# Make your own GIS service

## Summer School on Digital Humanities

Web site: https://bit.ly/dt4h-gis

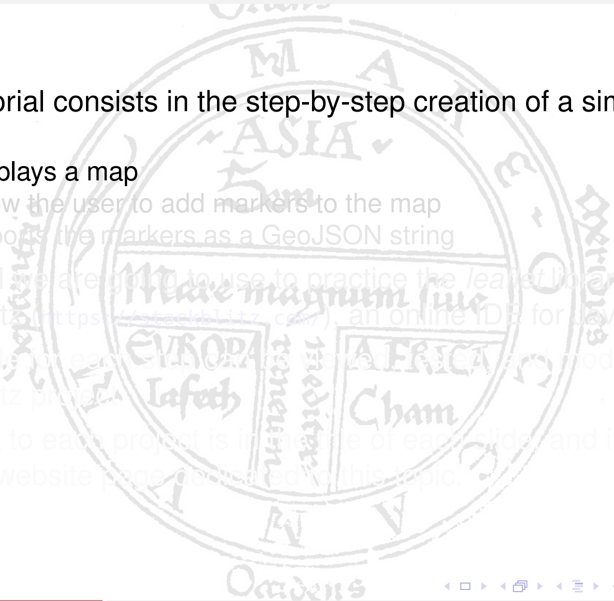
Augusto Ciuffoletti

13 giugno 2025

## How to proceed

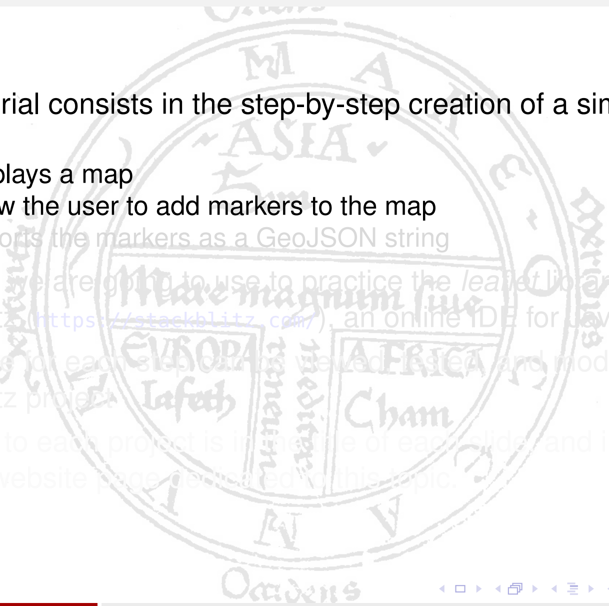
- The tutorial consists in the step-by-step creation of a simple app that:
  - Displays a map
  - Allow the user to add markers to the map
  - Exports the markers as a GeoJSON string
- The tool we are going to use to practice the *leaflet* library is Stackblitz (`https://stackblitz.com/`), an online IDE for JavaScript
- The code for each step can be viewed, tested and modified as a Stackblitz project
- The link to each project is in the title of each slide, and in the course website page dedicated to this topic.

## How to proceed



- The tutorial consists in the step-by-step creation of a simple app that:
    - Displays a map
    - Allow the user to add markers to the map
    - Exports the markers as a GeoJSON string
- The tool we are going to use to practice the *leaflet* library is Stackblitz (https://stackblitz.com/), an online IDE for JavaScript
- The code for each step can be viewed, tested and modified as a Stackblitz project
- The link to each project is in the title of each slide, and in the course website page dedicated to this topic.

## How to proceed



- The tutorial consists in the step-by-step creation of a simple app that:
  - Displays a map
  - Allow the user to add markers to the map
  - Exports the markers as a GeoJSON string
- The tool we are going to use to practice the *leaflet* library is Stackblitz (https://stackblitz.com/), an online IDE for JavaScript
- The code for each step can be viewed, tested, and modified as a Stackblitz project
- The link to each project is in the title of each slide, and in the course website page dedicated to this topic.

## How to proceed

- The tutorial consists in the step-by-step creation of a simple app that:
  - Displays a map
  - Allow the user to add markers to the map
  - Exports the markers as a GeoJSON string
- The tool we are going to use to practice the *leaflet* library is Stackblitz (`https://stackblitz.com/`), an online IDE for JavaScript
- The code for each step can be viewed, tested, and modified as a Stackblitz project
- The link to each project is in the title of each slide, and in the course website page dedicated to this topic.

## How to proceed

- The tutorial consists in the step-by-step creation of a simple app that:
  - Displays a map
  - Allow the user to add markers to the map
  - Exports the markers as a GeoJSON string
- The tool we are going to use to practice the *leaflet* library is Stackblitz (`https://stackblitz.com/`), an online IDE for JavaScript
- The code for each step can be viewed, tested, and modified as a Stackblitz project
- The link to each project is in the title of each slide, and in the course website page dedicated to this topic.

