CMPT 384 – Information Visualization

Creating Dashboard using

VGA (Visualization for Geospatial Analysis)

Framework

Course Instructor: Debajyoti Mondal

Lab Tutorial Instructor : **Arman Heydari**(arman.heydari@usask.ca)

What is VGA?

- A framework built to **combine visualizations and data processing** avoiding the effort of building visualization dashboard from scratch
  - You can **grab visualizations** built in d3 (and others) and **create a plugin** around them
  - You can write some **data processing** code and create a **plugin** based on that
  - You can use those **plugins to create visualization** dashboard
  - The vga framework will create visualizations simply as **json files**

We will now go through some tasks – Use a good code editor to make the coding easier --- you can use basic editor such as notepad++ that highlights the tags, or more advanced editors such as visual studio code

Start the server

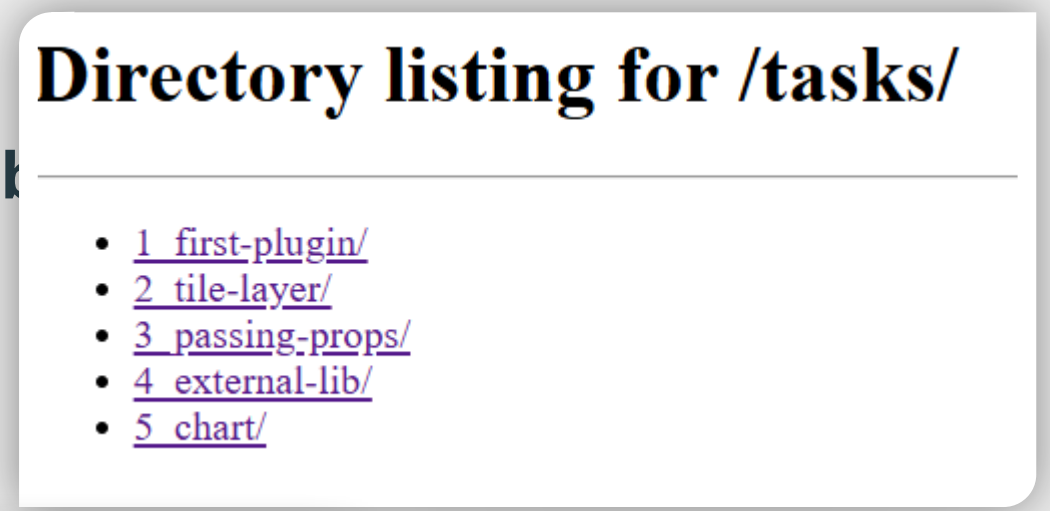
```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\> python -m SimpleHTTPServer 8888
C:\Users\jyoti\AppData\Local\Programs\Python\Python37-32\python.exe: No
module named SimpleHTTPServer

C:\> python -m http.server 8888
Serving HTTP on 0.0.0.0 port 8888 (http://0.0.0.0:8888/) ...
```

