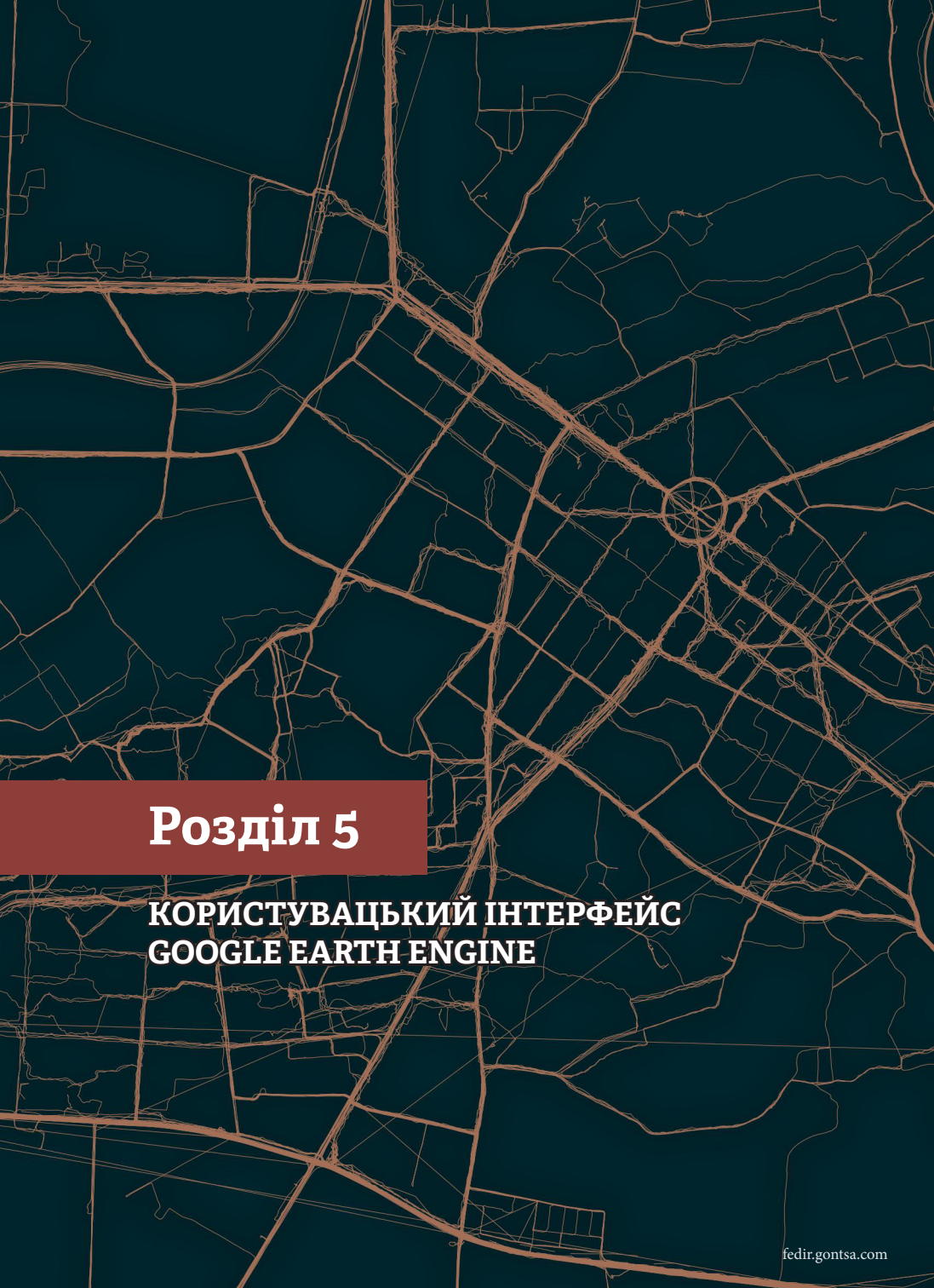
Розглянемо ще один приклад застосування функцій для обробки колекцій векторних даних:

```
var vectorCollection =                                  // Векторна колекція даних
ee.FeatureCollection('FAO/
GAUL/2015/level1');
var calculateTotalArea =                                // Функція для обчислення суми площ
function(featureCollection) {                           об'єктів у векторній колекції
  var reduceArea = ee.Reducer.sum().                    // Функція агрегації для обчислення
setOutputs(['total_area']);                             суми площ об'єктів

    var totalArea = featureCollection.                  // Застосування функції агрегації
reduceColumns({                                         reduceArea до векторної колекції
          reducer: reduceArea,
          selectors: ['Shape_Area'],                    // Вибір колонки таблиці з площею
  });                                                   для обчислень

  var totalAreaValue =                                  // Отримання суми площі усіх об'єктів
ee.Number(totalArea.get('total_area'));

  print('Загальна площа усіх об'єктів:',                // Виведення результату в консоль
totalAreaValue, 'кв. метрів');
};
calculateTotalArea(vectorCollection);                   // Виклик функції для обчислення суми
                                                        площі усіх об'єктів у векторній колекції
```

У цьому прикладі функція *calculateTotalArea()* обчислює суму площі всіх об'єктів у векторній колекції FAO/GAUL/2015/level1 *(FAO GAUL: Global Administrative Unit Layers 2015, First-Level Administrative Units)*. Функція *reduceArea* використовується для агрегації даних площі об'єктів у векторній колекції. Результат – сумарна площа – виводиться у консоль (див. *рис. 34*).

Рис. 34. Результат застосування функції *calculateTotalArea()*

Отже, функції високого порядку дають змогу зручно й ефективно обробляти та аналізувати колекції растрових і векторних даних у GEE.



Розділ 5

КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС GOOGLE EARTH ENGINE

5.1. Загальна характеристика модуля користувацького інтерфейсу

Користувацьким інтерфейсом послуговуються для створення окремих застосунків, у яких користувач буде взаємодіяти з інструментами для роботи з геоданими. На базі GEE можна створювати окремі програми для аналізу даних, відображення інформації для використання поза межами редактора коду.

Користувацький інтерфейс (UI) в GEE – це набір інструментів та об'єктів, які допомагають створювати інтерактивні інтерфейси для взаємодії користувача з геопросторовими даними. За допомогою UI ви можете створювати візуальні елементи, як-от кнопки, панелі, текстові поля та ін., для введення параметрів, відображення результатів, налаштування візуалізації тощо.

Основні переваги використання користувацького інтерфейсу (UI) у GEE:

- ***зручність:*** UI надає простий та зрозумілий інтерфейс, який дає змогу користувачам легко здійснювати операції з даними й аналізувати їх, навіть без попереднього досвіду роботи з програмуванням;
- ***візуалізація:*** UI дає змогу візуалізувати геопросторові дані в реальному часі за допомогою карт та графіків, що полегшує сприйняття інформації і забезпечує краще розуміння даних;
- ***інтерактивність:*** UI сприяє взаємодії користувачів із даними та їх аналізу в режимі реального часу, що дає змогу швидше робити висновки і приймати рішення;
- ***швидкість і продуктивність:*** використання UI не вимагає завантаження або встановлення додаткового програмного забезпечення, тому процес аналізу даних є ефективним та швидким;
- ***легка інтеграція:*** UI може бути легко інтегрований з іншими вебзастосунками і сервісами, що допомагає створювати потужні та комплексні рішення для геопросторового аналізу.

Загалом користувацький інтерфейс GEE дає змогу зробити платформу доступною та зрозумілою для широкого кола користувачів, включаючи тих, хто не має досвіду програмування, що робить її потужним інструментом для вивчення й аналізу геопросторових явищ.

GEE API надає різні компоненти в модулі UI для створення користувацького інтерфейсу. Ці компоненти часто називають віджетами.

Віджети – це невеликі графічні програми, метою яких є надання візуальної інформації та полегшення доступу до функцій, які часто використовуються.

У модулі UI GEE можна виділити три основні функціональні групи компонентів:

- віджети, які дають змогу користувачам відображати інформацію, наприклад: **ui.Label**, **ui.Chart**, **ui.Thumbnail**;
- віджети, які дають змогу користувачам задавати певні параметри або взаємодіяти з кодом, наприклад: **ui.Button**, **ui.CheckBox**, **ui.DateSlider**, **ui.Select**, **ui.Textbox**;
- віджети, які дають змогу групувати й організовувати інші елементи інтерфейсу для зручної роботи користувачів, наприклад: **ui.Map**, **ui.Panel**, **ui.SplitPanel**.

Поєднання віджетів різного типу в GEE дає змогу створювати інтерактивні інтерфейси різної складності, за допомогою яких користувач може виконувати пошук геопросторових даних, а також взаємодіяти з платформою, використовуючи інтерактивний JavaScript-код: налаштовувати різні параметри, проводити аналіз, створювати і візуалізувати моделі тощо для взаємодії користувачів із геопросторовими даними, налаштування параметрів, виконання аналітичних операцій та відображення їх результатів.

5.2. Основні компоненти модуля користувацького інтерфейсу

Докладну інформацію, синтаксис та приклади використання всіх доступних компонентів модуля UI можна знайти в документації GEE API (https://developers.google.com/earth-engine/apidocs/). Тут ми ознайомимося з найбільш типовими компонентами модуля UI, які можна зустріти практично в будь-якому вебзастосунку, створеному на основі Google Earth Engine API.

Віджет **ui.Label** дає змогу створювати текстові мітки і написи для відображення допоміжної інформації або заголовків в інтерфейсі.

Для налаштування зовнішнього вигляду текстових міток (написів) використовуються такі властивості об'єкта **ui.Label**, як *value* і *style*.

За допомогою властивості *value* створюють текстовий зміст напису.

Властивість *style* визначає стилізацію напису і містить, своєю чергою, різні параметри (атрибути) для налаштування зовнішнього вигляду, напри-

клад: розташування *(position),* розмір шрифту *(fontSize),* жирність шрифту *(fontWeight),* колір фону *(backgroundColor),* внутрішній відступ *(padding),* рамка *(border)* та багато ін.

Розглянемо приклад створення напису і додавання його на карту в GEE (див. *рис. 35)*.

```
var point = ee.Geometry.Point( 30.5, 50.5);                  // Створити локалізацію на карті для розташування напису

var label = ui.Label({                                       // Створити напис
  value: 'Геопросторові дані›,
  style: {                                                   // Налаштувати зовнішній вигляд напису
    position: 'top-center',
    fontSize: '20px',
    fontWeight: 'bold',
    backgroundColor: 'white',
    padding: '5px',
    border: '1px solid black'
  }
});

Map.add(label);                                              // Додати напис на карту

Map.centerObject(point, 10);                                 // Змінити розташування центру карти так, щоб було видно напис
```

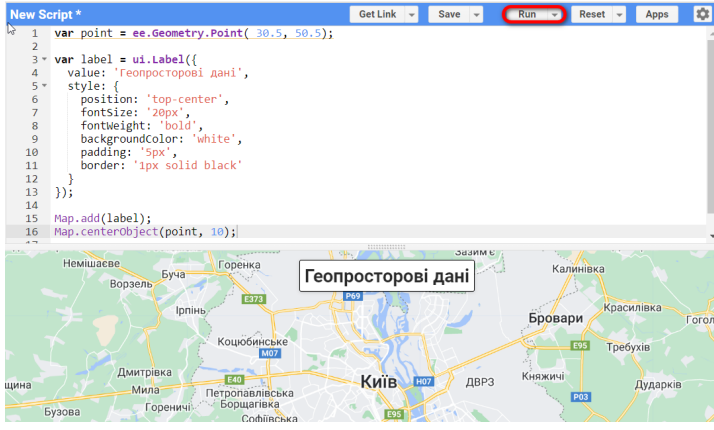


Рис. 35. Приклад створення напису в GEE

Віджет **ui.Chart** дає змогу створювати й відображати графіки та діаграми різного типу (лінійні, стовпчасті, кругові тощо) для візуалізації статистичних і числових даних. За замовчуванням графіки є інтерактивними: користувачеві достатньо навести курсор на точки, лінії, смуги тощо, щоби побачити відповідні значення. Додатково зовнішній вигляд графіків можна налаштувати за допомогою методу ui.Chart.setOptions(). Докладну інформацію щодо налаштування властивостей стилю діаграми можна знайти за покликанням: https://developers.google.com/earth-engine/guides/charts_style.

