Google Earth Engine kurzus 2019

A geoinformatikai adatbázisok kurzus keretén belül. Időpontok:

- *október 29.*
- ***november 5.***
- ***november 12.***

Gulácsi András, SZTE
https://github.com/SalsaBoy990

Google Earth Engine

- Earth Engine User Summit 2017: Welcoming Remarks by Rebecca Moore

Hogyan működik a Google Earth Engine?

1. Kliens kontra szerver
2. Kommunikáció a szerverrel JSON formátumban
3. Késleltetett végrehajtás
4. A lépték és a vetületek kezelése
5. Adatok importálása és exportálása
6. Earth Engine Apps, az eredmények publikálása
7. Moduláris szerkezet
8. UI komponensek és események

1. Kliens kontra szerver

- A szerveren található objektumokat **kliens oldali „proxy” objektumok (tárolók) manipulálásával** lehet változtatni.
- Minden „ee”-vel kezdődő dolog egy proxy objektum
- **Nem tartalmazzák magát az adatot** (csak a metaadatokat és a kép azonosítóját vagy ID-ját), mert ezek csupán a szerveren található objektumok kezelői
- Éppen ezért **a rajtuk végzett if-else kifejezések és for, while ciklusok nem működnek.** Nem lenne értelme sem, ha a mi számítógépünk dolgozna, hiszen éppen az a lényeg, hogy az internet felhő, a Google számítógépei megteszik ezt helyettünk.

1. Kliens kontra szerver

```
var kliensSztring = "2017-04-01";
print(typeof kliensSztring); //=> sztring
// Az ee.Date() konstruktorát meghívjuk.
var szerverSztring = ee.Date(kliensSztring);
print(
    "Ez egy EE objektum?",
    (szerverSztring instanceof ee.ComputedObject)
); //=> igaz
```

2. rész: Kommunikáció a szerverrel JSON formátumban

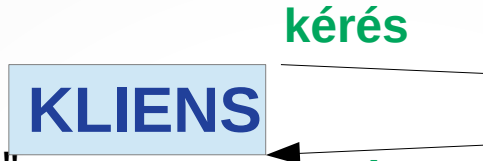
- A Google Earth Engine Kódszerkesztő és a szerver közötti kommunikáció JSON-ba csomagoltan történik:
- Kérést küldünk a szerver számára, hogy az végezze el az általunk megszabott műveletek sorozatát az adatbázisban található általunk meghatározott adatokra.
- A szerver ezt fogadja és elvégzi a feladatot, és JSON-ban válaszol (az eredményeket beleágyazza).

2. rész: Kommunikáció a szerverrel JSON formátumban

```
// Az adatok kiválasztása: SRTM domborzatmodell
var image = ee.Image('CGIAR/SRTM90_V4');
// Hozzáadunk 10-et a domborzatmodell minden cellájához
var operation = image.add(10);
// Kiírjuk a konzolra a kérésünket szövegként (JSON)
print(operation.toString());
// A szerver válasza (JSON)
print(operation);
```

2. rész: Kommunikáció a szerverrel JSON formátumban

```
ee.Image({
  "type": "Invocation",
  "arguments": {
    "image1": {
      "type": "Invocation",
      "arguments": {
        "id": "CGIAR/SRTM90_V4"
      },
      "functionName": "Image.load"
    },
    "image2": {
      "type": "Invocation",
      "arguments": {
        "value": 10
      },
      "functionName":
"Image.constant"
    }
  },
  "functionName": "Image.add"
})
```




```
{
  "type": "Image",
  "bands": [
    {
      "id": "elevation",
      "data_type": {
        "type": "PixelType",
        "precision": "int",
        "min": -32758,
        "max": 32777
      },
      "crs": "EPSG:4326",
      "crs_transform": [
        0.0008333333535119891,
        0,
        -180,
        0,
        -0.0008333333535119891,
        60
      ]
    }
  ]
}
```