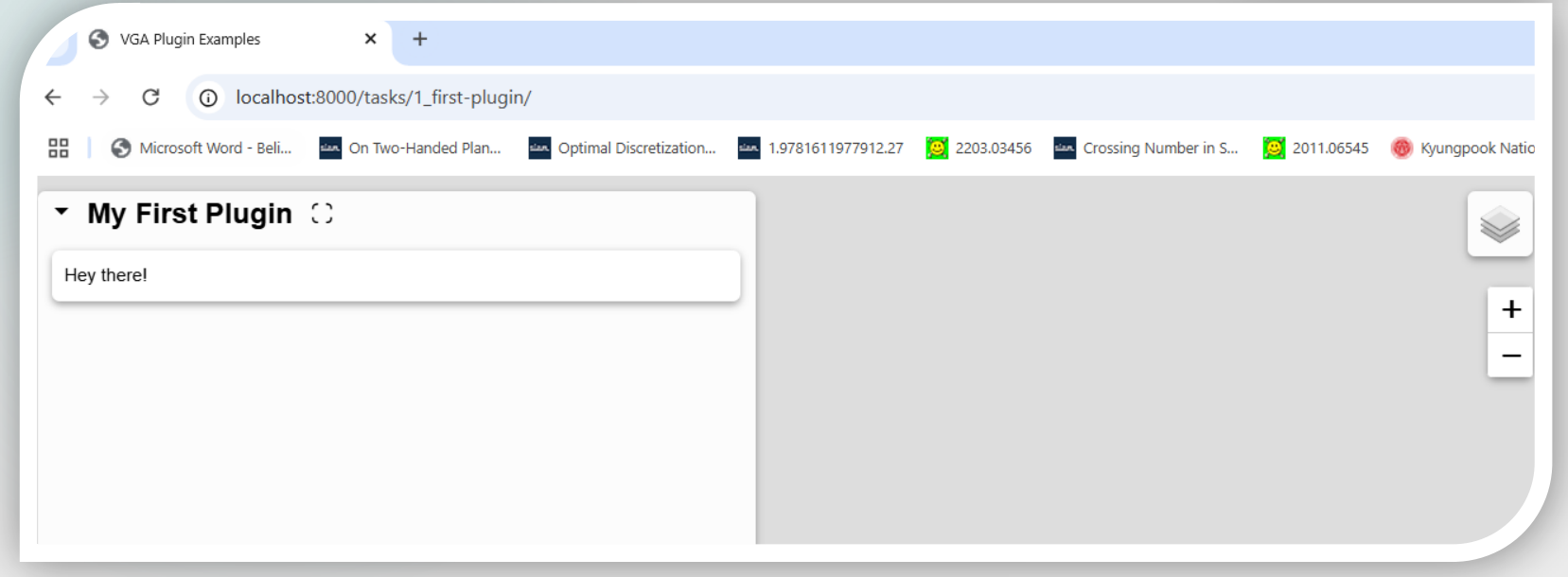
TASK 1

Creating Visualization Dashboard

- Navigate to 1_first-plugin/
- **This will show up an empty map dashb**

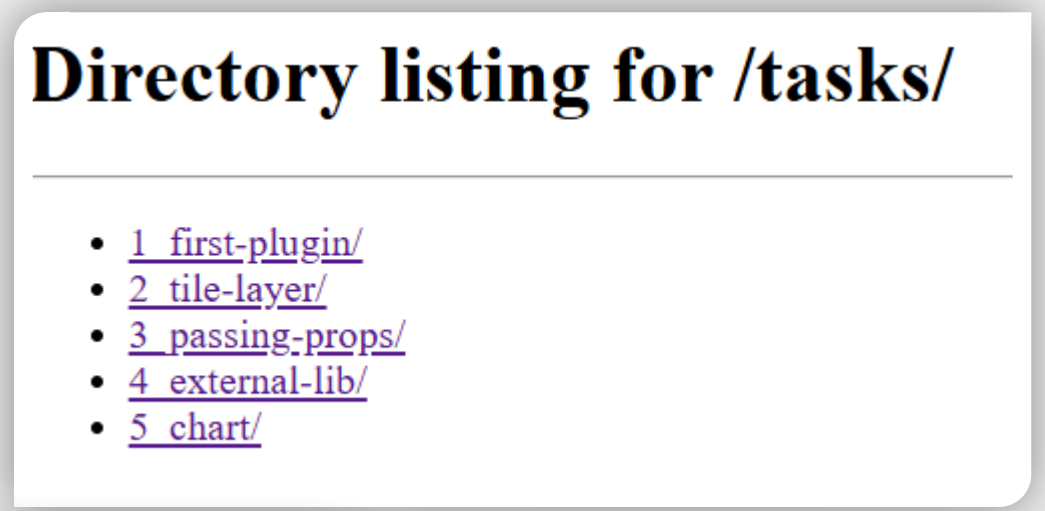
  with some text 'Hey there!'

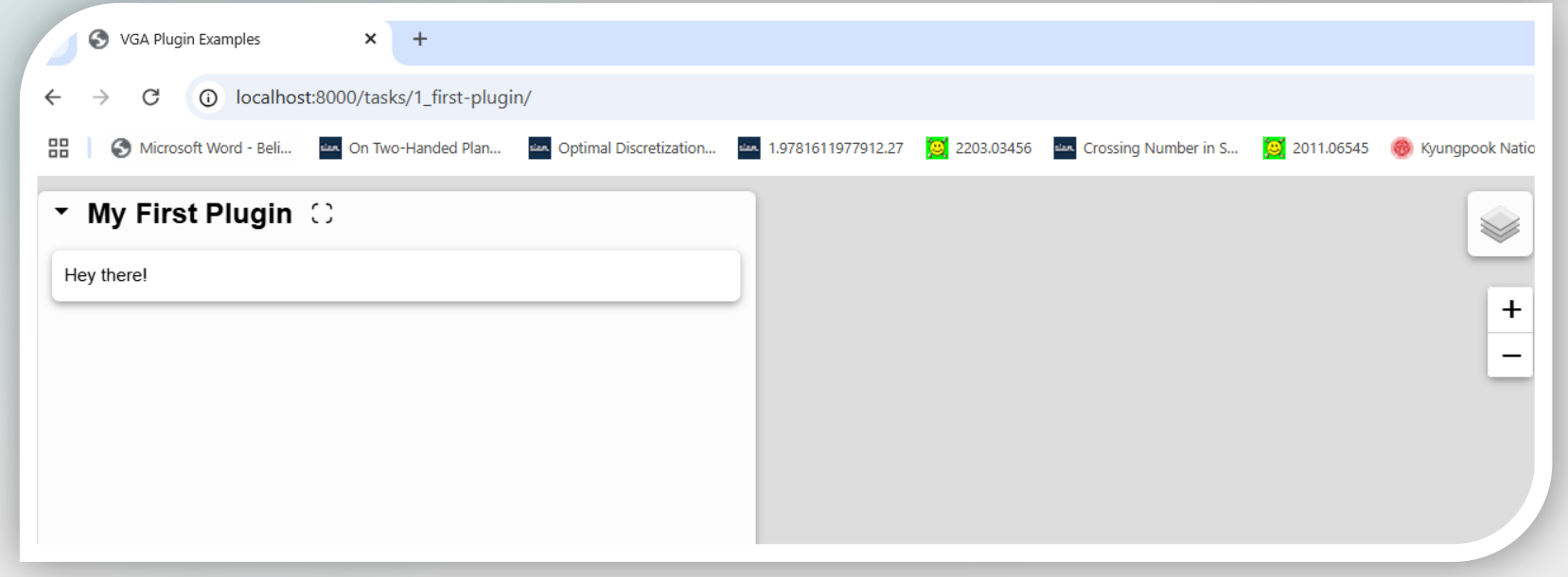




Creating Visualization Dashboard

- This is being created by creating a .json file and passing it to the vga framework