## How to proceed

- The tutorial consists in the step-by-step creation of a simple app that:
  - Displays a map
  - Allow the user to add markers to the map
  - Exports the markers as a GeoJSON string
- The tool we are going to use to practice the *leaflet* library is Stackblitz (https://stackblitz.com/), an online IDE for JavaScript
- The code for each step can be viewed, tested, and modified as a Stackblitz project
- The link to each project is in the title of each slide, and in the course website page dedicated to this topic.

## How to proceed

- The tutorial consists in the step-by-step creation of a simple app that:
  - Displays a map
  - Allow the user to add markers to the map
  - Exports the markers as a GeoJSON string
- The tool we are going to use to practice the *leaflet* library is Stackblitz (`https://stackblitz.com/`), an online IDE for JavaScript
- The code for each step can be viewed, tested, and modified as a Stackblitz project
- The link to each project is in the title of each slide, and in the course website page dedicated to this topic.

## Using the Stackblitz IDE

- Follow the project link for the first step
- In the right frame you see the preview of your service, showing a map
  - the URL on top of the frame is functional: try it!
- In the left frame there is the project content
- The selected file is shown in the center frame

## Using the Stackblitz IDE

- Follow the project link for the first step
- In the right frame you see the preview of your service, showing a map
  - the URL on top of the frame is functional: try it...
- In the left frame there is the project content
  - [illegible] describes the step
  - [illegible]
  - [illegible]
  - [illegible]
- The selected file is shown in the center frame
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]

## Using the Stackblitz IDE

- Follow the project link for the first step
- In the right frame you see the preview of your service, showing a map
  - the URL on top of the frame is functional: try it...
- In the left frame there is the project content
  - The *README.md* describes the step
  - The *index.html* is the HTML code for the page
  - The *index.js* file is the [illegible]
  - The other files [illegible]
- The selected file is shown in the center frame

## Using the Stackblitz IDE

- Follow the project link for the first step
- In the right frame you see the preview of your service, showing a map
  - the URL on top of the frame is functional: try it...
- In the left frame there is the project content
  - The *README.md* describes the step
  - The *index.html* is the HTML code for the page
  - The *index.js* file is the javascript code using the leaflet library
  - The other files are not of interest
- The selected file is shown in the center frame

## Using the Stackblitz IDE

- Follow the project link for the first step
- In the right frame you see the preview of your service, showing a map
  - the URL on top of the frame is functional: try it...
- In the left frame there is the project content
  - The *README.md* describes the step
  - The *index.html* is the HTML code for the page
  - The *index.js* file is the javascript code using the leaflet library
  - The other files are not of interest
- The selected file is shown in the center frame

## Using the Stackblitz IDE

- Follow the project link for the first step
- In the right frame you see the preview of your service, showing a map
  - the URL on top of the frame is functional: try it...
- In the left frame there is the project content
  - The *README.md* describes the step
  - The *index.html* is the HTML code for the page
  - The *index.js* file is the javascript code using the leaflet library
  - The other files are not of interest
- The selected file is shown in the center frame

## Using the Stackblitz IDE

- Follow the project link for the first step
- In the right frame you see the preview of your service, showing a map
  - the URL on top of the frame is functional: try it...
- In the left frame there is the project content
  - The *README.md* describes the step
  - The *index.html* is the HTML code for the page
  - The *index.js* file is the javascript code using the leaflet library
  - The other files are not of interest
- The selected file is shown in the center frame
  - You can edit the code [illegible]
    For instance [illegible] index.html and notice the preview [illegible]
    Your edits remain [illegible] you should register on Stackblitz

## Using the Stackblitz IDE

- Follow the project link for the first step
- In the right frame you see the preview of your service, showing a map
  - the URL on top of the frame is functional: try it...
- In the left frame there is the project content
  - The *README.md* describes the step
  - The *index.html* is the HTML code for the page
  - The *index.js* file is the javascript code using the leaflet library
  - The other files are not of interest
- The selected file is shown in the center frame
  - You can edit the code and see what happens
  - For instance, try to change the string in [illegible] index.html and notice the preview change
  - Your edits remain [illegible] you should register on Stackblitz

## Using the Stackblitz IDE

- Follow the project link for the first step
- In the right frame you see the preview of your service, showing a map
  - the URL on top of the frame is functional: try it...
- In the left frame there is the project content
  - The *README.md* describes the step
  - The *index.html* is the HTML code for the page
  - The *index.js* file is the javascript code using the leaflet library
  - The other files are not of interest
- The selected file is shown in the center frame
  - You can edit the code and see what happens
  - For instance, try to change the string in line 10 in index.html and notice the preview change
  - Your edits remain local. To save your project you should register on Stackblitz

## Using the Stackblitz IDE

- Follow the project link for the first step
- In the right frame you see the preview of your service, showing a map
    - the URL on top of the frame is functional: try it...
- In the left frame there is the project content
    - The *README.md* describes the step
    - The *index.html* is the HTML code for the page
    - The *index.js* file is the javascript code using the leaflet library
    - The other files are not of interest
- The selected file is shown in the center frame
    - You can edit the code and see what happens
    - For instance, try to change the string in line 10 in index.html and notice the preview change
    - Your edits remain local. To save your project you should register on Stackblitz