Приклад використання **ui.Chart** об'єкта для створення графіка динаміки значень індексу NDVI протягом 2022 р. для заданої точки на карті (див. *рис. 36*):

```
var data = ee.ImageCollection('MODIS/061/MOD13A2')
    .filterDate('2022-01-01', '2022-12-31')
    .select('NDVI');

var chart = ui.Chart.image.series({
  imageCollection: data,
  region: ee.Geometry.Point(30.3, 50.5),
  reducer: ee.Reducer.mean(),
  scale: 50
});

print(chart)
```

```
// Завантажити колекцію даних MODIS для NDVI і вибрати дані за 2022 рік

// Створення графіка

// Вивести графік у консоль
```

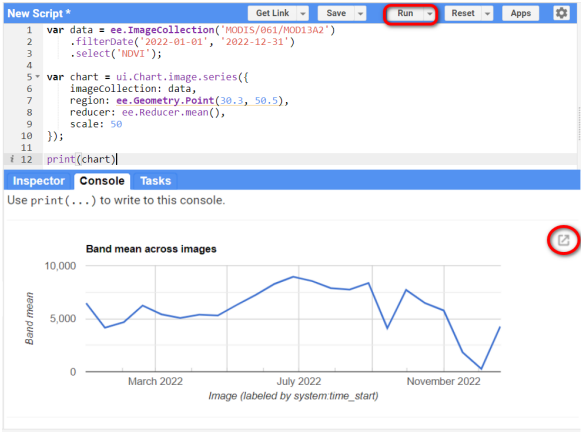


Рис. 36. Приклад створення **ui.Chart** у GEE

ui.Chart віджети можна відображати трьома способами: в консолі редактора коду, у віджеті-контейнері ui.Panel та в окремій вкладці браузера. Останній спосіб можна реалізувати, клацнувши на піктограму, що випливає, (open_in_new) у верхньому правому куті відображеного **ui.Chart** об'єкта. Нова сторінка забезпечує відображення створеного графіка чи діаграми в окремому вікні і надає опції для завантаження діаграми у вигляді графічного файлу (PNG або SVG) або файлу CSV з основними даними, використаними для побудови.

Віджет **ui.Thumbnail** можна використовувати для попереднього перегляду об'єктів **ee.Image** та **ee.ImageCollection**. Якщо створити цей віджет на основі ee.Image, він відображатиме статичне зображення; якщо використати **ee.ImageCollection**, буде створено анімацію, де кожен кадр відповідатиме зображенню в колекції вхідних даних. Приклад створення такої карти-мініатюри на основі знімка супутника Landsat для території м. Києва у травні 2021 р. (див. *рис. 37*):

```
var box = ee.Geometry.Polygon([                          // Створення області інтересу навколо міста Київ
  [30.3605, 50.3714],
  [30.5252, 50.3714],
  [30.5252, 50.5054],
  [30.3605, 50.5054]
]);

var image = ee.Image('LANDSAT/LE07/C01/T1_SR/LE07_181025_20210516').visualize({   // Візуалізувати зображення у режимі RGB
  bands: ['B3', 'B2', 'B1'],
  min: 0,
  max: 1200,
  gamma: [1.3]
});

var thumbnail = ui.Thumbnail({                           // Створити мініатюру із вказаними розмірами для області інтересу
  image: image,
  params: {
    dimensions: '256x256',
    region: box,
    format: 'png'
  }
});

thumbnail.style().set({                                  // Налаштувати відображення віджета
  height: '300px',
  width: '300px'
});

print(thumbnail);                                        // Вивести мініатюру в консоль
```

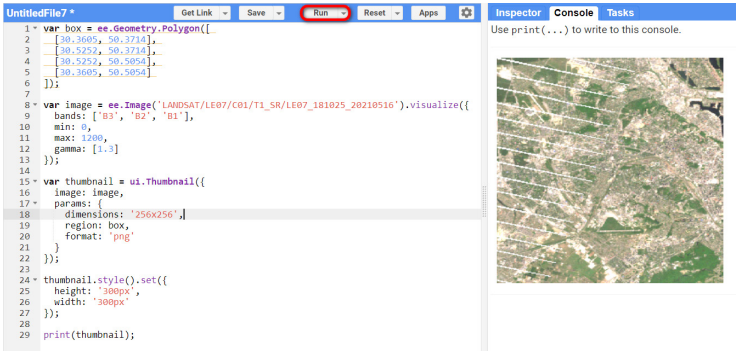


Рис. 37. Приклад створення мініатюри **ui.Thumbnail** у GEE

Віджет **ui.Button** являє собою кнопку, на яку користувач може натиснути для виконання певних дій. Для того щоб створена кнопка стала інтерактивним елементом інтерфейсу, потрібно визначити обробник подій *onClick*, який буде виконувати певні завдання при натисканні на неї. Наприклад (див. *рис. 38*):

```
var button = ui.Button('Визначити координати центру карти');          // Створити кнопку

button.onClick (function() {                                           // Задати функцію визначення
   print(Map.getCenter());                                             // координат поточного виду карти
});

print(button);                                                         // Відобразити кнопку в консолі
```

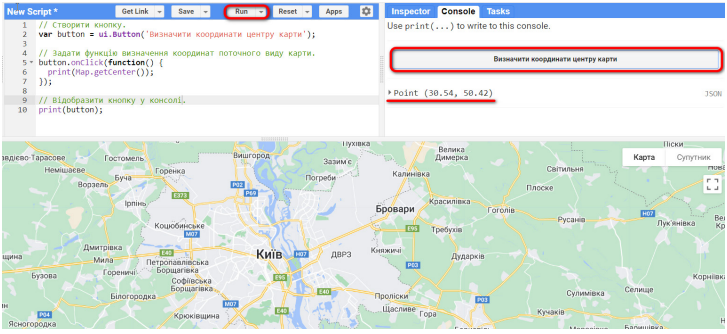


Рис. 38. Приклад створення кнопки в GEE

Віджет **ui.Textbox** виводить поле введення тексту, де користувач може вносити значення. Цей компонент може використовуватися для збору вхідних даних від користувача. Створюючи віджети такого типу, додатково можна задавати атрибут *placeholder*, який відображається в текстовому полі як пояснення для користувача. Також, як правило, використовується обробник подій *onChange*, який виконує указані дії при зміні значення текстового поля. Можна додатково налаштовувати **ui.Textbox**, встановлюючи початкове значення за допомогою *setValue()*, або отримувати значення з віджета за допомогою *getValue()*, задавати обмеження на введення даних і багато ін. Наприклад, так за допомогою віджета **ui.Textbox** і обробника подій можна вивести введений користувачем текст у консоль за допомогою команди *print* (див. *рис. 39*):

```
var textbox = ui.Textbox({                    // Створення поля введення даних
  placeholder: 'Введіть ім`я та
прізвище користувача',
  onChange: function(text) {                  // Задати функцію відображення введе-
    print(text);                              // них користувачем даних у консоль
    textbox.setValue('');                     // Очистити поле введення даних
  }
});
print(textbox);                               // Відобразити створене поле в консолі
```

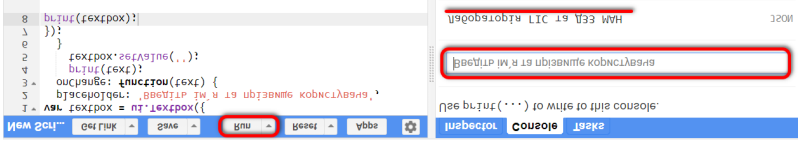


Рис. 39. Приклад створення поля введення тексту в GEE

Віджет **ui.Select** дає змогу створити випадне меню, де користувач може вибрати один елемент зі списку різних параметрів або категорій. Для налаштування цього компонента, як правило, використовуються властивості *items* (масив рядків, які представляють доступні опції у розкривному списку) і *placeholder* (текст, який відображається як підказка, коли користувач ще не вибрав жодної опції), а також обробник подій *onChange*, в якому вказується функція, що виконується при зміні вибраної опції. У наступному прикладі показано створення меню, яке дає змогу користувачеві вибрати розташування (див. *рис. 40*).

```
var places = {
  Київ: [30.3, 50.3],
  Львів: [24.0, 49.5],
  Харків: [36.1, 50.0]
};

var select = ui.Select({
  items: Object.keys(places),

onChange: function(key) {

Map.setCenter(places[key][0],
places[key][1]);
  }
});

select.setPlaceholder('Виберіть
локацію');

print(select);
```

```
// Створення об'єкта places, який містить
координати різних міст

// Створення віджета
// Опції вибору генеруються на основі
ключів об'єкта places
// Задати функцію відображення введе-
них користувачем даних у консоль
// При виборі опції змінюємо центр карти
(Map) на координати обраної локалізації

// Створення тексту-підказки для корис-
тувача

// Відобразити створене меню в консолі
```

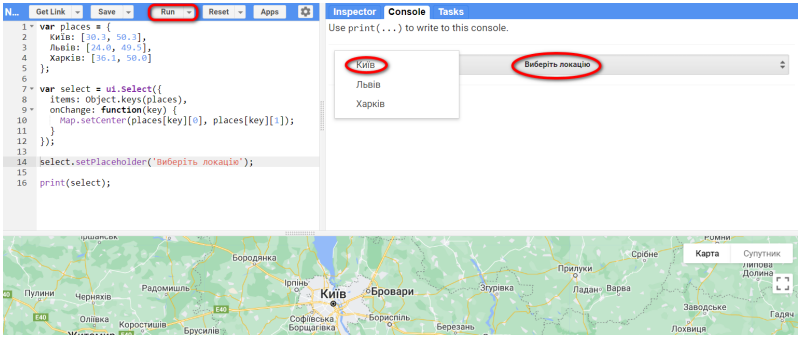


Рис. 40. Приклад створення випадного меню в GEE

ui.CheckBox – це компонент, який використовується для створення прапорців (чекбоксів) в інтерфейсі користувача для керування параметрами та об'єктами на карті. Наприклад, він дає змогу вмикати або вимикати відображення різних шарів даних на карті GEE. Виконання певних дії залежно від того, чекбокс увімкнено чи вимкнено, здійснюється за допомогою обробника подій *onChange*. Приклад створення подібного елемента (див. *рис. 41*):

```
var checkbox = ui.Checkbox({                    // Створення чекбоксу
  label: 'Показати шар даних',
  value: false                                  // Початковий стан: вимкнуто
});
checkbox.onChange(function(value) {             // Дія при зміні стану чекбоксу
 if (value) {                                   // Оперативний код, який виконуєть-
    } else {                                    // ся при зміні стану чекбоксу
    }
});
Map.add(checkbox);                              // Додавання чекбоксу до карти
```