Understanding the code

- 1_first-plugin/index.html

```html
<html lang="en">
  <head>
    <base href="." />
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>VGA Plugin Examples</title>
    <script type="module" src="../../lib/vga-core.js"></script>
    <script type="module" defer>
      const visHost = document.querySelector("vga-core");
      const config = await fetch("./config.json").then((response) =>
        response.json()
      );
      visHost.config = config;
    </script>
  </head>
  <body>
    <vga-core></vga-core>
  </body>
```

Import the vga library

Create an instance of vga

try to fetch the config file over the local server

Pass the config (.json) file to vga

The vga library will write the code here for you

Understanding the code

- 1_first-plugin/index.html

```html
<html lang="en">
  <head>
    <base href="." />
    <meta charset="UTF-8"
    <meta http-equiv="X-UA
    <meta name="viewport"
    <title>VGA Plugin Exam
    <script type="module"
    <script type="module"
      const visHost = docu
      const config = awai
        response.json()
      );
      visHost.config = co
    </script>
  </head>
  <body>
    <vga-core></vga-core>
  </body>
```

Where is this config file?

- config.json
- index.html
- my-first.plugin.js
- README.md

```json
{
  "view": {
    "center": [0, 0],
    "zoom": 3
  },
  "imports": {
    "my-first-plugin": "./my-first.plugin.js"
  },
  "plugins": [
    {
      "import": "my-first-plugin",
      "container": "sidebar"
    }
  ]
}
```

Import the vga library

Create an instance of vga

try to fetch the config file over the local server

Pass the config (.json) file to vga

The vga library will write the code here for you

Understanding the code

- 1_first-plugin/index.html

```
<html lang="en">
  <head>
    <base href="." />
    <meta charset="UTF-8"
    <meta http-equiv="X-UA
    <meta name="viewport"
    <title>VGA Plugin Exam
    <s
    <s
    </
  </head>
  <body>
    <vga-core></vga-core>
  </body>
```

```
config.json
index.html
my-first.plugin.js
README.md
```

```
ntent="IE=edge" />
device-width, initial-scale=1.0" />
```

```
      "container": "sidebar"
    }
  ]
}
```

Where is this config file?

What should we do?

- Understand the functions of .js
- Understand .json
- Understand the changes in dashboard once you change .js or .json

Import the vga library

reate an instance of vga

ry to fetch the config file over the local server

ass the config (.json) file to vga

The vga library will write the code here for you

Understanding the code

- config.json

```
{
  "view": {
    "center": [0, 0],
    "zoom": 3
  },
  "imports": {
    "my-first-plugin": "./my-first.plugin.js"
  },
  "plugins": [
    {
      "import": "my-first-plugin",
      "container": "sidebar"
    }
  ]
}
```

These are map properties. You can change this to [56,-106] for centering the map on Canada but nothing will change as a map is not currently loaded

This is the way of importing the .js plugin.

What you define when you are importing .js should be the same here.

Whatever you render in plugin, should be visualized in “sidebar”. There are two other options "hidden" or "main“ that would come up later. Change and see what happens.

The text in purple are vga tags and this should not be changed (except for the string that you define when importing .js)

If changes in viz does not show up after you change something in the plugin, then remember to hard reload, or disable cache. Check

The Plugin .js code

```
export default class extends HTMLElement {
    /** This is a mandatory method to be implemented that returns the header of the plugin to be shown. */
    obtainHeaderCallback = () => `My First Plugin`;
    constructor() {
        super();
        /*
        Here `this` is the HTML element itself, we use the `attachShadow` fun
        The returned node can be used as our plugin's UI container.
        */
        const container = this.attachShadow({ mode: "open" });
        /*
        We can directly assign its `innerHTML`
        */
        // TODO 1: Modify the `innerHTML` to `"Hello World!"`
        container.innerHTML = "Hey there!!";
        /*
        We can also create a HTML element and append to it.
        */
        // TODO 2: Uncomment the following line
        // this.#renderButton(container);
        // TODO 3: Try adding something else that your want

    }
    #renderButton(container) {
        const button = document.
        button.innerText = "Click me!";
        button.addEventListener("click", () => alert("Button clicked."));
        container.append(button);
    }
}
```

Assume all these to be default to create a container where everything of this plugin will be rendered.

Whatever you write here should show up in the sidebar (assuming your .json file mentions “sidebar”)

You can write a javascript function and call it here. Whatever you render will show up in the sidebar

But how do I write this code? I only know d3 and very basics of javascript.