## Using the Stackblitz IDE

- Follow the project link for the first step
- In the right frame you see the preview of your service, showing a map
  - the URL on top of the frame is functional: try it...
- In the left frame there is the project content
  - The *README.md* describes the step
  - The *index.html* is the HTML code for the page
  - The *index.js* file is the javascript code using the leaflet library
  - The other files are not of interest
- The selected file is shown in the center frame
  - You can edit the code and see what happens
  - For instance, try to change the string in line 10 in index.html and notice the preview change
  - Your edits remain local. To save your project you should register on Stackblitz

## Using the Stackblitz IDE

- Follow the project link for the first step
- In the right frame you see the preview of your service, showing a map
  - the URL on top of the frame is functional: try it...
- In the left frame there is the project content
  - The *README.md* describes the step
  - The *index.html* is the HTML code for the page
  - The *index.js* file is the javascript code using the leaflet library
  - The other files are not of interest
- The selected file is shown in the center frame
  - You can edit the code and see what happens
  - For instance, try to change the string in line 10 in index.html and notice the preview change
  - Your edits remain local. To save your project you should register on Stackblitz

## Step 1: the background (project)

- The first step in our tutorial consists of using the *Leaflet* library to display an OpenStreetMap raster
- How to:
  - The reference to the library is in the *package-lock* file
  - In the HTML file:
    - [illegible]
    - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
    - [illegible]
    - [illegible]
  - [illegible]

## Step 1: the background (project)

- The first step in our tutorial consists of using the *Leaflet* library to display an OpenStreetMap raster
- How to:
  - The reference to the library is in the *package-lock* file
  - In the HTML file:
    - [illegible]
    - [illegible]
  - The index.js file contains the JavaScript code of our App
  - The capital L stands for the *Leaflet* class
  - So we create a map with two parameters
    - [illegible]
    - [illegible]
  - Next we add a background raster, which is *OpenStreetMap*

## Step 1: the background (project)

- The first step in our tutorial consists of using the *Leaflet* library to display an OpenStreetMap raster
- How to:
  - The reference to the library is in the *package-lock* file
  - In the HTML file:
    - a head element with the CSS for the *Leaflet* library
    - a div element for the map (its id is map)
  - The index.js file contains the JavaScript code of our App
  - The capital L stands for the *Leaflet* class
  - So we create a map with two parameters
    - [illegible]
    - [illegible]
  - Next we add a background raster, which is *OpenStreetMap*

## Step 1: the background (project)

- The first step in our tutorial consists of using the *Leaflet* library to display an OpenStreetMap raster
- How to:
  - The reference to the library is in the *package-lock* file
  - In the HTML file:
    - a `head` element with the CSS for the *Leaflet* library
    - a `div` element for the map (its `id` is `mapid`)
  - The index.js file contains the JavaScript code of our App
  - The capital `L` stands for the *Leaflet* class
  - So we create a map with two parameters
    - [illegible]
    - [illegible]
  - Next we add a background raster, which is *OpenStreetMap*

## Step 1: the background (project)

- The first step in our tutorial consists of using the *Leaflet* library to display an OpenStreetMap raster
- How to:
  - The reference to the library is in the *package-lock* file
  - In the HTML file:
    - a `head` element with the CSS for the *Leaflet* library
    - a `div` element for the map (its `id` is `mapid`)
  - The index.js file contains the JavaScript code of our App
  - The capital `L` stands for the *Leaflet* class
  - So we create a map with two parameters
    - [illegible]
    - [illegible]
  - Next we add a background raster, which is *OpenStreetMap*

## Step 1: the background (project)

- The first step in our tutorial consists of using the *Leaflet* library to display an OpenStreetMap raster
- How to:
  - The reference to the library is in the *package-lock* file
  - In the HTML file:
    - a `head` element with the CSS for the *Leaflet* library
    - a `div` element for the map (its `id` is `mapid`)
  - The index.js file contains the JavaScript code of our App
  - The capital `L` stands for the *Leaflet* class
  - So we create a map with two parameters
  - Next we add a background raster, which is *OpenStreetMap*

## Step 1: the background (project)

- The first step in our tutorial consists of using the *Leaflet* library to display an OpenStreetMap raster
- How to:
  - The reference to the library is in the *package-lock* file
  - In the HTML file:
    - a `head` element with the CSS for the *Leaflet* library
    - a `div` element for the map (its `id` is `mapid`)
  - The index.js file contains the JavaScript code of our App
  - The capital `L` stands for the *Leaflet* class
  - So we create a map with two parameters
    - [illegible]
    - A JavaScript [illegible] center and zoom level
  - Next we add a background raster, which is *OpenStreetMap*

## Step 1: the background (project)

- The first step in our tutorial consists of using the *Leaflet* library to display an OpenStreetMap raster
- How to:
  - The reference to the library is in the *package-lock* file
  - In the HTML file:
    - a `head` element with the CSS for the *Leaflet* library
    - a `div` element for the map (its `id` is `mapid`)
  - The index.js file contains the JavaScript code of our App
  - The capital `L` stands for the *Leaflet* class
  - So we create a map with two parameters
    - the id of the DOM element hosting the raster (our `mapid`)
    - A JavaScript object that describes position of map center and zoom level
  - Next we add a background raster, which is *OpenStreetMap*