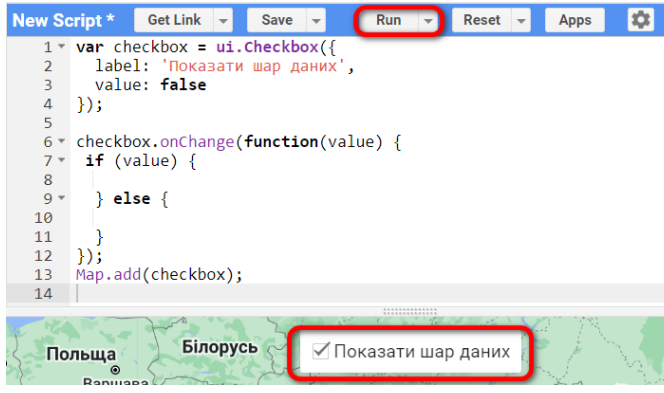


Рис. 41. Приклад створення чекбоксу в GEE

ui.DateSlider – це елемент інтерфейсу GEE, який являє собою слайдер для вибору дати або діапазону дат. Він надає можливість користувачам визначати дату або діапазон дат для обмеження аналізу або візуалізації геопросторових даних. **ui.DateSlider** дає змогу визначити початкову й кінцеву дату, мінімальну і максимальну доступні дати, а також налаштувати крок між датами на слайдері. Користувачі можуть переміщувати повзунок по слайдеру, щоб вибрати конкретну дату або діапазон дат. Наприклад, ось так можна створити **ui.DateSlider** і використати його в інтерфейсі (див. *рис. 42*):

```
var dateSlider = ui.DateSlider({
  start: '2010-01-01',
  end: '2022-12-31',
  style: {width: '200px'},
  period: 1,

  onChange: function(date) {

            print('Обрана дата: ', date);
  }
});
Map.add(dateSlider);
```

```
// Створення ui.DateSlider
// Початкова дата
// Кінцева дата
// Задання розмірів (ширини) об'єкта
// Крок між датами (дні)

// Функція, що виконується при зміні значення слайдера – виведення обраної дати в консоль

// Додавання ui.DateSlider до вікна карти
```

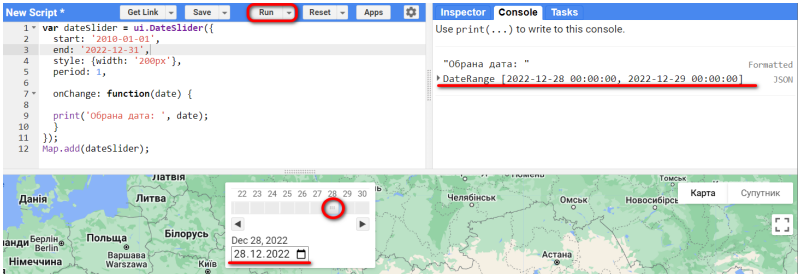


Рис. 42. Приклад створення **ui.DateSlider**

Користувач може застосовувати **ui.DateSlider** для обмеження аналізу даних до конкретного періоду часу, налаштовувати візуалізацію на основі вибраної дати чи періоду часу. Цей віджет ви можете використовувати окремо або разом з іншими компонентами UI, щоб створювати більш складні інтерактивні інтерфейси для роботи з геопросторовими даними.

Об'єкт **ui.Map** є візуальним контейнером, який відображає карти і дає змогу взаємодіяти з геопросторовими даними. Треба відзначити, що платформа GEE має два класи об'єктів, які можуть бути використані для відображення геопросторових даних на карті: **ui.Map** та **ee.Map**. **ee.Map** використовується для створення статичних або інтерактивних мап на основі даних із GEE, аналізу растрових і векторних шарів, обчислення та відображення результатів обробки даних. Тоді як **ui.Map** забезпечує можливість взаємодії з користувачем та відображення інтерактивних мап на вебсторінці GEE. За допомогою елементу **ui.Map** можна не лише створювати карти, а й додавати шари, керувати виглядом карт (переміщення, масштабування тощо) та додавати еле-

менти інтерфейсу, наприклад, панелі і кнопки. Розглянемо приклад створення карти-врізки за допомогою **ui.Map** (див. *рис. 43)*.

```
var map = ui.Map();                                  // Створення карти-контейнера

var createInset = function() {                       // Додати в карту-контейнер карту-
  var bounds = ee.Geometry.                          врізку з основної карти
Rectangle(Map.getBounds());
  map.centerObject(bounds);
  map.clear();
  map.addLayer(bounds);
};
createInset();
Map.onClick(createInset);                            // Оновлювати карту-врізку, залежно
                                                     від обраної користувачем локалізації
print(map);                                          // Відобразити карту-врізку в консолі
```

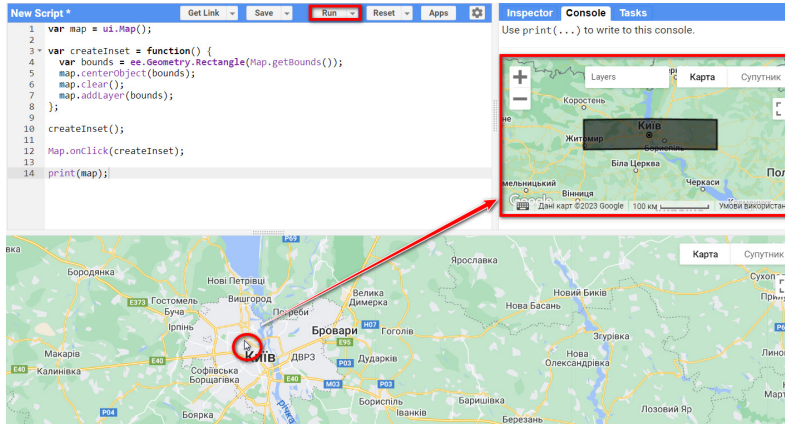


Рис. 43. Приклад створення інтерактивної карти-врізки за допомогою **ui.Map**

ui.Map.Linker є компонентом інтерфейсу Google Earth Engine, який дає змогу зв'язувати кілька екземплярів **ui.Map** разом, щоб забезпечити синхронізацію їхнього відображення та взаємодії. Коли ви використовуєте **ui.Map.Linker**, будь-які зміни в одному екземплярі **ui.Map** будуть автоматично відображатися в усіх підключених екземплярах **ui.Map**. Це корисний інструмент, коли вам потрібно відобразити одні й ті самі географічні дані або шари на декількох картографічних вікнах одночасно. Наприклад, якщо ви маєте два екземпляри **ui.Map**, ви можете зв'язати їх за допомогою **ui.Map.Linker**, щоби

під час переміщення або масштабування однієї карти інша також автоматично змінювалася відповідно. У результаті будь-які зміни в одному екземплярі **ui.Map** будуть відображатися в інших екземплярах **ui.Map**, які пов'язані за допомогою **ui.Map.Linker**.

Розглянемо приклад використання такого компонента (див. *рис. 44*):

```
var map1 = ui.Map();                                    // Створення двох карт
var map2 = ui.Map();

ui.root.widgets().reset([map1, map2]);                  // Додати створені карти до користувацького інтерфейсу
var linker = new ui.Map.Linker([map1, map2]);           // Створення компонента ui.Map.Linker

map1.setCenter(34.5385, 47.5494,10)                     // Налаштувати центр карти для відображення

var image = ee.Image('COPERNICUS/S2_SR_HARMONIZED/20230705T083601_20230705T084552_T36TWT');
map1.addLayer(image, {bands: ['B4', 'B3', 'B2'], max: 3000}, '2023');   // Додати дані до карти 1 (супутниковий знімок Sentinel-2 для території Каховського водосховища, липень 2023 рік)

var image2 = ee.Image('COPERNICUS/S2_SR_HARMONIZED/20220703T084611_20220703T085107_T36TWT');
map2.addLayer(image2, {bands: ['B4', 'B3', 'B2'], max: 3000}, '2022');  // Додати дані до карти 2 (супутниковий знімок Sentinel-2 для території Каховського водосховища, липень 2022 рік)
```

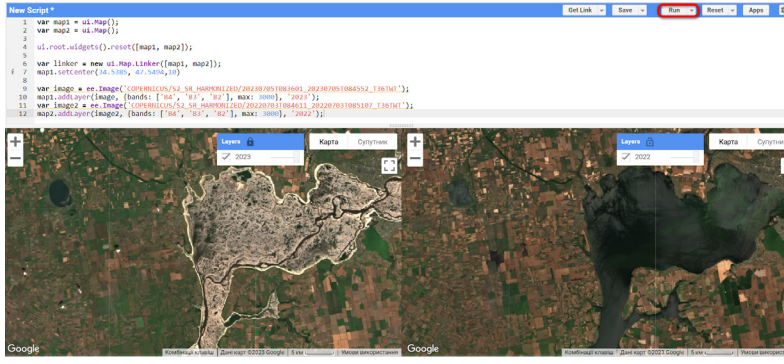


Рис. 44. Застосування **ui.Map.Linker** для порівняння двох супутникових знімків за різні дати

За допомогою **ui.Map.Linker** ви можете забезпечити синхронізацію відображення і взаємодії між різними екземплярами **ui.Map**, що полегшує порівняння, аналіз та візуалізацію географічних даних у різних контекстах. Так, за допомогою скрипту в розглянутому прикладі можна легко порівняти вигляд території Каховського водосховища після обміління у 2023 р. (внаслідок підриву греблі) з її виглядом у 2022 р.

ui.SplitPanel є компонентом інтерфейсу GEE, який дає змогу розділяти вікно на дві частини з можливістю регулювання розміру кожної з них. Завдяки цьому користувачі можуть відображати два різні зображення (наприклад, мапи, графіки або інші віджети) поряд і налаштовувати співвідношення їхніх розмірів, що полегшує порівняння, взаємодію й аналіз даних у різних контекстах.

Приклад використання **ui.SplitPanel** наведено нижче. У цьому прикладі **ui.SplitPanel** створює вертикальний роздільник і розташовує *map1* і *map2* як два окремі зображення поруч зліва і справа від роздільника. Користувачі можуть регулювати розмір кожного зображення шляхом перетягування роздільника. На відміну від компонента **ui.Map.Linker**, **ui.SplitPanel** не синхронізує зображення, що дає змогу сумісно переглядати знімки і карти та інші об'єкти для різних локалізацій і в різних масштабах. Наприклад (див. *рис. 45*):

`var map1 = ui.Map();` `var map2 = ui.Map();`	`// Створення двох карт`
`var image = ee.Image('COPERNICUS/S2/20210804T083601_20210804T084956_T36SUF');` `map1.addLayer(image, {bands: ['B4', 'B3', 'B2'], max: 3000});`	`// Додати дані до карти 1`
`var image2 = ee.Image('COPERNICUS/S2/20210804T083601_20210804T084956_T36SUF');` `map2.addLayer(image2, {bands: ['B4', 'B3', 'B2'], max: 3000});`	`// Додати дані до карти 2`
`map1.setCenter(30.9085, 37.0005,14),`	`// Налаштувати центр карти 1 для відображення`
`map2.setCenter(30.1000, 37.003,10)`	`// Налаштувати центр карти 2 для відображення`

`var splitPanel = ui.SplitPanel({`
`  firstPanel: map1,`
`  secondPanel: map2,`
`  orientation: 'horizontal',`
`});`
	`// Створення компонента ui.SplitPanel`

`ui.root.widgets().reset([splitPanel]);`	`// Додати компонент ui.SplitPanel до користувацького інтерфейсу`