If changes in viz does not show up after you change something in the plugin, then remember to hard reload, or disable cache. Check

The Plugin .js code

```
export default class extends HTMLElement {
    /** This is a mandatory method to be implemented that returns the header of the plugin to be shown. */
    obtainHeaderCallback = () => `My First Plugin`;
    constructor() {
        super();
        /*
        Here `this` is the HTML element itself, we use the `attachShadow` fun
        The returned node can be used as our plugin's UI container.
        */
        const container = this.attachShadow({ mode: "open" });
        /*
        We can directly assign its `innerHTML`
        */
        // TODO 1: Modify the `innerHTML` to `"Hello World!"`
        container.innerHTML = "Hey there!!";
        /*
        We can also create a HTML element and append to it.
        */
        // TODO 2: Uncomment the following line
        // this.#renderButton(container);
        // TODO 3: Try adding something else that your want
    }
}
```

Assume all these to be default to create a container where everything of this plugin will be rendered.

Whatever you write here should show up in the sidebar (assuming your .json file mentions “sidebar”)

You can write a javascript function and call it here. Whatever you render will show up in the sidebar

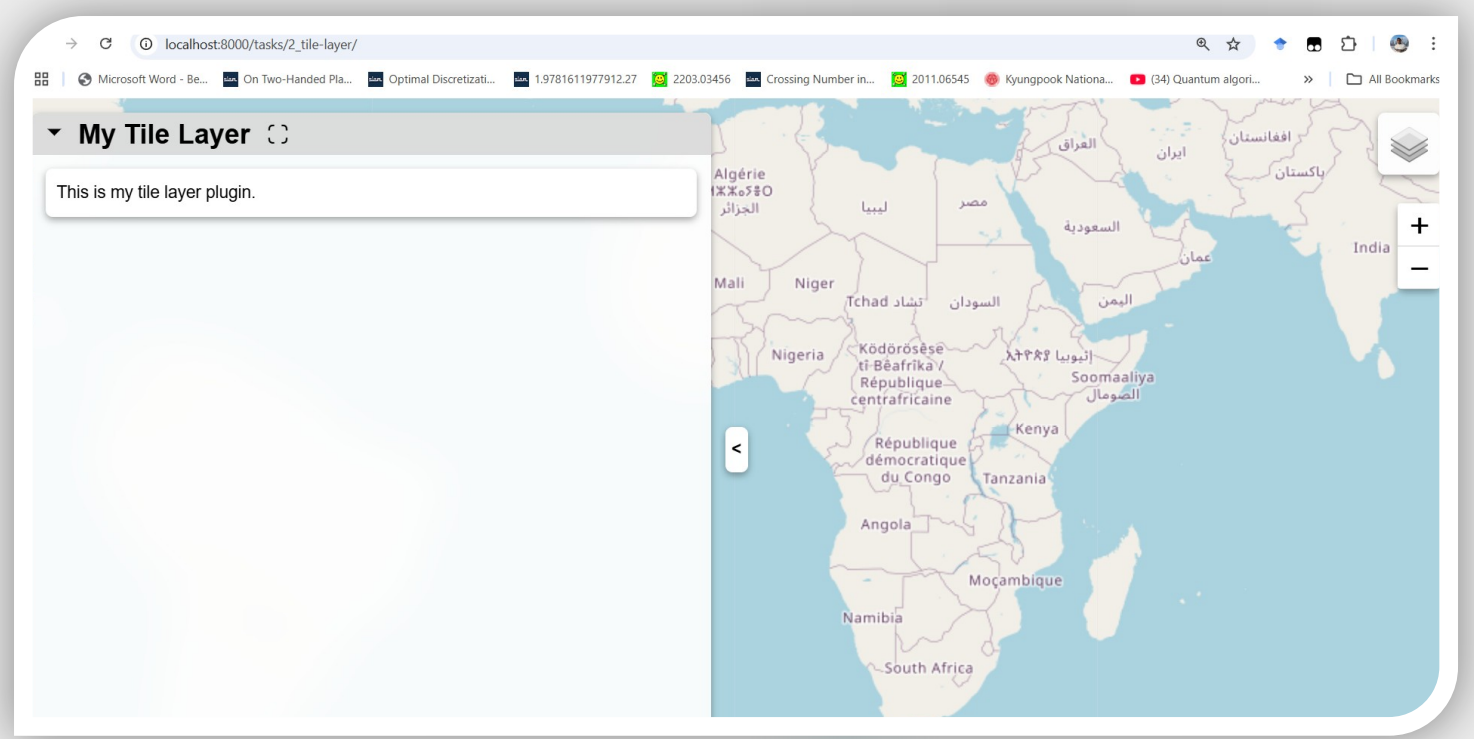
If you want to write a d3 function to create circles, this will NOT work yet but we will show how to do it

var svg = d3.select("body").append("svg").attr("width", 200).attr("height", 200);
svg.append('circle').attr('cx', 100).attr('cy', 100).attr('r', 50).attr('stroke', 'black’);

TASK 2

The html and .json files are the same as TASK1. The only change is in the .js file

- Navigate to 2_tile-layer/ and edit the .js file as follows.
- // TODO 1: Uncomment the following line
  this.#createAndAddATileLayerIntoMap();
- **This will show up a map**



If changes in viz does not show up after you change something in the plugin, then remember to hard reload, or disable cache. Check