## Step 1: the background (project)

- The first step in our tutorial consists of using the *Leaflet* library to display an OpenStreetMap raster
- How to:
  - The reference to the library is in the *package-lock* file
  - In the HTML file:
    - a `head` element with the CSS for the *Leaflet* library
    - a `div` element for the map (its `id` is `mapid`)
  - The index.js file contains the JavaScript code of our App
  - The capital `L` stands for the *Leaflet* class
  - So we create a map with two parameters
    - the id of the DOM element hosting the raster (our `mapid`)
    - A JavaScript object that describes position of map center and zoom level
  - Next we add a background raster, which is *OpenStreetMap*

## Step 1: the background (project)

- The first step in our tutorial consists of using the *Leaflet* library to display an OpenStreetMap raster
- How to:
  - The reference to the library is in the *package-lock* file
  - In the HTML file:
    - a `head` element with the CSS for the *Leaflet* library
    - a `div` element for the map (its `id` is `mapid`)
  - The index.js file contains the JavaScript code of our App
  - The capital `L` stands for the *Leaflet* class
  - So we create a map with two parameters
    - the id of the DOM element hosting the raster (our `mapid`)
    - A JavaScript object that describes position of map center and zoom level
  - Next we add a background raster, which is *OpenStreetMap*

## Step 1: the background (project)

- The first step in our tutorial consists of using the *Leaflet* library to display an OpenStreetMap raster
- How to:
  - The reference to the library is in the *package-lock* file
  - In the HTML file:
    - a `head` element with the CSS for the *Leaflet* library
    - a `div` element for the map (its `id` is `mapid`)
  - The index.js file contains the JavaScript code of our App
  - The capital `L` stands for the *Leaflet* class
  - So we create a map with two parameters
    - the id of the DOM element hosting the raster (our `mapid`)
    - A JavaScript object that describes position of map center and zoom level
  - Next we add a background raster, which is *OpenStreetMap*

## Step 1: the background (project)

- The first step in our tutorial consists of using the *Leaflet* library to display an OpenStreetMap raster
- How to:
  - The reference to the library is in the *package-lock* file
  - In the HTML file:
    - a `head` element with the CSS for the *Leaflet* library
    - a `div` element for the map (its `id` is `mapid`)
  - The index.js file contains the JavaScript code of our App
  - The capital `L` stands for the *Leaflet* class
  - So we create a map with two parameters
    - the id of the DOM element hosting the raster (our `mapid`)
    - A JavaScript object that describes position of map center and zoom level
  - Next we add a background raster, which is *OpenStreetMap*

## Step 1: Lab activity

- Browse the web to find the coordinates of a place at your choice as the center of the raster
- Modify/remove the zoom factor

IMPORTANT:

- relax: you **cannot** damage my repo (you'd need my credentials)
- you may *Fork* (button on top-left corner) a branch in a repo of your own (recommended not strictly needed)
- you can undo unsaved updates with Ctrl-z
- after forking and signing up you can save your work

## Step 2: show the coordinates (project)

- When the user clicks on the map an alert appears with the coordinates of the click
- How to:
    - The HTML file is identical, we added management of a click event in the JavaScript
    - We apply the on method to the map to catch click events

## Step 2: show the coordinates (project)

- When the user clicks on the map an alert appears with the coordinates of the click
- How to:
  - The HTML file is identical, we added management of a click event in the JavaScript
  - We apply the *on* method to the map to catch *click* events

## Step 2: show the coordinates (project)

- When the user clicks on the map an alert appears with the coordinates of the click
- How to:
  - The HTML file is identical, we added management of a click event in the JavaScript
  - We apply the *on* method to the map to catch *click* events
    - the first parameter is the name of the event we want to capture
    - the second parameter is a callback that takes the event description as a parameter
    - the callback displays [illegible] extracted from the event description
    - the event description is an object
    - we extract the [illegible]

## Step 2: show the coordinates (project)

- When the user clicks on the map an alert appears with the coordinates of the click
- How to:
  - The HTML file is identical, we added management of a click event in the JavaScript
  - We apply the *on* method to the map to catch *click* events
    - the first parameter is the name of the event we want to capture
    - the second parameter is a callback that takes the event description as a parameter
    - the callback displays an alert containing data extracted from the event descriptor `e`
    - the event descriptor is an object
    - we extract the `lat` and `lng` fields in the `latlng` field.

## Step 2: show the coordinates (project)

- When the user clicks on the map an alert appears with the coordinates of the click
- How to:
    - The HTML file is identical, we added management of a click event in the JavaScript
    - We apply the *on* method to the map to catch *click* events
        - the first parameter is the name of the event we want to capture
        - the second parameter is a callback that takes the event description as a parameter
        - the callback displays an alert containing data extracted from the event descriptor `e`
        - the event descriptor is an object
        - we extract the `lat` and `lng` fields in the `latlng` field.