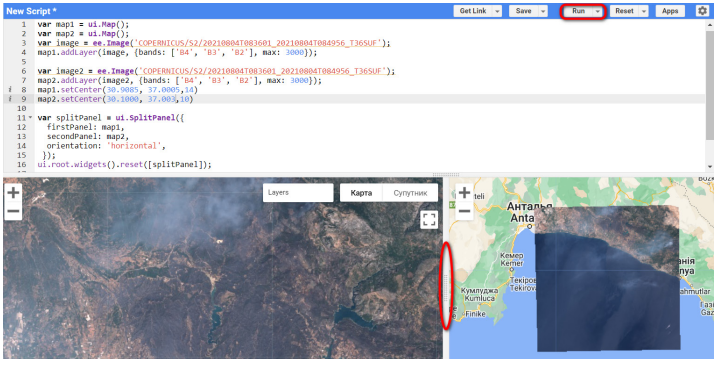


Рис. 45. Приклад застосування **ui.SplitPanel** у GEE

Якщо для порівняння двох карт необхідно переглядати знімки сумісно, то використання **ui.SplitPanel** комбінують із віджетом **ui.Map.Linker**. Також додатково можна використовувати аргумент *wipe*, значення якого за замовчуванням – **false** (вимкнено). Коли цей режим увімкнено (значення – **true**), обидві панелі займають увесь доступний простір, а перетягування роздільника не визначає розмір панелей, а визначає частину кожної панелі, яка буде показана. Наприклад, наведений нижче код у GEE дає змогу переглядати територію Анталії (Туреччина), яка постраждала від пожеж у серпні 2021 р., одночасно використовуючи різні візуалізації – композити *True color* і *False color* (див. *рис. 46*).

```
var map1 = ui.Map();                                                    // Створення двох карт
var map2 = ui.Map();

var image = ee.Image('COPERNICUS/S2/20210804T083601_20210804T084956_T36SUF');   // Додати дані (зображення Sentinel-2 за 2021 рік) до карти 1

var image2 = ee.Image('COPERNICUS/S2/20210804T083601_20210804T084956_T36SUF');  // Додати дані (зображення Sentinel-2 за 2021 рік) до карти 2

map1.addLayer(image, {bands: ['B4', 'B3', 'B2'], max: 3000}, 'True color');     // Створення композиту каналів B4, B3, B2
map2.addLayer(image, {bands: ['B8', 'B4', 'B2'], max: 3000}, 'False color');    // Створення композиту каналів B8, B4, B2
map1.setCenter(31.5085, 37.005, 12);

var linker = ui.Map.Linker([map1, map2]);                               // Створення компонента ui.Map.Linker

var splitPanel = ui.SplitPanel({                                        // Створення компонента ui.SplitPanel
  firstPanel: map1,
  secondPanel: map2,
  orientation: 'horizontal',
  wipe: true

});
ui.root.widgets().reset([splitPanel]);                                  // Додати компонент ui.SplitPanel до користувацького інтерфейсу
```

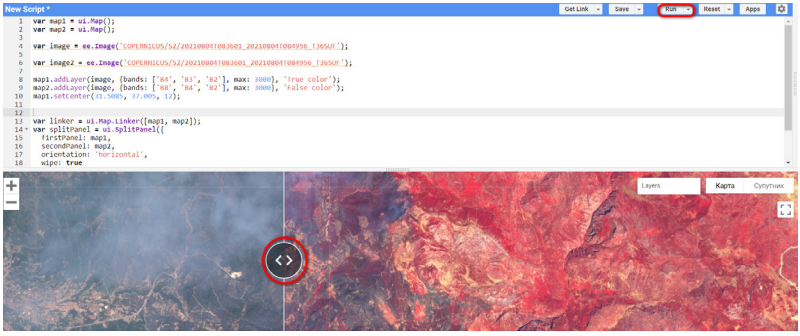


Рис. 46. Приклад застосування **ui.SplitPanel**
(при увімкненому режимі *wipe*) у GEE

Для розміщення інших віджетів, організації елементів інтерфейсу і створення більш складних макетів використовується віджет-контейнер **ui.Panel.** Наведемо приклад створення компонента такого типу (див. *рис. 47*):

```
var panel = ui.Panel({                          // Створити панель
  widgets: [
    ui.Label('Інформаційна панель'),            // Додати напис
    ui.Button('OK', function() {                // Додати кнопку та обробник подій
    print('Додаток активовано');                для неї
    })
  ],
  layout: ui.Panel.Layout.flow('vertical'),     // Встановити розташування у верти-
  style: {                                      кальному режимі
    width: '250px',                             // Встановити ширину панелі
    backgroundColor: 'lightgray'                // Задати колір фону панелі
  }
});

Map.add(panel);                                 // Додати панель на карту
```

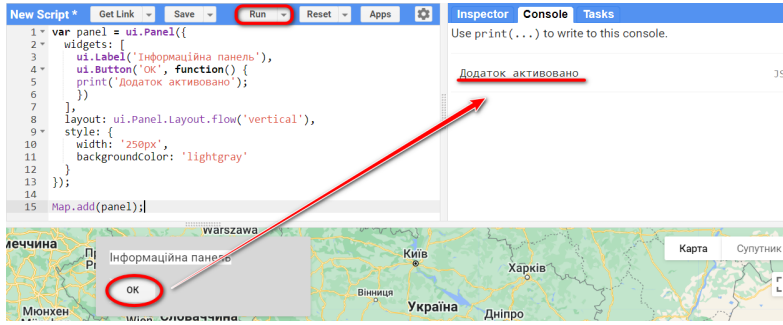


Рис. 47. Приклад створення панелі в GEE

Розробляючи вебзастосунки, можна довільно комбінувати розглянуті вище компоненти інтерфейсу для створення функціональних застосунків, зручних і зрозумілих для користувача.

Цікаво

UI (User Interface) дизайн – *це створення різних кнопок, меню та інших елементів інтерфейсу, які бачить на екрані комп'ютера чи телефону користувач, і за допомогою яких він може виконувати певні дії в програмі. UI дизайн гарний і зрозумілий на вигляд, він робить інтерфейс дружнім та легким для користування.*

UX (User Experience) дизайн – це розробка користувацького інтерфейсу таким, щоби було комфортно використовувати програми та сайти. Мета UX дизайну – створення інтерфейсу таким, щоб користувачі легко знаходили потрібні інструменти й функції і залюбки використовували застосунки.

Загалом UI/UX дизайн допомагає зробити технології зручними і приємними для користувачів, створюючи красиві та легкі для розуміння інтерфейси.

Створюючи різні візуальні елементи в GEE, як-от віджети, кнопки, карти, графіки тощо, їх можна розміщувати в консолі (*print ()*), додавати безпосередньо до вікна карти (*Map.add()*) чи організовувати за допомогою спеціально створеної панелі-контейнера **ui.Panel**, а також додавати до кореневого контейнера користувацького інтерфейсу за допомогою компонентів **ui.root**. Усі з перелічених способів були застосовані в розглянутих вище прикладах створення і використання різних віджетів. Вибір того чи іншого способу залежить від конкретних завдань вебзастосунку, що розробляється. Основна мета – створити зручний і зрозумілий користувачеві інтерфейс. При цьому найбільш часто застосовуються об'єкти **ui.root** і **ui.Panel**. Як уже зазначалось, **ui.Panel**

допомагає організувати інші елементи, вирівнювати їх у горизонтальному або вертикальному положенні, надає зручний спосіб розміщення елементів поруч один з одним, тоді як **ui.root** – це основний контейнер для створення інтерфейсу користувача в GEE. Він є кореневим об'єктом, до якого можна додавати інші UI-елементи, як-от **ui.Panel**, **ui.Label**, **ui.Button** тощо. Він більш гнучкий щодо розміщення та оформлення елементів, порівняно з **ui.Panel**, надає можливість створення складніших інтерфейсів із різними розміщеннями та оформленням елементів. Ось приклад створення кореневого контейнера інтерфейсу та додавання до нього декількох віджетів (див. *рис. 48*):

```
var years = [2019, 2020, 2021, 2022];                    // Створення віджета
var yearSelector = ui.Select({
  items: years.map(function(year) {
    return {label: year.toString(), value: year};
  }),
  value: years[0],
  style: {width: '100px'}
});
var showButton = ui.Button({
  label: 'Відобразити',
  style: {width: '100px'},
  onClick: function() {
    displayImagesForYear(yearSelector.
getValue());
  }
});
var instructions = ui.Label('Виберіть рік та
натисніть "Відобразити":');
var rootContainer = ui.Panel({                           // Створення кореневого кон-
  widgets: [instructions, yearSelector,                  тейнера для інтерфейсу
showButton],
  layout: ui.Panel.Layout.Flow('vertical'),
  style: {width: '250px'}
});

                                                         // Додати кореневий контейнер
ui.root.add(rootContainer);                              до користувацького інтерфейсу
```

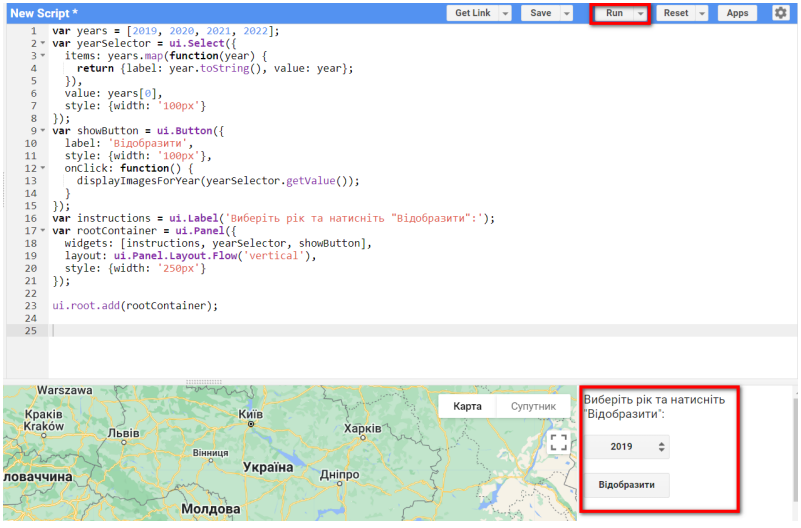


Рис. 48. Приклад створення і додавання групи компонентів до кореневого інтерфейсу GEE

Отже, GEE надає широкий перелік спеціальних об'єктів, які можуть бути використані для створення інтерактивних вебзастосунків для взаємодії з користувачем, управління та відображення інформації.

5.3. Огляд застосунків Earth Engine Apps

Earth Engine Apps – інструмент, який дає змогу створювати вебзастосунки з різними елементами інтерфейсу.

Проєкти, опубліковані як вебзастосунки Earth Engine Apps, будуть доступні всім користувачам без потреби взаємодіяти з вихідним кодом. Публікація результатів досліджень на такій платформі дає змогу візуалізувати дані, а широке коло інструментів – створити стилістично індивідуальний геоінформаційний продукт. Для пошуку та використання кінцевого продукту необхідний лише інтернет-браузер, що значно полегшує доступ до результатів дослідження й розширює охоплення зацікавленої аудиторії. Застосування візуального інтерфейсу в таких вебзастосунках дає змогу користувачам працювати з GEE, не використовуючи безпосередньо код або складні команди. Зручний та інтуїтивно зрозумілий графічний інтерфейс для взаємодії з геопросторовими даними значно полегшує аналіз та візуалізацію даних на платформі Google

Earth Engine, спрощує доступ до масивних обчислень та обробки великого обсягу геопросторових даних, що зберігаються у хмарі.