Understanding the code

- config.json

```
{
  "view": {
    "center": [0, 0],
    "zoom": 3
  },
  "imports": {
    "my-first-plugin": "./my-first.plugin.js"
  },
  "plugins": [
    {
      "import": "my-first-plugin",
      "container": "sidebar"
    }
  ]
}
```

These are map properties. You now can change this to [56,-106] for centering the map on Canada. You can also change this to other [lat, long]
https://dwtkns.com/pointplotter/

You can also try to change zoom – typically between 1 to 12

If changes in viz does not show up after you change something in the plugin, then remember to hard reload, or disable cache. Check

Task2: Plugin .js code

- There is a popular map rendering library called **leaflet**
- **VGA** is built on top of leaflet and you can access all functionalities of leaflet by using **this.leaflet**
- **Here we use the leaflet to create the map**

```
hostFirstLoadedCallback() {
    // TODO 1: Uncomment the following line
     this.#createAndAddATileLayerIntoMap();
}
#createAndAddATileLayerIntoMap() {
    // This creates a Leaflet tile layer
    const tileLayer = this.leaflet?.tileLayer(
        "https://tile.openstreetmap.org/{z}/{x}/{y}.png",
        {
            attribution:
                '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors',
        },
    );
    // This adds the tile layer into the map
    this.addMapLayerDelegate?.(
        tileLayer,
        "My Tile Layer", // This is label of the layer that would be shown in the layer control
        "base-layer", // This is type of the layer (base-layer or overlay)
        true, // This is to determine whether it would be active by default
    );
}
```

OpenStreetMap is a free, open map database updated and maintained by a community of volunteers via open collaboration, which can be accessed in this link. We also add proper attribution.

You can use the addMapLayerDelegate function to add the map and you can specify the location using three parameters: title, type of layer, active or not

Change to overlay and see that on the top right corner you will have a checkbox to select the layer

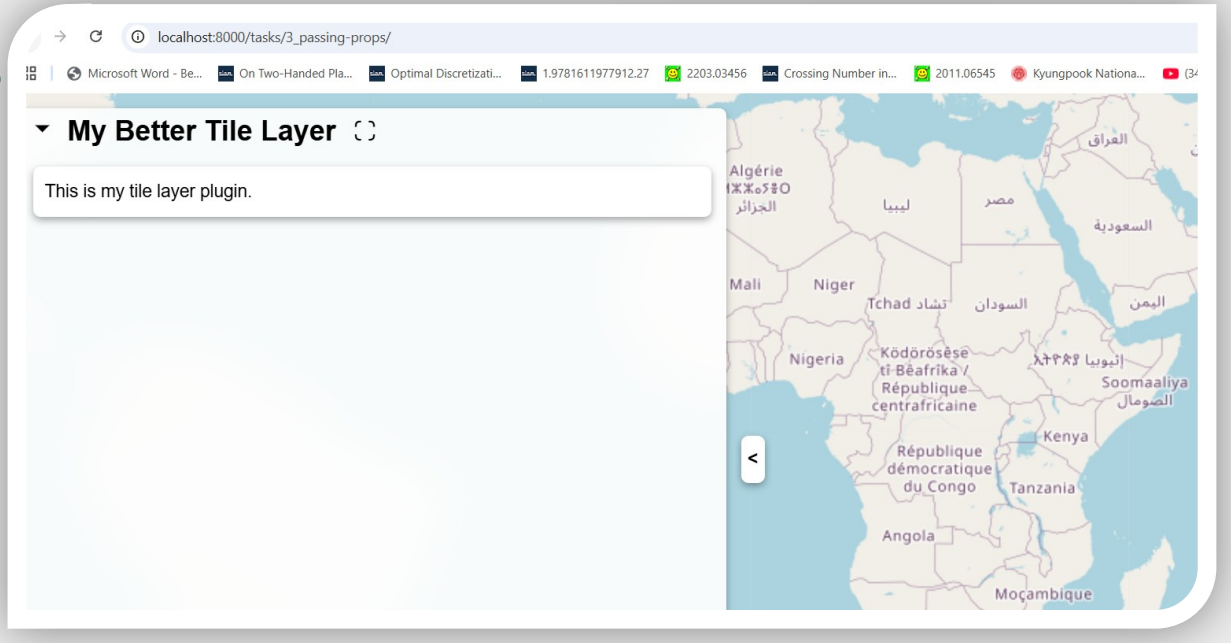
Making this false avoids rendering the layer unless selected by user

If changes in viz does not show up after you change something in the plugin, then remember to hard reload, or disable cache. Check

TASK 3

This task shows how you can specify properties in .json file and access them in .js file

Navigate to 3_passing-props/
The output will be the s



If changes in viz does not show up after you change something in the plugin, then remember to hard reload, or disable cache. Check

We now provide all properties in .json file and use them in .js file as needed

```
{
  "view": {
    "center": [0, 0],
    "zoom": 3
  },
  "imports": {
    "my-better-tile-layer": "./my-better-tile-layer.plugin.js"
  },
  "plugins": [
    {
      "import": "my-better-tile-layer",
      "container": "sidebar",
      "props": {
        "layerName": "World_Imagery",
        "layerType": "base-layer",
        "active": true,
        "urlTemplate": "https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}"
        "options": {
          "attribution": "Source: Esri, Maxar, Earthstar Geographics, and the GIS User Community"
        }
      }
    }
  ]
}
```