## Step 2: show the coordinates (project)

- When the user clicks on the map an alert appears with the coordinates of the click
- How to:
  - The HTML file is identical, we added management of a click event in the JavaScript
  - We apply the *on* method to the map to catch *click* events
    - the first parameter is the name of the event we want to capture
    - the second parameter is a callback that takes the event description as a parameter
    - the callback displays an alert containing data extracted from the event descriptor `e`
    - the event descriptor is an object
    - we extract the `lat` and `lng` fields in the `latlng` field.

## Step 2: show the coordinates (project)

- When the user clicks on the map an alert appears with the coordinates of the click
- How to:
  - The HTML file is identical, we added management of a click event in the JavaScript
  - We apply the *on* method to the map to catch *click* events
    - the first parameter is the name of the event we want to capture
    - the second parameter is a callback that takes the event description as a parameter
    - the callback displays an alert containing data extracted from the event descriptor `e`
    - the event descriptor is an object
    - we extract the `lat` and `lng` fields in the `latlng` field.

## Step 2: show the coordinates (project)

- When the user clicks on the map an alert appears with the coordinates of the click
- How to:
  - The HTML file is identical, we added management of a click event in the JavaScript
  - We apply the *on* method to the map to catch *click* events
    - the first parameter is the name of the event we want to capture
    - the second parameter is a callback that takes the event description as a parameter
    - the callback displays an alert containing data extracted from the event descriptor `e`
    - the event descriptor is an object
    - we extract the `lat` and `lng` fields in the `latlng` field.

## Step 2: show the coordinates (project)

- When the user clicks on the map an alert appears with the coordinates of the click
- How to:
  - The HTML file is identical, we added management of a click event in the JavaScript
  - We apply the *on* method to the map to catch *click* events
    - the first parameter is the name of the event we want to capture
    - the second parameter is a callback that takes the event description as a parameter
    - the callback displays an alert containing data extracted from the event descriptor `e`
    - the event descriptor is an object
    - we extract the `lat` and `lng` fields in the `latlng` field.

## Step 2: Lab activity

- create a named <div> and write within the coordinates. Use

```
document.getElementById("myDiv").textContent = ...
```

## Step 3: collect coordinates (project)

- Each click on the map adds a marker, and their coordinates are shown on the page
- How to:
  - We add a *div* for the coordinates in the html
  - In the JavaScript we add to the event callback the creation of the new marker
    - its position is computed using the [illegible]
  - The coordinates are appended to the [illegible] element of the DOM

## Step 3: collect coordinates (project)

- Each click on the map adds a marker, and their coordinates are shown on the page
- How to:
  - We add a *div* for the coordinates in the html
  - In the JavaScript we add in the event callback the creation of the new marker
    - its position is computed using the latlng in the event descriptor
  - The coordinates are appended to the list in a `div` element of the DOM

## Step 3: collect coordinates (project)

- Each click on the map adds a marker, and their coordinates are shown on the page
- How to:
    - We add a *div* for the coordinates in the html
    - In the JavaScript we add to the event callback the creation of the new marker
        - Its position is computed using the `latlng` field in the event descriptor
    - The coordinates are appended to the list in a `div` element of the DOM

## Step 3: collect coordinates (project)

- Each click on the map adds a marker, and their coordinates are shown on the page
- How to:
  - We add a *div* for the coordinates in the html
  - In the JavaScript we add to the event callback the creation of the new marker
    - its position is computed using the `latlng` field in the event descriptor
  - The coordinates are appended to the list in a `div` element of the DOM

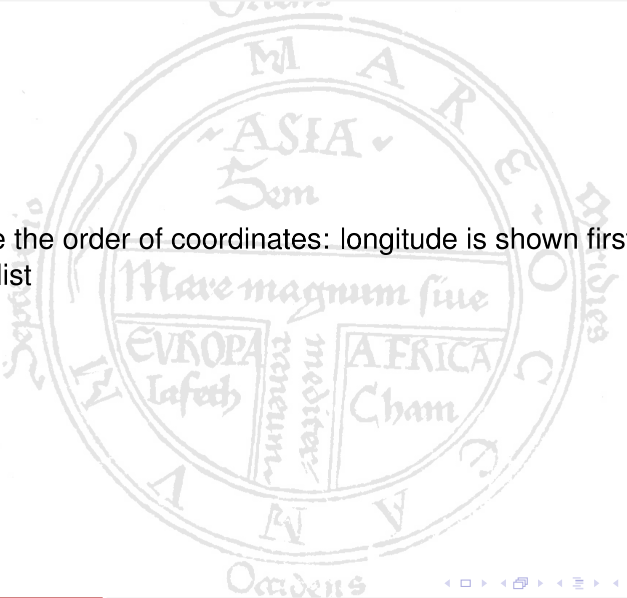
## Step 3: collect coordinates (project)

- Each click on the map adds a marker, and their coordinates are shown on the page
- How to:
  - We add a *div* for the coordinates in the html
  - In the JavaScript we add to the event callback the creation of the new marker
    - its position is computed using the `latlng` field in the event descriptor
  - The coordinates are appended to the list in a `div` element of the DOM