На сайті Earth Engine Apps представлені деякі приклади таких застосунків (за покликанням https://www.earthengine.app/) із можливістю відкрити вихідний код, що може бути корисним для вивчення цього інструменту (див. *рис. 49*).

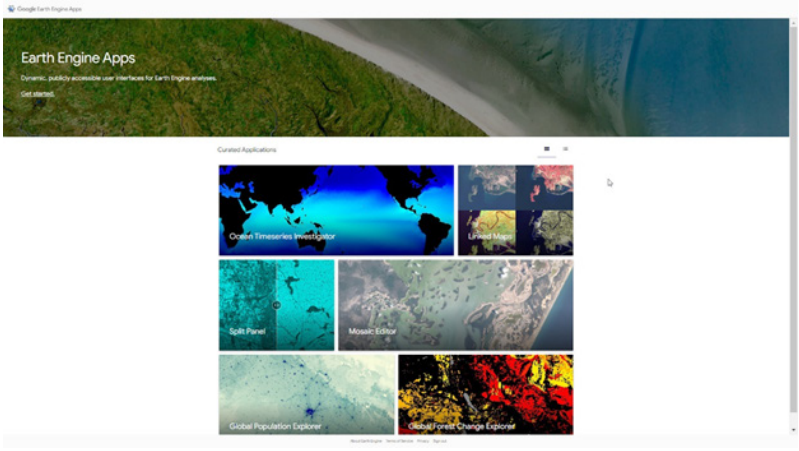


Рис. 49. Вебзастосунки Earth Engine Apps із відкритим кодом

Більше прикладів вебзастосунків різної тематики можна знайти за покликанням https://datawrapper.dwcdn.net/4cHkZ/5/, де сформовано перелік опублікованих у вільному доступі різними користувачами застосунків GEE Apps станом на 23 грудня 2020 р.

Також ознайомитися з деякими опублікованими у вільному доступі застосунками Google Earth Engine Apps, розробленими для вивчення різноманітних явищ і процесів на нашій планеті, ви можете, скориставшись інформацією, яка розміщена в додатках до цього посібника.

Слід зазначити, що серед дослідників та науковців різних галузей природничих наук зростає зацікавленість можливостями, які надає платформа GEE. Це підтверджується, зокрема, результатами бібліометричного аналізу [11–18] статей, тематика яких пов'язана з GEE, опублікованих у виданнях, що індексуються в базі даних Scopus до грудня 2022 р. Результати аналізу [11] свідчать, що кількість статей, пов'язаних із GEE, стрімко зростає (так, серед більш ніж 2000 статей, розміщених у рецензованих виданнях Scopus, майже 85% опубліковано за останні три роки). Авторами більшості статей, пов'язаних із GEE, є вчені з Китаю та США, що становить 58% від загальної

кількості [12; 13]. Трьома основними сферами, де GEE широко застосовується, є науки про Землю [14; 15], довкілля [16; 17], а також сільськогосподарські та біологічні науки [18].

Застосування GEE охоплює широкий спектр тем – від землекористування, водних ресурсів, змін клімату до картографування сільськогосподарських культур як для регіонального, так і для глобального рівня досліджень (див. *рис. 50)*.

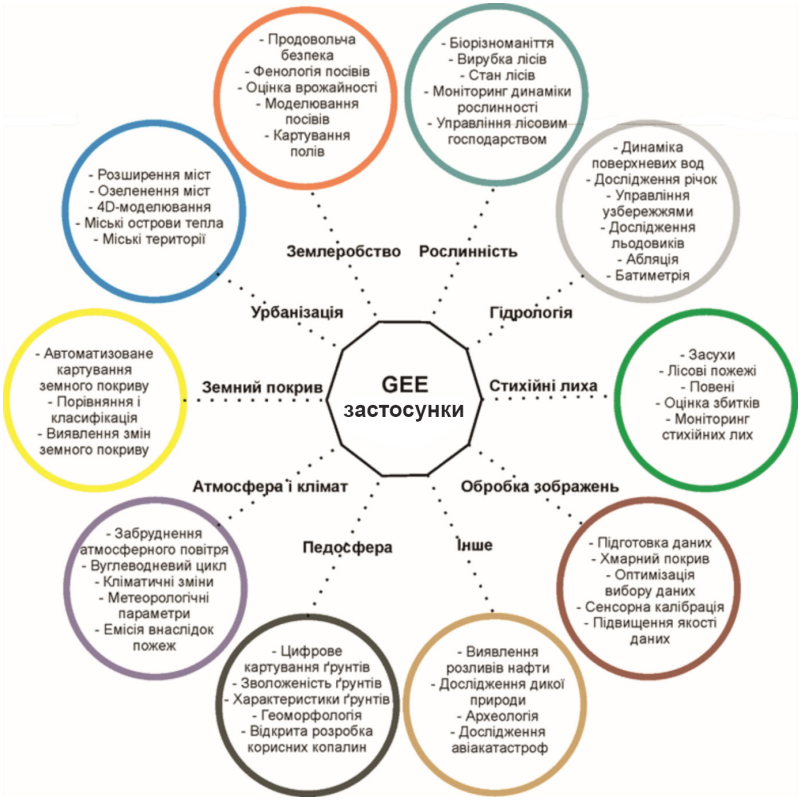


Рис. 50. Тематичні категорії розроблених на основі GEE програмних застосунків, результати використання яких описані в наукових публікаціях (за даними, наведеними в [3])

Загалом вебзастосунки в GEE – це потужний інструмент для того, щоб ділитися власними дослідженнями.

5.4. Публікація інтерактивного застосунку

Щоб опублікувати власний вебзастосунок, створений у GEE, спочатку необхідно завантажити скрипт, який потрібно перетворити на програму у вікні редактора коду, а потім відкрити панель керування застосунками, натиснувши кнопку ***Apps*** *(Застосунок)* (див. *рис. 51*).

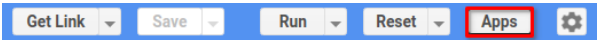


Рис. 51. Кнопка керування застосунками

У діалоговому вікні, що відкриється, можна створити новий вебзастосунок ***New App*** *(Новий застосунок)*, вибравши для його розміщення наявний проєкт Google Cloud (див. *рис. 52*).

Рис. 52. Кнопка створення нового вебзастосунку

Якщо раніше Google Cloud Projects не використовувався, необхідно його створити, це буде ***Add cloud project*** *(Додати хмарний проєкт)*.

Важливо! *Один і той самий проєкт Google Cloud можна використовувати для всіх створених користувачем застосунків.*

Цікаво

Працюючи із хмарними технологіями, замість того, щоб використовувати власні ресурси – комп'ютери та програми – ви користуєтеся службами, які надають доступ до різних ресурсів. Google Cloud складається з набору фізичних активів, таких як комп'ютери та жорсткі диски, і віртуальних ресурсів (віртуальних машин), які містяться в центрах обробки даних Google по всьому світу. Google Cloud надає доступ користувачам до багатьох ресурсів і сервісів, список яких постійно росте. Після того як ви обрали потрібний сервіс, ви додаєте свій власний код – це інструкції щодо того, як використовувати дані і ресурси для вирішення конкретних завдань. Усі ресурси Google Cloud, які ви обираєте й використовуєте, мають належати до проєкту. Проєкт складається з налаштувань, дозволів та інших метаданих, які описують ваші програми. Ресурси в рамках одного проєкту можуть легко працювати разом, наприклад, поєднуючись через внутрішню мережу.

Кожен проєкт Google Cloud повинен мати назву, яку надає користувач, ідентифікатор проєкту, який може надати Google Cloud або користувач може створити самостійно, і номер проєкту, який генерує Google Cloud. Рекомендовано створити проєкт під назвою «ee-USERNAME», де USERNAME – назва

облікового запису GEE. Також, якщо використовується установчий обліковий запис, потрібно вибрати назву організації.

Далі потрібно налаштувати параметри доступу до редагування застосунку, що дає змогу розробляти його як командний проєкт (див. *рис. 53*).

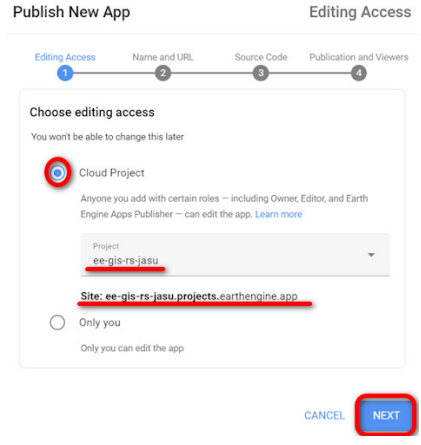


Рис. 53. Налаштування доступу для редагування вебзастосунку

Наступний крок дає змогу задати ім'я застосунку і згенерувати його URL (див. *рис. 54*).

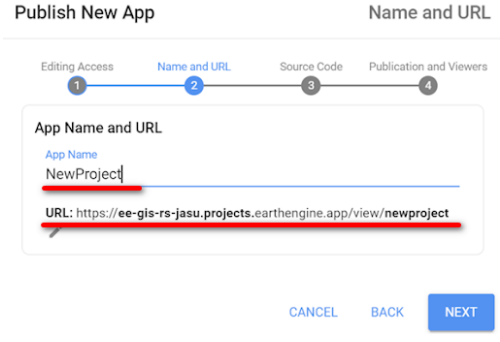


Рис. 54. Створення імені й адреси вебзастосунку

Далі потрібно вказати розташування вихідного коду програми (див. *рис. 55*).

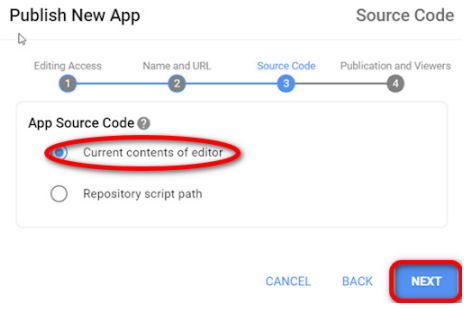


Рис. 55. Призначення джерела програмного коду вебзастосунку

На завершальному етапі необхідно на вкладці ***Gallery*** *(Галерея)* активувати опцію ***Feature this app in your Public Apps Gallery*** *(Розмістіть цей застосунок у власній галереї застосунків)*, щоби вебзастосунок був опублікований у загальнодоступній галереї, за адресою USERNAME.users.earthengine.app. Додатково тут можна завантажити спеціальне зображення (логотип застосунку) для попереднього перегляду програми, а також її опис.

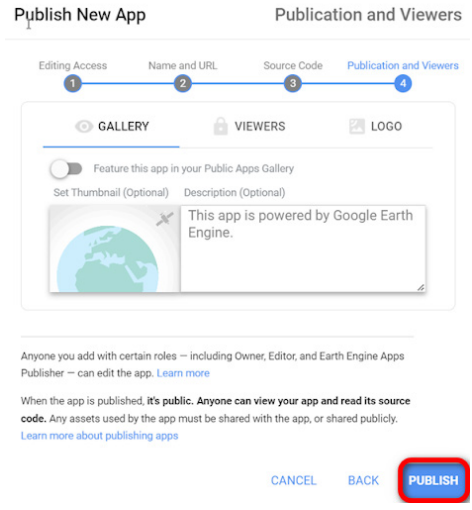


Рис. 56. Публікація і налаштування особливостей перегляду вебзастосунку

Важливо! *Якщо ви хочете розмістити вебзастосунок у загальному доступі, то будь-які ресурси (зображення або таблиці), які використовуються у ньому, мають бути загальнодоступними, щоб належно відображатися. Для цього в діалоговому вікні налаштування спільного доступу виберіть параметр Anyone can read (Будь-хто може читати) або виберіть із випадного списку назву застосунку, в якому будуть використовуватися дані* (див. *рис. 57*).