You can also use

"World_Physical_Map", "NatGeo_World_Map", "World_Street_Map“

This time we are using map provided by arcgis

If changes in viz does not show up after you change something in the plugin, then remember to hard reload, or disable cache. Check

We now provide all properties in .json file and use them in .js file as needed

Previous plugin .js

```
this.addMapLayerDelegate?.(
    tileLayer,
    "My Tile Layer",
    "base-layer",
    true,
);
```

New plugin .js

```
this.addMapLayerDelegate?.(
    tileLayer,
    layerName,
    layerType,
    true,
);
```

.json file

```
    "props": {
      "layerName": "World_Imagery",
      "layerType": "base-layer",
      "active": true,
      "urlTemplate": "https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}"
      "options": {
        "attribution": "Source: Esri, Maxar, Earthstar Geographics, and the GIS User Community"
      }
    }
  }
]
}
```

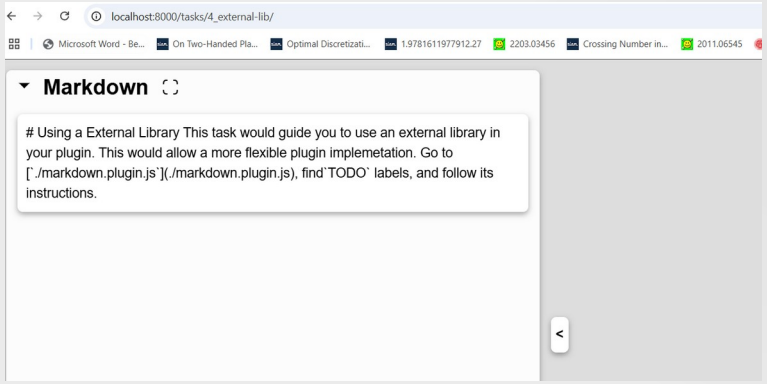
If changes in viz does not show up after you change something in the plugin, then remember to hard reload, or disable cache. Check

TASK 4

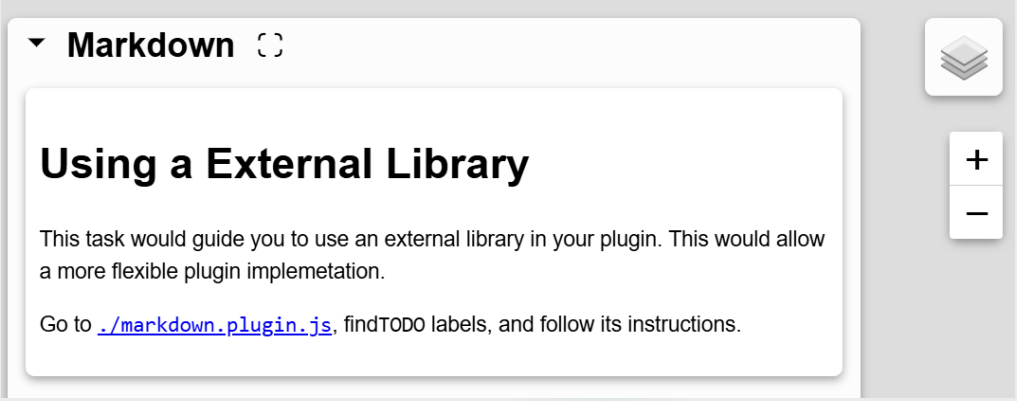
VGA helps to integrate external library. Here we incorporating an external text parsing library

https://www.npmjs.com/package/marked

Navigate to 4_external-lib/
The output will show a long text



We will pass the text
to the parser library
to get



If changes in viz does not show up after you change something in the plugin, then remember to hard reload, or disable cache. Check

We have already specified our .js file plugin in the .json file

```
"imports": {
  "markdown": "./markdown.plugin.js"
},
"plugins": [
  {
    "import": "markdown",
    "container": "sidebar"
  }
```

Let's now examine the plugin

```
import { marked } from "https://esm.sh/marked@^15";
```

Importing the external library

```
    const MARKDOWN_URL = "./README.md";
    this.#renderMarkdown(
        new URL(MARKDOWN_URL, document.baseURI),
        container,
    );
}

async #renderMarkdown(url, container) {
    if (!url) {
        return;
    }
    const content = await this.#fetchMarkdownContent(url);
    container.innerHTML = marked.parse(content);
}

async #fetchMarkdownContent(url) {
    return await fetch(url).then((res) => res.text());
}
```

We have written a javascript function that will read a file from the url specified and render the content in help container

Here we are fetching the file from the server

Here we are using the marked.parse function provided by the parser to reformat the content

If changes in viz does not show up after you change something in the plugin, then remember to hard reload, or disable cache. Check