## Step 3: collect coordinates (project)

- Each click on the map adds a marker, and their coordinates are shown on the page
- How to:
  - We add a *div* for the coordinates in the html
  - In the JavaScript we add to the event callback the creation of the new marker
    - its position is computed using the `latlng` field in the event descriptor
  - The coordinates are appended to the list in a `div` element of the DOM

## Step 3: Lab activity



- Reverse the order of coordinates: longitude is shown first in the bottom list

## Step 4: enumerated markers (project)

- A progressive index is assigned to each new point
- The index is shown in the list and added as a *title* field in the marker definition
  - the title field is automatically displayed when the mouse hovers on the marker
- How to:

## Step 4: enumerated markers (project)

- A progressive index is assigned to each new point
- The index is shown in the list and added as a *title* field in the marker definition
  - the title field is automatically displayed when the mouse hovers on the marker
- How to:
  - HTML is the same
  - We add a new global variable in the JavaScript
  - The [illegible]
  - The value of [illegible]
  - The marker constructor [illegible] the marker [illegible]

## Step 4: enumerated markers (project)

- A progressive index is assigned to each new point
- The index is shown in the list and added as a *title* field in the marker definition
  - the title field is automatically displayed when the mouse hovers on the marker
- How to:
  - HTML is the same
  - We add a new global variable in the JavaScript
  - [illegible]
  - [illegible]
  - [illegible]

## Step 4: enumerated markers (project)

- A progressive index is assigned to each new point
- The index is shown in the list and added as a *title* field in the marker definition
    - the title field is automatically displayed when the mouse hovers on the marker
- How to:
    - HTML is the same
    - We add a new global variable n in the JavaScript
    - The event callback increments the variable each time it is run
    - The value of n is displayed on each line in the list
    - The marker constructor now takes a second parameter containing the marker options

## Step 4: enumerated markers (project)

- A progressive index is assigned to each new point
- The index is shown in the list and added as a *title* field in the marker definition
  - the title field is automatically displayed when the mouse hovers on the marker
- How to:
  - HTML is the same
  - We add a new global variable `n` in the JavaScript
  - The event callback increments the variable each time it is run
  - The value of `n` is displayed on each line in the list
  - The marker constructor now takes a second parameter containing the marker options

## Step 4: enumerated markers (project)

- A progressive index is assigned to each new point
- The index is shown in the list and added as a *title* field in the marker definition
  - the title field is automatically displayed when the mouse hovers on the marker
- How to:
  - HTML is the same
  - We add a new global variable `n` in the JavaScript
  - The event callback increments the variable each time it is run
  - The value of `n` is displayed on each line in the list
  - The marker constructor now takes a second parameter containing the marker options

## Step 4: enumerated markers (project)

- A progressive index is assigned to each new point
- The index is shown in the list and added as a *title* field in the marker definition
    - the title field is automatically displayed when the mouse hovers on the marker
- How to:
    - HTML is the same
    - We add a new global variable `n` in the JavaScript
    - The event callback increments the variable each time it is run
    - The value of `n` is displayed on each line in the list
    - The marker constructor now takes a second parameter containing the marker options

## Step 4: enumerated markers (project)

- A progressive index is assigned to each new point
- The index is shown in the list and added as a *title* field in the marker definition
  - the title field is automatically displayed when the mouse hovers on the marker
- How to:
  - HTML is the same
  - We add a new global variable `n` in the JavaScript
  - The event callback increments the variable each time it is run
  - The value of `n` is displayed on each line in the list
  - The marker constructor now takes a second parameter containing the marker options
    - among which the `title` option

## Step 4: enumerated markers (project)

- A progressive index is assigned to each new point
- The index is shown in the list and added as a *title* field in the marker definition
    - the title field is automatically displayed when the mouse hovers on the marker
- How to:
    - HTML is the same
    - We add a new global variable `n` in the JavaScript
    - The event callback increments the variable each time it is run
    - The value of `n` is displayed on each line in the list
    - The marker constructor now takes a second parameter containing the marker options
        - among which the `title` option

## Step 4: enumerated markers (project)

- A progressive index is assigned to each new point
- The index is shown in the list and added as a *title* field in the marker definition
  - the title field is automatically displayed when the mouse hovers on the marker
- How to:
  - HTML is the same
  - We add a new global variable `n` in the JavaScript
  - The event callback increments the variable each time it is run
  - The value of `n` is displayed on each line in the list
  - The marker constructor now takes a second parameter containing the marker options
    - among which the `title` option

## Step 4: Lab activity

- Configure the marker as draggable (ignore that the displayed coordinates become inconsistent)
  - hint: in the `marker` variable definition add a `draggable: true` property after the title, separated by a ","

## Step 5: all markers in an array (project)

- Record the markers in an array to have them accessible
  - in the previous steps the marker was a local variable in the callback
- How to:
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]

## Step 5: all markers in an array (project)

- Record the markers in an array to have them accessible
  - in the previous steps the marker was a local variable in the callback
- How to:
  - Create an array for the markers
  - Push markers in the array
  - mIndex corresponds to array length
    - [illegible]