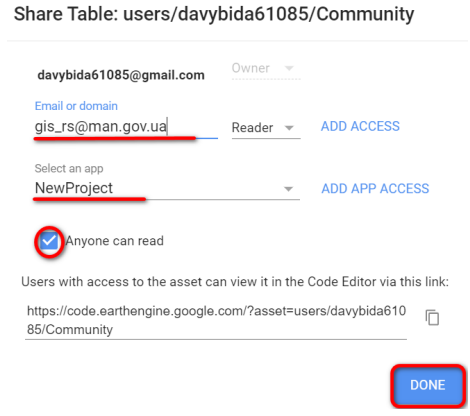


Рис. 57. Приклад налаштування доступу до ресурсів, що використані у вебзастосунку

Для керування вебзастосунками також використовується уже знайома вам кнопка **Apps**. Натиснувши на неї, отримуємо доступ до переліку створених застосунків, які можна оновлювати або видаляти в разі потреби (див. *рис. 58*).

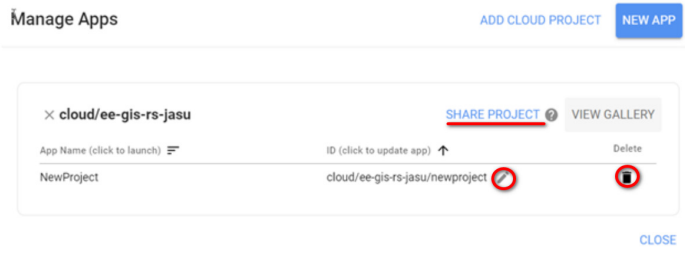
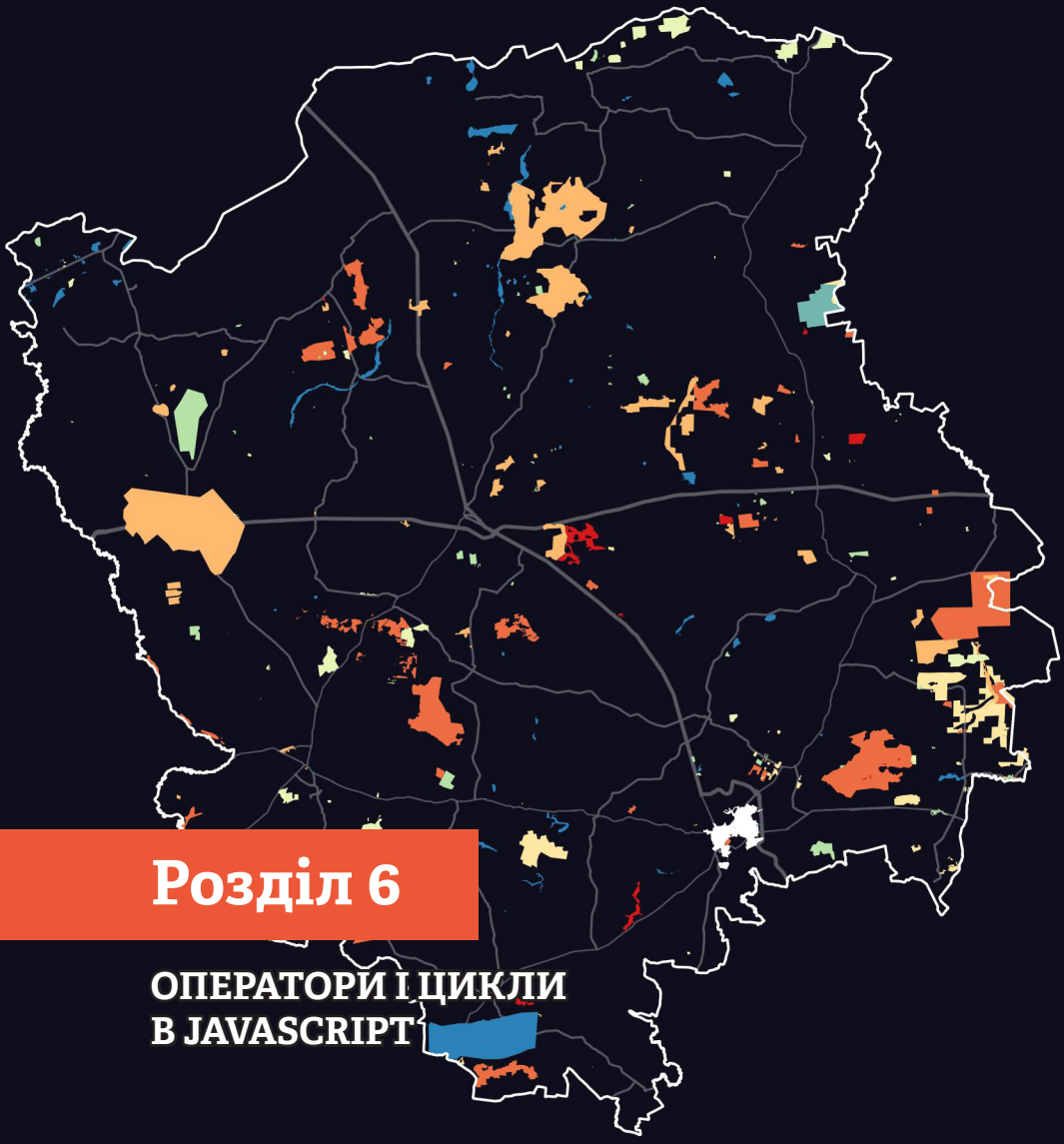


Рис. 58. Панель керування вебзастосунками

Після публікації вебзастосунку GEE покликання на нього можна поширювати серед інших користувачів.



Розділ 6

ОПЕРАТОРИ І ЦИКЛИ В JAVASCRIPT

6.1. Математичні оператори

Умовні оператори в програмуванні – це такі «команди», які дають змогу програмі самостійно приймати рішення залежно від певних умов. Це схоже на вибір, який ви робите у повсякденному житті. Наприклад, якщо надворі йде дощ, то ви берете із собою парасольку, якщо погода безхмарна – сонцезахисні окуляри. Умовний оператор допомагає програмі приймати рішення так само.

Отже, в процесі роботи з геоданими в GEE нерідко виникає потреба в складних аналітичних рішеннях, залежних від певних умов. Це створює необхідність використання певних операторів, які дають змогу вибирати між різними діями на основі заданих критеріїв. У цьому розділі ми детально розглянемо умовні оператори, зокрема оператори порівняння, логічні оператори, і наведемо приклади їх використання.

Значення умовних операторів в аналізі даних

Умовні оператори – це інструкції, які використовуються для прийняття рішень на основі вказаних умов. Вони дають вам змогу контролювати перебіг виконання програми, виконуючи дії лише в тому разі, якщо задані умови відповідають певним значенням. Умовні оператори є невід'ємною частиною аналітичних процесів, вони дають змогу робити ваші аналізи більш гнучкими й ефективними.

Використання таких структур робить ваш скрипт більш лаконічним.

Математичні оператори використовуються для виконання арифметичних операцій, як-от обчислення середнього значення температур, сум опадів, розрахунку площі або будь-яких інших математичних операцій, які можуть знадобитися під час аналізу геоданих. Основні математичні оператори містять:

+ (додавання)
- (віднімання)
* (множення)
/ (ділення)
% (взяття залишку)

```
var x = 10;                          // Змінна x дорівнює 10
var y = 5;                           // Змінна y дорівнює 5

var sum = x + y;                     // Сума x та y
print(sum)
var difference = x - y;              // Різниця x та y
print(difference)
var product = x * y;                 // Множення x та y
print(product)
var quotient = x / y;                // Ділення x та y
print(quotient)
var remainder = x % y;               // Ділення без залишку x та y
print(remainder)
```

6.2. Оператори порівняння

Порівняльні оператори використовуються для порівняння значень. Вони дають змогу програмі порівнювати два значення і визначати, чи одне значення більше, менше, рівне чи не рівне іншому значенню. Наприклад, ці оператори використовуються для порівняння, площа якої країни більша чи менша, ніж України, в яких містах нашої держави кількість населення більша або дорівнює 1 млн жителів тощо. Основні порівняльні оператори містять:

== (рівність)
!= (нерівність)
> (більше)
< (менше)
>= (більше або рівне)
<= (менше або рівне)

```
var a = 10;                          // Змінна a дорівнює 10
var b = 5;                           // Змінна b дорівнює 5

var isEqual = a == b;                // Рівність a та  b
print(isEqual)
var isNotEqual = a != b;             // Нерівність a та  b
print(isNotEqual)
var isGreater = a > b;               // a більше b
print(isGreater)
var isLess = a < b;                  // a менше b
print(isLess)
var isGreaterOrEqual = a >= b;       // a більше або дорівнює b
print(isGreaterOrEqual)
var isLessOrEqual = a <= b;          // a менше або дорівнює b
print(isLessOrEqual)
```

6.3. Логічні оператори

Логічні оператори використовуються для об'єднання та порівняння логічних значень (true або false). Вони дають змогу програмі виконувати операції на основі правдивості або хибності виразів. Наприклад, логічні оператори можна використовувати для визначення придатності території для сільськогосподарського використання, аналізуючи значення NDVI (яке має бути вище ніж 0, але не більше ніж 0,55), дані про висоту (вище ніж 200 м над рівнем моря, але не більше ніж 400) та інші характеристики поверхні, якщо всі задані критерії підпадають під значення true, то програма видасть ці території у результуючу карту. Основні логічні оператори містять:

&& (логічне І, «і»)
|| (логічне АБО, «або»)
! (логічне НЕ, «не»)
Приклади:

```
var x = 10;
var y = 5;

var resultAnd = x > 0 && y > 0;
print("x > 0 і y > 0: " + resultAnd);

var resultOr = x < 0 || y < 0;
print("x < 0 або y < 0: " + resultOr);


var resultNot = !(x === 0);
print("x не дорівнює 0: " + resultNot);
```

```
// Логічне І (&&): Перевірка, чи обидва числа більші 0

// Логічне АБО (||): Перевірка, чи хоча б одне число менше 0

// Логічне НЕ (!): Перевірка, чи x не дорівнює 0
```

6.4. Умовні оператори if else і цикли for, while

Умовні оператори – це спеціальні команди, які дають змогу програмі виконувати різні дії залежно від заданої умови. Ці оператори допомагають вибрати дані, які відповідають певним критеріям, виконати дії, якщо певна умова виконується, а також визначити, чи виконується певна умова, тощо.

У GEE ви можете використовувати такі основні умовні оператори:

- **if** (якщо): дає змогу виконати певні дії, якщо певна умова виконується;
- **else** (інакше): використовується для визначення дій, які будуть виконані, якщо умова вище не виконується;

- **else if** (інакше, якщо): дає змогу перевірити додаткові умови, якщо попередня умова не виконується.

```
var rain= true;                          // Змінна, що показує, чи дощить (true або false)

if (rain) {

  print("Візьми парасольку");
} else {                                 // Умовний оператор if

  print("Не бери парасольку");           // Якщо дощить, візьми парасольку
}

                                         // Якщо не дощить, не бери парасольку
```

Умовні оператори можуть бути використані для різних аспектів аналізу даних у GEE. Уявіть собі, що ви хочете визначити зону ризику повені в певній області. Ви можете використовувати умовні оператори, щоб визначити, які райони найбільш схильні до повеней. За допомогою даних та умовних операторів ви можете зробити запит і визначити, чи є піксель нижче рівня моря, чи він у низині, чи розташований він на схилі більше 30 градусів, тощо.