Instead of hard-coding the file path in the plugin code, you can specify it in the .json file and access the property in .js file

```
import { marked } from "https://esm.sh/marked@^15";
```

```
    const MARKDOWN_URL = "./README.md";
    this.#renderMarkdown(
        new URL(MARKDOWN_URL, document.baseURI),
        container,
    );
}

async #renderMarkdown(url, container) {
    if (!url) {
        return;
    }
    const content = await this.#fetchMarkdownContent(url);
    container.innerHTML = marked.parse(content);
}

async #fetchMarkdownContent(url) {
    return await fetch(url).then((res) => res.text());
}
```

```
{
  "view": {
    "center": [0, 0],
    "zoom": 3
  },
  "imports": {
    "markdown": "./markdown.plugin.js"
  },
  "plugins": [
    {
      "import": "markdown",
      "container": "sidebar",
      "props": {
        "myurl":  "./README.md"
      }
    }
  ]
}
```

```
    const MARKDOWN_URL = this.myurl;
    this.#renderMarkdown(
        new URL(MARKDOWN_URL, document.baseURI),
        container,
    );
```

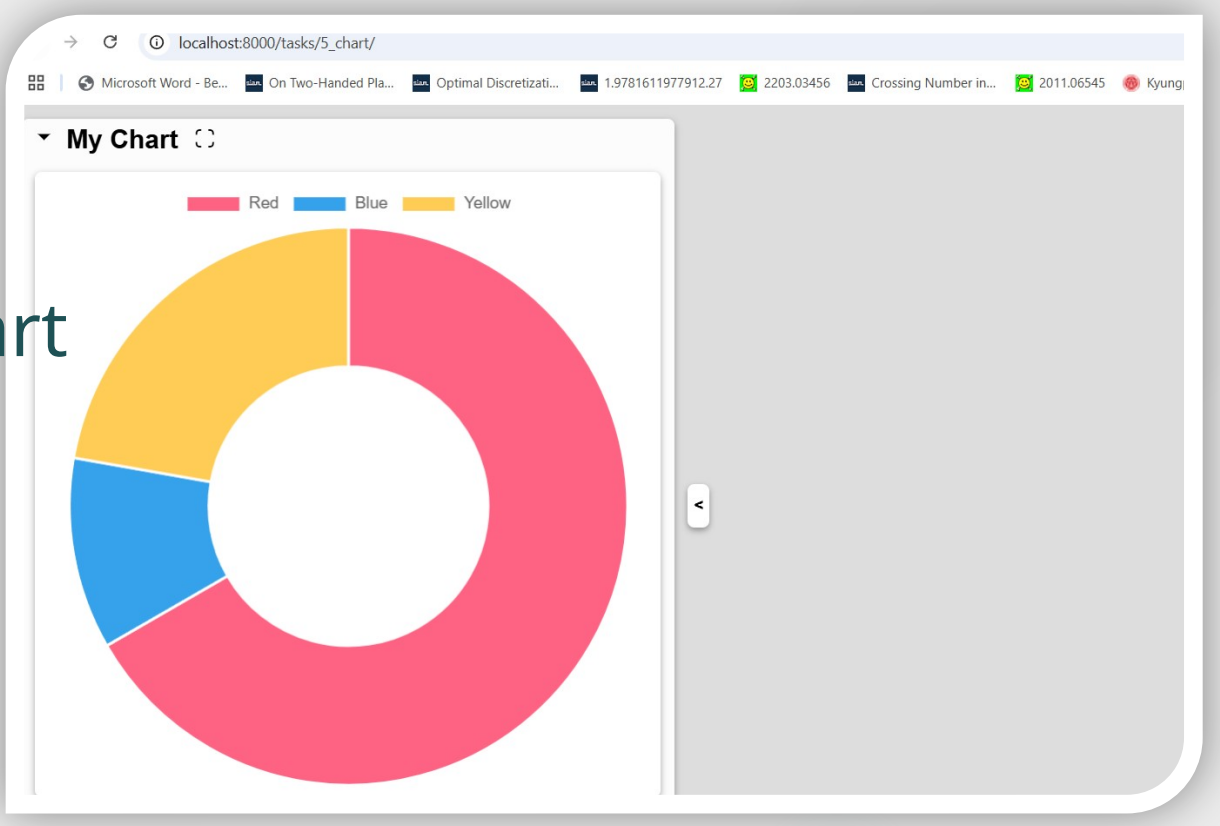
If changes in viz does not show up after you change something in the plugin, then remember to hard reload, or disable cache. Check

TASK 5

VGA helps to integrate external library. Here we incorporating an external chart building library

https://www.chartjs.org/docs/

Navigate to 5_chart
The output will show a chart



If changes in viz does not show up after you change something in the plugin, then remember to hard reload, or disable cache. Check

First import the chart library
Create a and append it to the html container
Append a canvas to the
Render a chart using the importe

```
import Chart from "https://esm.sh/chart.js@^4/auto";
export default class extends HTMLElement {
    #mainContainer;
    obtainHeaderCallback = () => `My Chart`;
    constructor() {
        super();
        const container = this.attachShadow({ mode: "open" });
        // create a `div` as the main chart container
        this.#mainContainer = document.createElement("div");
        this.#mainContainer.style.position = "relative";
        this.#mainContainer.style.height = "100%";
        this.#mainContainer.style.width = "100%";
        container.append(this.#mainContainer);
    }
    hostFirstLoadedCallback() {
        // create a `canvas` inside the main container
        const chartCanvas = document.createElement("canvas");
        this.#mainContainer.append(chartCanvas);
        // render the chart
        this.#renderChart(chartCanvas);
    }
```