## Step 5: all markers in an array (project)

- Record the markers in an array to have them accessible
  - in the previous steps the marker was a local variable in the callback
- How to:
  - Create an array for the markers
  - Push markers in the array
  - n index corresponds to array length

## Step 5: all markers in an array (project)

- Record the markers in an array to have them accessible
  - in the previous steps the marker was a local variable in the callback
- How to:
  - Create an array for the markers
  - Push markers in the array
  - n index corresponds to array length

## Step 5: all markers in an array (project)

- Record the markers in an array to have them accessible
  - in the previous steps the marker was a local variable in the callback
- How to:
  - Create an array for the markers
  - Push markers in the array
  - n index corresponds to array length
    - no need to increment it

## Step 5: all markers in an array (project)

- Record the markers in an array to have them accessible
  - in the previous steps the marker was a local variable in the callback
- How to:
  - Create an array for the markers
  - Push markers in the array
  - n index corresponds to array length
    - no need to increment it

## Step 5: all markers in an array (project)

- Record the markers in an array to have them accessible
  - in the previous steps the marker was a local variable in the callback
- How to:
  - Create an array for the markers
  - Push markers in the array
  - `n` index corresponds to array length
    - no need to increment it

## Step 5: Lab activity

- Create a button that fades-out the markers
- Hint
  - loop through all items in the array with a for loop

```
for (let m in markers) {...}
```

  - use the `setOpacity(0.5)` on each marker

## Step 6: all markers in a layer (project)

- Having all markers in a layer is more practical than in an array
- How to:
  - HTML is always the same
  - Replace the array with a *layerGroup* object added to the map (markers)
  - [illegible] layerGroup
  - Add [illegible]
  - The [illegible]
    [illegible]
  - See the [illegible]

## Step 6: all markers in a layer (project)

- Having all markers in a layer is more practical than in an array
- How to:
    - HTML is always the same
    - Replace the array with a *layerGroup* object added to the map (`markers`)
    - Replace the *push* operation with an *addLayer* applied to the layerGroup
    - Add a layer control to the map to toggle layer visibility
    - The control creation takes two object arguments
    - See the effect on the layers button top right in the map

## Step 6: all markers in a layer (project)

- Having all markers in a layer is more practical than in an array
- How to:
  - HTML is always the same
  - Replace the array with a *layerGroup* object added to the map (`markers`)
  - Replace the *push* operation with an *addLayer* applied to the layerGroup
  - Add a layer control too to the map to toggle layer visibility
  - The control creation takes two object arguments
    - [illegible]
    - [illegible]
  - See the effect on the layers button top right in the map

## Step 6: all markers in a layer (project)

- Having all markers in a layer is more practical than in an array
- How to:
  - HTML is always the same
  - Replace the array with a *layerGroup* object added to the map (`markers`)
  - Replace the *push* operation with an *addLayer* applied to the layerGroup
  - Add a layer control to the map to toggle layer visibility
  - The control creation takes two object arguments
  - See the effect on the layers button top right in the map

## Step 6: all markers in a layer (project)

- Having all markers in a layer is more practical than in an array
- How to:
  - HTML is always the same
  - Replace the array with a *layerGroup* object added to the map (`markers`)
  - Replace the *push* operation with an *addLayer* applied to the layerGroup
  - Add a layer control icon to the map to toggle layer visibility
  - The control creation takes two object arguments
    - One for the base layers (radio button, just one)
    - One for the overlay layers (multiple choice)
  - See the effect on the layers button top right in the map

## Step 6: all markers in a layer (project)

- Having all markers in a layer is more practical than in an array
- How to:
  - HTML is always the same
  - Replace the array with a *layerGroup* object added to the map (`markers`)
  - Replace the *push* operation with an *addLayer* applied to the layerGroup
  - Add a layer control icon to the map to toggle layer visibility
  - The control creation takes two object arguments
    - One for the base layers (radio button, just one)
    - One for the overlay layers (multiple choice)
  - See the effect on the layers button top right in the map

## Step 6: all markers in a layer (project)

- Having all markers in a layer is more practical than in an array
- How to:
  - HTML is always the same
  - Replace the array with a *layerGroup* object added to the map (`markers`)
  - Replace the *push* operation with an *addLayer* applied to the layerGroup
  - Add a layer control icon to the map to toggle layer visibility
  - The control creation takes two object arguments
    - One for the base layers (radio button, just one)
    - One for the overlay layers (multiple choice)
  - See the effect on the layers button top-right in the map

## Step 6: all markers in a layer (project)

- Having all markers in a layer is more practical than in an array
- How to:
  - HTML is always the same
  - Replace the array with a *layerGroup* object added to the map (`markers`)
  - Replace the *push* operation with an *addLayer* applied to the layerGroup
  - Add a layer control icon to the map to toggle layer visibility
  - The control creation takes two object arguments
    - One for the base layers (radio button, just one)
    - One for the overlay layers (multiple choice)
  - See the effect on the layers button top-right in the map