If else оператори

Ось приклад застосування **if else** операторів у GEE.

У цьому прикладі ми використовуємо дві змінні: a і b, які використовуються в інструкції **if** для перевірки того, чи b більше за a. Оскільки a дорівнює «33», b дорівнює «200», ми знаємо, що 200 більше за 33, і тому виводимо на екран «b більше за a».

```
var a = 33;                              // Приклад оператора if
var b = 200;

if (b > a) {
  print("b більше ніж a");
}
```

if else

Ключове слово **if else** – це спосіб мови JavaScript сказати: «Якщо попередні умови не були істинними, то спробуйте виконати наступну умову».

```
var a = 33;
var b = 200;

if (b > a) {
  print("b більше ніж a");
} else if (a === b) {
  print("b менше ніж a");
}
```

// Приклад оператора **else if**

Ключове слово **else** перехоплює все, що не перехоплено попередніми умовами. Тобто якщо умова **if** не виконана, то умова **else** береться до виконання.

```
var a = 33;
var b = 33;

if (b > a) {
  print("b більше ніж a");
} else if (a === b) {
  print("b дорівнює a");
}
```

// Приклад оператора **else**

Також **if else** може бути в коді скільки завгодно.

```
var a = 33;
var b = 11;

if (b > a) {
  print("b більше ніж a");
} else if (a === b) {
  print("b дорівнює a");
}
else if (b < a) {
  print("b менше ніж a");
}
```

// Приклад оператора **if else**

Короткі **if else**

Якщо вам потрібно виконати лише один оператор, один для **if** і один для **else**, ви можете розмістити їх в одному рядку.

```
var a = 2;
var b = 330;

print(a > b ? "A" : "B");
```

```
// Порівняння двох значень
```

While Loops *(Цикли з передумовою)*

Цикли з передумовою (**while loops**) використовуються для виконання блоку коду, поки певна умова є істинною. Наприклад, **while loops** можна використовувати для аналізу великих наборів даних, повторюючи певну операцію, поки не буде проаналізований весь набір даних.

Ось приклад циклу **while** у GEE:

```
var number = 1;

while (number <= 5) {
  print(number);
  number= number + 1;
}
```

```
// Приклад циклу while в GEE
```

Цей код буде виводити числа від 1 до 5 у вікні консолі. Умова число <= 5 перевіряє, чи число менше або дорівнює 5. Поки ця умова виконується (тобто число менше або дорівнює 5), цикл буде виконуватися.

For Loops *(Цикли for)*

Цикл **for** використовуються для виконання певного блоку коду певну кількість разів. Вони дуже корисні для ітерації, тобто повторення, через масиви або інші структури даних у GEE (див. *рис. 59*).

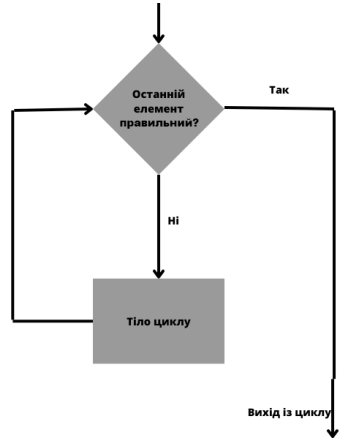


Рис. 59. Приклад ітерації (повторення) коду для вхідних даних

Синтаксис написання циклу **for** такий:
він приймає три аргументи.
Ініціалізація задається як i = 1, де початок 1
Умова i <= 5, тобто менше 5, включаючи 5
Вираз i = i + 1 крок ітерації

```
for (ініціалізація; умова; вираз) {
 // Код, який виконується в кожній ітерації
 }
```

```
for (var i = 1; i <= 5; i = i + 1) {
  print(i);
}
```

```
// var i = 1: Ініціалізація змінної і зі значенням 1.
// i <= 5: Умова, яка перевіряється перед кожним виконанням циклу.
//  i = i + 1: Вираз, який виконується після кожного виконання циклу, для збільшення значення і.
```

Ітерація (від лат. *iteratio* – повторювання) – результат багаторазового повторення якоїсь математичної операції.

У цьому прикладі цикл **for** виводить числа від 1 до 5 у консоль GEE. Кожна ітерація збільшує значення і на 1, і цикл виконується, поки і не стане дорівнювати 5.

6.5. Розширене використання операторів

Для того щоб зрозуміти різницю між циклами **for** і **while**, припустимо, у вас є коробка з різними предметами. Цикл **for** буде подібний до того, як ви по черзі виймаєте кожен предмет з коробки і щось із ним робите. Не потрібно думати про те, коли ви закінчите, бо цикл **for** автоматично пройде через усі предмети в коробці.

Натомість цикл **while** подібний до того, як ви виймаєте предмети з коробки, поки не знайдете необхідний предмет. Ви продовжуєте діставати предмети, доки не виконається умова.

Отже, **for** – це як розглядати кожен предмет по черзі, тоді як **while** – робити це, доки не досягнеш певної умови (коробка порожня).

Умова виконання:

- у циклі **for** визначається фіксована кількість ітерацій на початку виконання; зазвичай відомо, скільки разів цикл повинен виконуватися;
- у циклі **while** виконання триває, доки вказана умова залишається істинною. Умова перевіряється перед кожною ітерацією, і якщо вона стає хибною, виконання циклу припиняється.

Ситуації застосування:

- **for** зазвичай використовується, коли кількість ітерацій вам відома заздалегідь, ітерації базуються на послідовностях або колекціях;
- **while** корисний, коли ви хочете виконувати цикл, доки деяка умова є істинною, і ви не знаєте точну кількість ітерацій заздалегідь.

Нескінченні цикли:

- цикл **for** не так просто використовувати для нескінченних ітерацій, особливо якщо кількість ітерацій не відома вам наперед;
- **while** може легко створити нескінченний цикл, який виконується, доки умова істинна.

Отже, вибір між **for** і **while** залежить від конкретної задачі та структури даних, з якими ви працюєте.

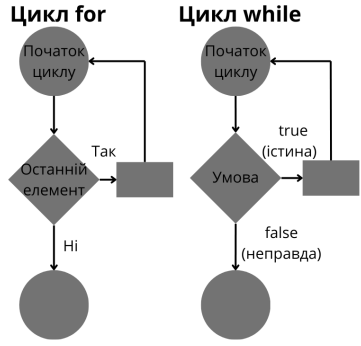


Рис. 60. Різниця між циклами **for** і **while**

Звісно, всі ці оператори та методи можна використовувати і для супутникових знімків та географічних даних. Розглянемо приклад, де буде використовуватися оператор **if else** для знаходження хмарних знімків.

```
var image = ee.Image('LANDSAT/LC08/C02/T1_TOA/LC08_186026_20230330');
print(image)
var cloud = image.select('B9');

var cloudThreshold = 60;

var meanCloudValue = cloud.reduceRegion({
  reducer: ee.Reducer.mean(),
  geometry: image.geometry(),
  scale: 30,
  maxPixels: 1e9
});

if (ee.Number(meanCloudValue.get('B9')).gt(cloudThreshold)) {
  print('Бери парасольку, хмари на знімку');
} else {
  print('Не бери парасольку, хмар немає на знімку');
}

Map.addLayer(image,{bands:['B3','B2','B1'], min: 0, max: 1},'image');
```

```
// Задаємо зображення Landsat 8

// Вибираємо канал хмар

// Задаємо поріг для визначення наявності хмар
// Визначаємо середнє значення каналу хмар для всього зображення
// Масштаб зображення
// Максимальна кількість пікселів для обробки

// Використовуємо оператор if else для визначення, чи потрібно взяти парасольку

// Додавання шару на мапу
```

Уявімо, що нам необхідно знайти кількість хмарних пікселів на всьому супутниковому знімку або на певній території. Для цього може бути вдало використаний інший ітератор **while**, який повторює дії задану кількість разів. У наступному прикладі maxIterations вказує на кількість разів, яку скрипт пройде через блок коду.

```
var image = ee.Image('LANDSAT/LC08/C02/T1_TOA/LC08_185026_20211112');                    // Задаємо зображення Landsat 8

var cloud = image.select('B9');

var initialCloudThreshold = 0.2;                                                         // Початковий поріг для визначення наявності хмар

var maxIterations = 10;                                                                  // Максимальна кількість ітерацій

var cloudCount = ee.Number(cloud.gt(initialCloudThreshold).reduceRegion({                // Початкова кількість хмар на зображенні
  reducer: ee.Reducer.sum(),
  geometry: image.geometry(),
  scale: 30,
  maxPixels: 1e9
}).get('B9'));

print('Початкова кількість хмар:', cloudCount);                                         // Виводимо початкову кількість хмар

var iteration = 0;                                                                       // Цикл while для ітеративного зменшення порогу
while (iteration < maxIterations && cloudCount.gt(100)) {

  initialCloudThreshold = initialCloudThreshold - 0.05;                                  // Зменшуємо поріг

  cloudCount = ee.Number(cloud.gt(initialCloudThreshold).reduceRegion({                  // Визначаємо кількість хмар за новим порогом
    reducer: ee.Reducer.sum(),
    geometry: image.geometry(),
    scale: 30,
    maxPixels: 1e9
  }).get('B9'));

  print('Ітерація', iteration + 1, 'Поріг:', initialCloudThreshold, 'Кількість хмар:', cloudCount);   // Виводимо інформацію
  iteration = iteration + 1;                                                             // Збільшуємо лічильник ітерацій
}
print('Кінцевий поріг:', initialCloudThreshold);                                         // Виводимо кінцевий поріг
Map.addLayer(image)
```

Щоб детальніше розглянути цикл, можна створити об'єкт і розмалювати його, поки в списку кольорів ще є вільні кольори. Тобто в списку colors є п'ять кольорів. Цикл **while** буде працювати, поки в списку ще будуть залишатися об'єкти.

```
var colors = ['red', 'orange',
'yellow', 'green', 'blue'];

var colorIndex = 0;

while (colorIndex < colors.length) {

  Map.addLayer(ee.Geometry.
Point([0, 0]).buffer(500), {color:
colors[colorIndex]}, 'Колір: ' +
colors[colorIndex]);

  colorIndex++;
}
Map.setCenter(0, 0, 12);
```

```
// Ініціалізуємо список кольорів.

// Ініціалізуємо індекс кольору.

// Використовуємо цикл while для
відображення кольорів на карті.

// Створюємо та відображаємо точку на
карті з кольором.

// Збільшуємо індекс для наступної
ітерації.
```

Загалом кожен представлений метод може використовуватися для різноманітних задач для автоматизації ваших розрахунків, пошуку потрібної інформації тощо.

Висновки

Цей посібник укладено з метою познайомити МАНівську спільноту з хмарним сервісом GEE і мовою програмування JavaScript для обробки та аналізу геоданих. Почавши з історії створення GEE, ми перейшли до основ JavaScript, охопили інформацію про змінні, типи даних та взаємодію між стороною сервера і стороною користувача.

Наступні кроки були такі: розгляд колекцій даних у GEE, в тому числі растрових і векторних колекцій геоданих, ознайомлення з методами фільтрації колекцій, які дають змогу вибирати лише необхідні для нашого дослідження дані, розгляд редукторів для операцій з даними, в тому числі для статистичних обчислень.

У розділі про функції розглянули, як створювати і використовувати функції в JavaScript для оптимізації нашого коду та полегшення роботи з даними.