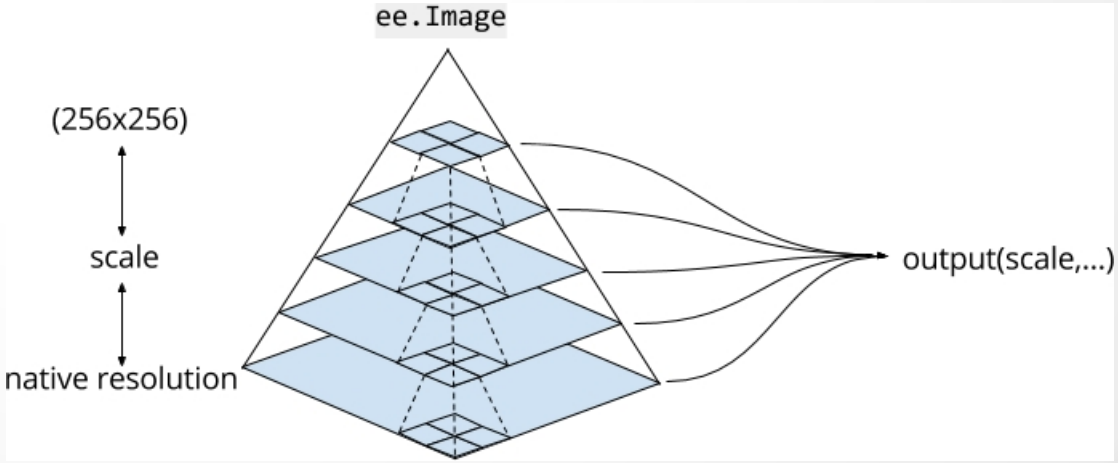
3. Késleltetett végrehajtás

- Ha egy szkriptet írunk, akkor a kód nem közvetlenül fut le a Google szerverein.
- Ehelyett, a kliens könyvtár átkódolja a szkriptet JSON objektumok sorozatává, majd ezeket elküldi a Google-nek, és válaszra vár.
- Semmi sem lesz elküldve a Google számára feldolgozásra, ha explicite nem adunk rá utasítást. Fölöslegesen nem dolgoz fel semmit.
- A **print()** vagy a **Map.addLayer()** utasítás viszont elegendő a kérés elküldéséhez

4. A lépték és a vetületek kezelése

- A **Map.addLayer()** segítségével a térképen kirajzolt kép eltérő bemenetekből készül a nagyítási szint és a térképnézet határaitól függően.
- Képpiramisok: minden egyes cella értéke egy adott piramisszinten az alatta levő szint 2x2-es blokkjának átlagértéke.

A GEE azt a szintet választja, ami a legközelebb esik az általunk megadott léptékhez



4. A lépték és a vetületek kezelése

Nagyítási szint	Pixelméret az Egyenlítőnél
11	76 m
12	38 m
13	19 m
14	9,6 m
15	4,8 m
16	2,4 m

***Nagyon FONTOS a LÉPTÉK megadása**, ha elemzést végzünk vagy statisztikát számolunk. Az adatok eredeti léptékében dolgozzunk (pl. Landsat 30 méter)!*

4. A lépték és a vetületek kezelése

- A Google a térképek megjelenítéséhez a **MERCATOR vetületet** használja (*WGS 84 / Pseudo-Mercator, EPSG:3857*), így aztán a képpiramis megfelelő szintjén, a megjelenítést megelőzően, vetületi transzformációra kerül sor (röptében).
- Ha lehetséges, akkor **célszerű elkerülni az egyéb vetületi transzformációkat.** Meg kell hagyni az adatokat eredeti vetületükben!
- **Az EOV nem támogatott.** Ne is próbáld használni, mert rossz lesz az eredmény!

5. Adatok importálása és exportálása

- Több lehetőség áll rendelkezésünkre az exportálásra: az adatainkat *GeoTIFF-be/HD videóba* menthetjük
  - a **Google Drive**-ra,
  - a Google felhő-alapú tárolóegységre (**Google Cloud Storage**) vagy
  - az **Assets** mappánkba (150 GB limit). Ez utóbbi helyre a saját GeoTIFF raszterállományainkat is feltölthetünk.
- A vektoros adatokat **Shape fájl** formátumban ajánlott feltölteni be, amit betölthetünk a szkriptünkbe az azonosítója segítségével. A készített idősorok adatai is lementhetők CSV-be (pontosvesszővel tagolt értékek).

6. Earth Engine Apps, az eredmények publikálása

- Az Earth Engine Apps egy dinamikus és nyilvánosan hozzáférhető felhasználói felület, amivel a Google Earth Engine elemzéseink eredményeit publikálhatjuk. Kísérleti stádiumban.
- MODIS NDDI alapú 8 napos aszálymonitoring egy Homokhátság-Délvidék mintaterületen (Water@Risk közös magyar-szerb projekt, tanulmány). Mindig a legújabb 8-napos aszálytérképet mutatja, késleltetéssel
- https://gulandras90.users.earthengine.app/view/modis-nddi-aszalymonitoring

7. Moduláris szerkezet

- Nem egy fájlban van az összes kódunk, hanem **több fájlban felosztva, amiket be tudunk importálni a fő szkriptünkbe**
- Ez hasonlítható a C nyelvben található #include direktívához
- Így csatoljuk a forrásfájlunkhoz a standard könyvtár előre megírt függvényeit: #include <stdlib.h>
- A JavaScript esetén is használhatunk több fájlt, amit a **require()** segítségével csatolhatunk.