## Step 6: all markers in a layer (project)

- Having all markers in a layer is more practical than in an array
- How to:
  - HTML is always the same
  - Replace the array with a *layerGroup* object added to the map (`markers`)
  - Replace the *push* operation with an *addLayer* applied to the layerGroup
  - Add a layer control icon to the map to toggle layer visibility
  - The control creation takes two object arguments
    - One for the base layers (radio button, just one)
    - One for the overlay layers (multiple choice)
  - See the effect on the layers button top-right in the map

## Step 6: all markers in a layer (project)

- Having all markers in a layer is more practical than in an array
- How to:
    - HTML is always the same
    - Replace the array with a *layerGroup* object added to the map (`markers`)
    - Replace the *push* operation with an *addLayer* applied to the layerGroup
    - Add a layer control icon to the map to toggle layer visibility
    - The control creation takes two object arguments
        - One for the base layers (radio button, just one)
        - One for the overlay layers (multiple choice)
    - See the effect on the layers button top-right in the map

## Step 6: Lab activity

- move the `markers` *layerGroup* in the base layers. Any change?

## Step 7: GeoJSON serialization (project)

- It is handy to have a standard string representation of a piece of data (serialization)
  - e.g. to store the data in a file
- The GeoJSON representation can be easely transformed into a JSON string, and viceversa
- We want to print in the console the JSON string for our markers
- The `toGeoJSON` method converts the markers layer into a JavaScript object with the GeoJSON format

[illegible] we lose the [illegible]

- The `stringify` method serializes the object as a String object
- The string is finally recorded put on the display

## Step 7: GeoJSON serialization (project)

- It is handy to have a standard string representation of a piece of data (serialization)
  - e.g. to store the data in a file
- The GeoJSON representation can be easely transformed into a JSON string, and viceversa
- We want to print in the console the JSON string for our markers
- The toGeoJSON method converts the markers layer into a JavaScript object with the GeoJSON format
  - [illegible] we lose [illegible]
- The stringify method serializes the object as a String object
- The string is finally recorded out on the display

## Step 7: GeoJSON serialization (project)

- It is handy to have a standard string representation of a piece of data (serialization)
  - e.g. to store the data in a file
- The GeoJSON representation can be easely transformed into a JSON string, and viceversa
- We want to print in the console the JSON string for our markers
- The `toGeoJSON` method converts the markers layer into a JavaScript object with the GeoJSON format
  - alas, in [illegible] way we lose the [illegible]
- The `stringify` method serializes the object as a String object
- The string is finally recorded out on the display

## Step 7: GeoJSON serialization (project)

- It is handy to have a standard string representation of a piece of data (serialization)
  - e.g. to store the data in a file
- The GeoJSON representation can be easely transformed into a JSON string, and viceversa
- We want to print in the console the JSON string for our markers
- The `toGeoJSON` method converts the markers layer into a JavaScript object with the GeoJSON format
  - alas, in this way we lose the *title* field
- The `stringify` method serializes the object as a String object
- The string is finally recorded out on the display

## Step 7: GeoJSON serialization (project)

- It is handy to have a standard string representation of a piece of data (serialization)
  - e.g. to store the data in a file
- The GeoJSON representation can be easely transformed into a JSON string, and viceversa
- We want to print in the console the JSON string for our markers
- The `toGeoJSON` method converts the markers layer into a JavaScript object with the GeoJSON format
  - alas, in this way we lose the *title* field
- The `stringify` method serializes the object as a String object
- The string is finally recorded put on the display

## Step 7: GeoJSON serialization (project)

- It is handy to have a standard string representation of a piece of data (serialization)
  - e.g. to store the data in a file
- The GeoJSON representation can be easely transformed into a JSON string, and viceversa
- We want to print in the console the JSON string for our markers
- The `toGeoJSON` method converts the markers layer into a JavaScript object with the GeoJSON format
  - alas, in this way we lose the *title* field
- The `stringify` method serializes the object as a String object
- The string is finally recorded/put on the display

## Step 7: GeoJSON serialization (project)

- It is handy to have a standard string representation of a piece of data (serialization)
  - e.g. to store the data in a file
- The GeoJSON representation can be easely transformed into a JSON string, and viceversa
- We want to print in the console the JSON string for our markers
- The `toGeoJSON` method converts the markers layer into a JavaScript object with the GeoJSON format
  - alas, in this way we lose the *title* field
- The `stringify` method serializes the object as a String object
- The string is finally recorded put on the display

## Step 7: GeoJSON serialization (project)

- It is handy to have a standard string representation of a piece of data (serialization)
  - e.g. to store the data in a file
- The GeoJSON representation can be easely transformed into a JSON string, and viceversa
- We want to print in the console the JSON string for our markers
- The `toGeoJSON` method converts the markers layer into a JavaScript object with the GeoJSON format
  - alas, in this way we lose the *title* field
- The `stringify` method serializes the object as a String object
- The string is finally recorded put on the display

## Step 7: Lab activity

- Copy the generated GeoJSON and feed it to an online viewer, like `https://www.geometrymapper.com/`
- Is there any way to record the *title* field in the JSON string?
- Study the GeoJSON format in the console and find a solution