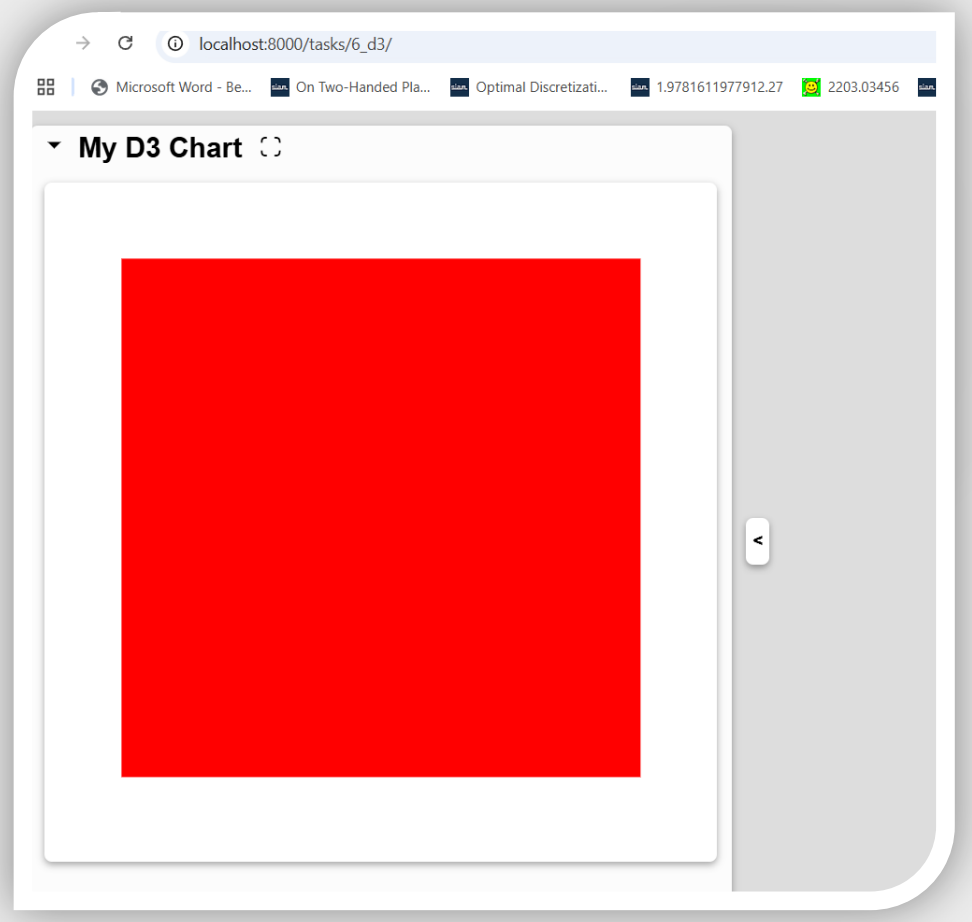
```
    #renderChart(canvas) {
        const data = {
            labels: [
                "Red",
                "Blue",
                "Yellow",
            ],
            datasets: [{
                label: "My First Dataset",
                data: [300, 50, 100],
                backgroundColor: [
                    "rgb(255, 99, 132)",
                    "rgb(54, 162, 235)",
                    "rgb(255, 205, 86)",
                ],
                hoverOffset: 4,
            }],
        };
        const config = {
            type: "doughnut",
            data: data,
        };
        new Chart(canvas, config);
    }
}
```

A code created following the external chart library specification

If changes in viz does not show up after you change something in the plugin, then remember to hard reload, or disable cache. Check

TASK 6

VGA helps to integrate external library. Here we incorporating an external d3 library



Navigate to 6_d3
The output will be a rectangle drawn using d3

If changes in viz does not show up after you change something in the plugin, then remember to hard reload, or disable cache. Check

First import the chart library
Create a and append it to the html container
Append a canvas to the
Render a chart using the imported lib

```
import * as d3 from "https://esm.sh/d3@^7";

export default class extends HTMLElement {
    #mainContainer;

    obtainHeaderCallback = () => `My D3 Chart`;

    constructor() {
        super();
        const container = this.attachShadow({ mode: "open" });

        // create a `div` as the main chart container
        this.#mainContainer = document.createElement("div");
        this.#mainContainer.style.position = "relative";
        this.#mainContainer.style.height = "100%";
        this.#mainContainer.style.width = "100%";
        container.append(this.#mainContainer);
    }

    hostFirstLoadedCallback() {
        // render the chart
        this.#renderChart(this.#mainContainer);
    }
```

A code created following the d3 library specification

```
    #renderChart(mainContainer) {
        const svg = d3.select(mainContainer)
            .append("svg")
            .attr("height", "100%")
            .attr("width", "100%")
            .attr("viewBox", "0 0 100 100")
            .attr("preserveAspectRatio", "xMidYMid meet");
        svg.append("rect")
            .attr("x", 10)
            .attr("y", 10)
            .attr("width", 80)
            .attr("height", 80)
            .attr("fill", "red");
    }
}
```

If changes in viz does not show up after you change something in the plugin, then remember to hard reload, or disable cache. Check