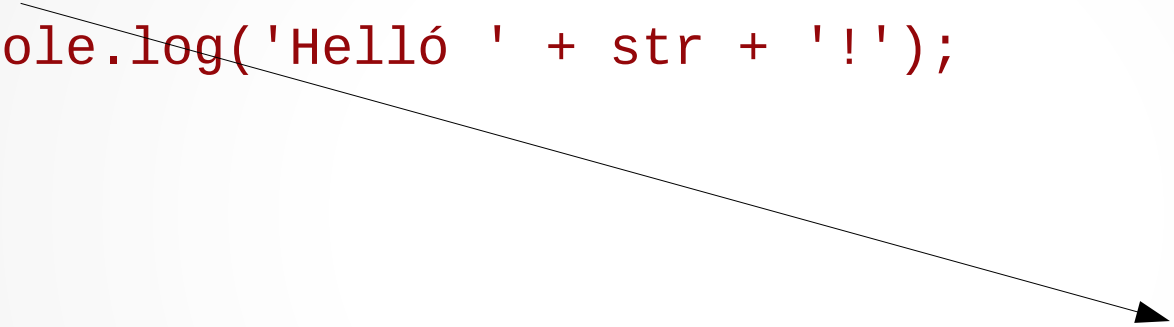
Modulok

- Kis programrészek, amik valamilyen részfeladatot végeznek el:
- Például: adatbekérés a felhasználótól, spektrális indexek számolása, felhőszűrés elvégzése, osztályozás elvégzése
- Sokkal átláthatóbbá teszi a programunkat, mert nem ömlesztve van több száz sor egyben, a bug-okat könnyebb javítani, a karbantarthatóbb kód
- Minden egyes modul rendelkezik egy **exports** nevű objektummal. Ehhez adunk hozzá tulajdonságokat
- Privát változók, amik csak a modulon belül élnek (kívülről nem hozzáférhetők)

Modulok

module.js

```
var str = 'világ';
exports.greet = function () {
  console.log('Helló ' + str + '!');
}
```



mainScript.js

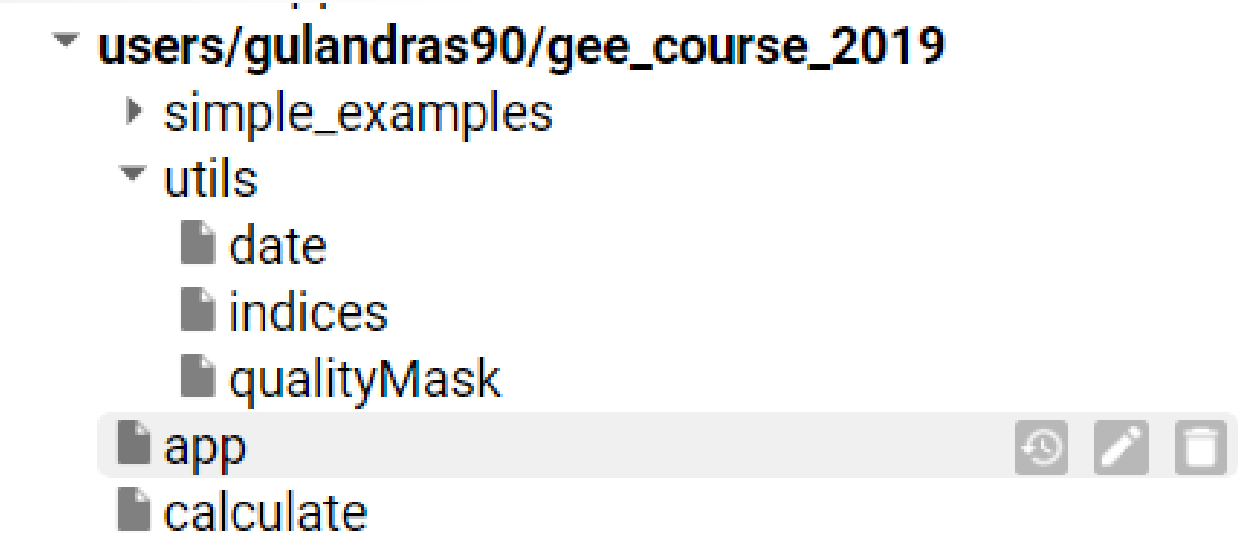
```
var sayHello = require('./module');
sayHello.greet();
// => Helló világ!
console.log(sayHello.str);
// => undefined
```