Крім того, ми ознайомилися з користувацьким інтерфейсом GEE, включаючи модулі та застосунки Earth Engine Apps. Також ми навчилися публікувати інтерактивні застосунки для спрощення взаємодії з користувачами.

У підсумковому розділі розглянули оператори й цикли в мові програмування JavaScript, у тому числі умовні оператори, оператори порівняння, логічні оператори й цикли. Це важливі концепції для створення складних програм та аналізу даних.

Загальною метою цього посібника було не лише надати вам теоретичні знання, а й сформувати практичні навички роботи з Google Earth Engine за допомогою мови програмування JavaScript. Сподіваємося, що він буде корисним у вашій науковій та освітянській діяльності і допоможе досягти нових висот у світі геоінформатики та дистанційного зондування Землі.

Список використаних джерел

1. Google Introduces the Google Earth Engine. URL: https://www.historyofinformation.com/detail.php?id=2740 (дата звернення: 12.09.2023).
2. NASA Earth Sciences' Applied Science Program. URL: https://appliedsciences.nasa.gov/ (дата звернення: 12.09.2023).
3. Landsat Data Enriches Google Earth. URL: https://spinoff.nasa.gov/Spinoff2015/ee_1.html (дата звернення: 12.09.2023).
4. Google Earth Engine: Planetary-scale geospatial analysis for everyone / N. Gorelick et al. *Remote Sensing of Environment*. 2017. Vol. 202. DOI: https://doi.org/10.1016/j.rse.2017.06.031.
5. Introducing Google Earth Engine. URL: https://blog.google/outreach-initiatives/sustainability/introducing-google-earth-engine/ (дата звернення: 12.09.2023).
6. Datasets tagged nasa in Earth Engine. URL: https://developers.google.com/earth-engine/datasets/tags/nasa (дата звернення: 12.09.2023).
7. Chapter 4. How JavaScript Was Created. URL: https://web.archive.org/web/20200227184037/https://speakingjs.com/es5/ch04.html (дата звернення: 12.09.2023).
8. A planetary-scale platform for Earth science data & analysis. URL: https://earthengine.google.com/ (дата звернення: 12.09.2023).
9. Programming and Remote Sensing Basics. JavaScript and the Earth Engine API. URL: https://google-earth-engine.com/Programming-and-Remote-Sensing-Basics/JavaScript-and-the-Earth-Engine-API/ (дата звернення: 12.09.2023).
10. FAO GAUL: Global Administrative Unit Layers 2015, First-Level Administrative Units. URL: https://developers.google.com/earth-engine/datasets/catalog/FAO_GAUL_2015_level1 (дата звернення: 12.09.2023).
11. Pham-Duc B., Nguyen H., Phan H., Tran-Anh Q. Trends and applications of Google Earth Engine in remote sensing and Earth science research: a bibliometric analysis using Scopus database. *Earth Sci Inform*. 2023. Vol. 16. Pp. 2355–2371. DOI: https://doi.org/10.1007/s12145-023-01035-2.
12. Google Earth Engine: A Global Analysis and Future Trends / Andrés Velastegui-Montoya et al. *Remote Sens*. 2023. Vol. 15. Issue 14. P. 3675. DOI: https://doi.org/10.3390/rs15143675.
13. Google Earth Engine Cloud Computing Platform for Remote Sensing Big Data Applications: A Comprehensive Review / M. Amani et al. *IEEE Journal of*

Selected Topics in Applied Earth Observations and Remote Sensing. 2020. Vol. 13. Pp. 5326–5350. DOI: https://doi.org/10.1109/JSTARS.2020.3021052.

14. Давибіда Л. І. Аналіз можливостей і досвіду використання платформи Google Earth Engine для вирішення задач моніторингу довкілля. *Екологічна безпека та збалансоване ресурсокористування*. 2022. № 2. С. 75–86. DOI: https://doi.org/10.31471/2415-3184-2021-2(24)-75-86.

15. Trochim E. A new era of spatial analysis: AK CASC Fellow Erin Trochim hosts Google Earth Engine workshop. URL: https://akcasc.org/2020/02/25/a-new-era-of-spatial-analysis-ak-casc-fellow-erin-trochim-hosts-google-earth-engine-workshop/ (дата звернення: 12.09.2023).

16. Tripathy P., Malladi T. Global Flood Mapper: a novel Google Earth Engine application for rapid flood mapping using Sentinel-1 SAR. *Natural Hazards*. 2022. Vol. 114. Pp. 1341–1363. DOI: https://doi.org/10.1007/s11069-022-05428-2.

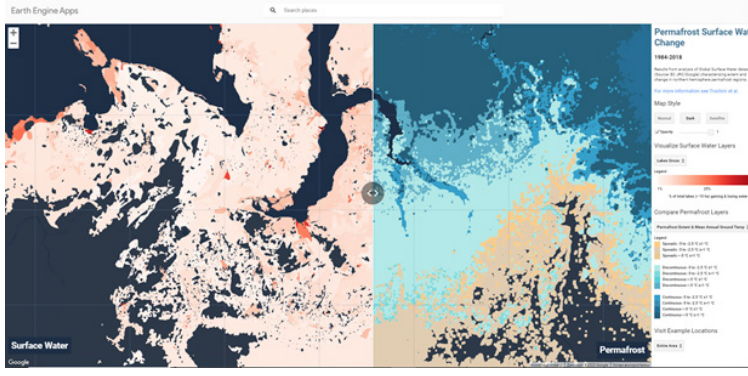
17. Naive Bayes classification-based surface water gap-filling from partially contaminated optical remote sensing image / B. Bai et al. *Journal of Hydrology*. 2023. Vol. 616. Pp. 128791. DOI: https://doi.org/10.1016/j.jhydrol.2022.128791.

18. Dong E., Du H., Gardner L. An interactive web-based dashboard to track COVID-19 in real time. *The Lancet Infectious Diseases*. 2020. Vol. 20. Pp. 533–534. DOI: https://doi.org/10.1016/S1473-3099(20)30120-1/.

Додатки

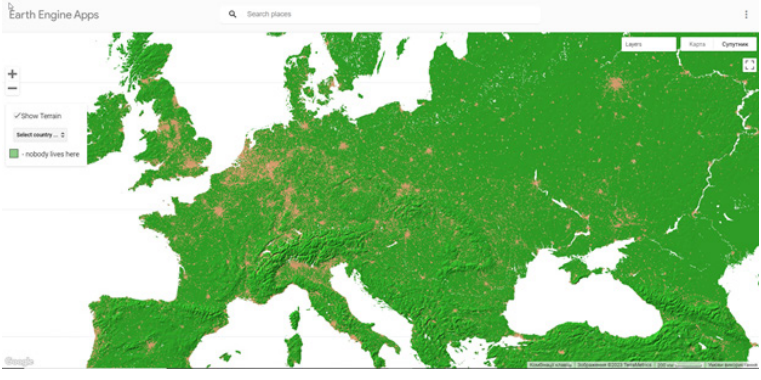
Приклади опублікованих вебзастосунків Earth Engine Apps

1. Вебзастосунок для відстеження змін обводненості поверхні територій з багаторічною мерзлотою:
https://edtrochim.users.earthengine.app/view/permafrostsurfacewater

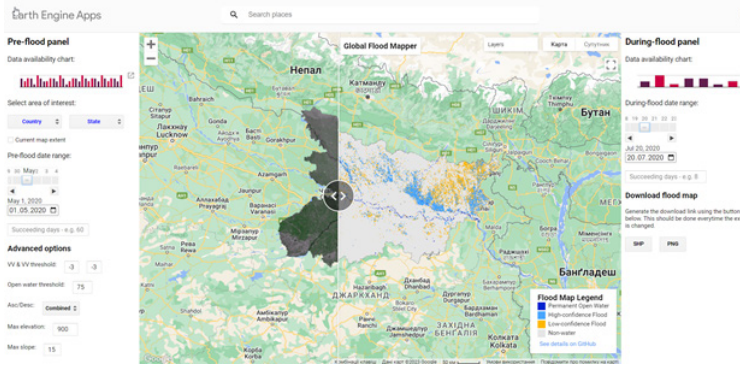


2. Вебзастосунок для статистичної оцінки захворюваності під час пандемії коронавірусу у 2020 р.:
https://gena.users.earthengine.app/view/corona-virus

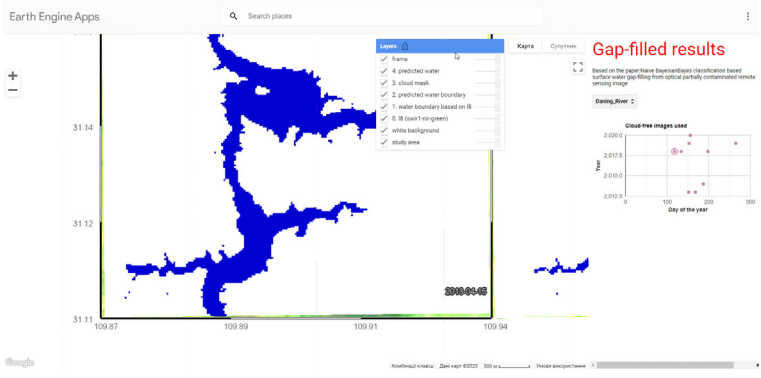
3. Карта щільності заселення територій світу:
https://gena.users.earthengine.app/view/nobody-lives-here



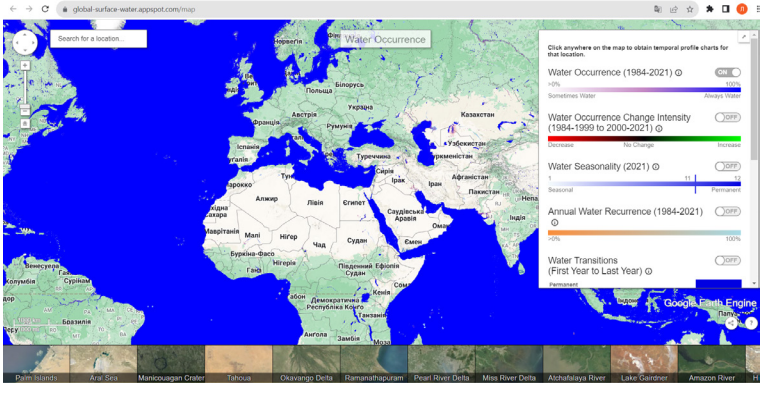
4. Вебзастосунок для відображення підтоплених територій:
https://pratyush-tripathy.users.earthengine.app/view/global-flood-mapper-advanced?fbclid=IwAR0cI_X8gKQPnGk1j61affCflXc-dNmT-ZQ6ijAiwzno3TCrBlqPXcsJjKc



5. Вебзастосунок для класифікації об’єктів поверхневих вод із використанням супутникових знімків низької якості:
https://1021403057.users.earthengine.app/view/gap-filled-results



6. Вебзастосунок для глобального моніторингу поверхневих вод: https://global-surface-water.appspot.com/map



Навчальне видання

Дистанційне зондування Землі: обробка та аналіз супутникових знімків на платформі Google Earth Engine

Навчальний посібник

Редагування *І. В. Братащук*
Верстання *О. А. Жупанська*
Дизайн обкладинки *Б. Л. Лісовський*

Формат 60×84 1/16. Папір офс. 80 г/м^2.
Друк цифровий. Ум. друк. арк. 6,74.
Наклад 300 прим.

Видавництво: Національний центр «Мала академія наук України»,
Кловський узвіз, буд. 8, м. Київ, 01021

Свідоцтво суб'єкта видавничої справи:
ДК № 6999 від 04.12.2019