Példa: Date modul

```
// This function gets user input to construct date range for filtering
exports.getDateRange = function(year) {
  year = parseInt(year, 10);
  return { // returns object, invoke ee.Date() constructor
    start: ee.Date(year + '-01-01'),
    finish: ee.Date(year + '-12-31'),
  };
};
```

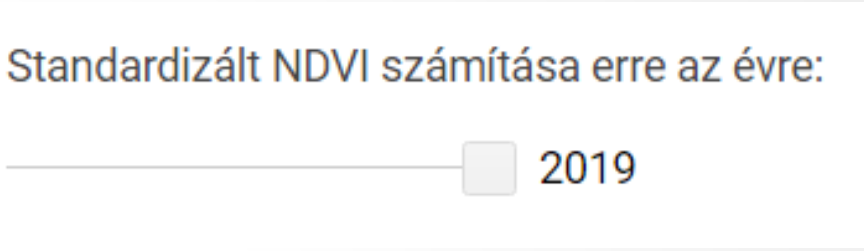
Date modul betöltése

- `// Import date function`
- `var date = require('users/gulandras90/gee_course_2019:utils/date');`



8. UI komponensek és események (events)

```
// Year selection slider
var yearSlider = ui.Slider({
  min: 2013,
  max: new Date().getFullYear(),
  value: new Date().getFullYear(),
  step: 1,
  onChange: function () {
    var year = yearSlider.getValue(); // get selected year value
    var dateRange = date.getDateRange(year);
  ... }
});
```



Event Emitter

- 1 569 API elem

Server-side proxy objects and their methods

ee.Algorithms	ee.Feature	ee.PixelType
ee.Array	ee.FeatureCollection	ee.Projection
ee.Blob	ee.Filter	ee.Reducer
ee.Classifier	ee.Geometry	ee.String
ee.Clusterer	ee.Image	ee.Terrain
ee.ConfusionMatrix	ee.ImageCollection	ee.apply
ee.Date	ee.Join	ee.call
ee.DateRange	ee.Kernel	ee.data
ee.Dictonary	ee.List	ee.initialize
ee.ErrorMargin	ee.Number	ee.reset

Client-side objects and methods

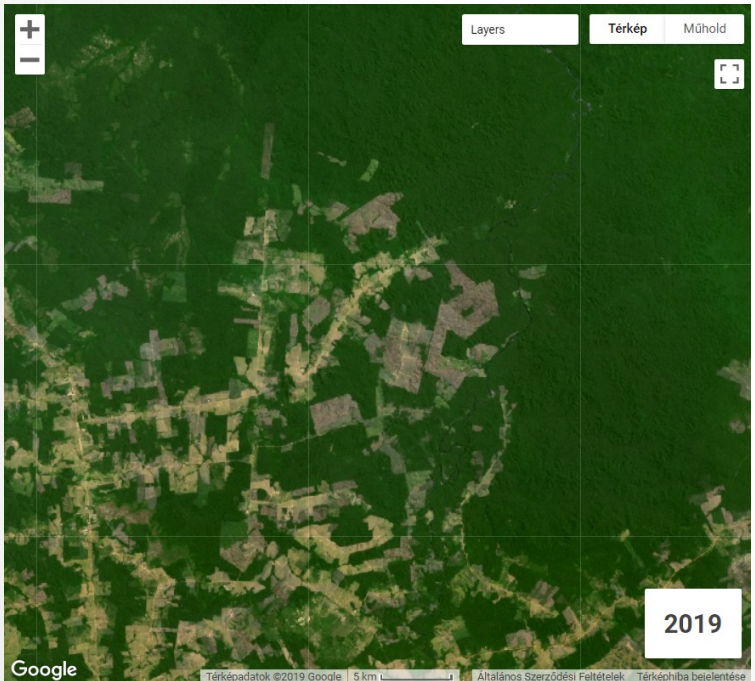
Export
Map
exports
print
require

User interface elements

ui.Button	ui.Slider
ui.Chart	ui.SplitPanel
ui.Checkbox	ui.Textbox
ui.DateSlider	ui.Thumbnail
ui.Label	ui.data
ui.Map	ui.root
ui.Panel	ui.util
ui.Select	

Záróprojekt: Amazonas NDVI Monitoring

- Ezt a kis webalkalmazást fogjuk elkészíteni:
- https://gulandras90.users.earthengine.app/view/amazonas-ndvi-monitoring-test



erdőirtások