Make your own GIS service

Summer School on Digital Humanities

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

Augusto Ciuffoletti

13 giugno 2025

Make your own GIS service

- We have seen how open services provide many functionalities to
  - produce a *live* map, with web links and multimedia contents
  - share it with others
  - export the data across tools and other services
- However, we may have requirements that do not exactly match an existing service
- In that case we need to create our own web service
- The task is simplified by the existence of powerful open source library, *leaflet*
- In this concluding tutorial we search the source of this tool to understand its potential

Make your own GIS service

- We have seen how open services provide many functionalities to
  - produce a *live* map, with web links and multimedia contents
  - share it with others
  - export the data across tools and other services
- However we may have requirements that do not exactly match an existing service
- In that case we need to code our own web service
- The task is simplified by the existence of powerful open source library, *leaflet*
- In this concluding tutorial we sketch the source of this tool to understand its potential

Make your own GIS service

- We have seen how open services provide many functionalities to
  - produce a *live* map, with web links and multimedia contents
  - share it with others
  - export the data across tools and other services
- However we may have requirements that do not exactly match an existing service
- In that case we need to code our own web service
- The task is simplified by the existence of powerful open source library, *leaflet*
- In this concluding tutorial we search the source of this tool to understand its potential

Make your own GIS service

- We have seen how open services provide many functionalities to
  - produce a *live* map, with web links and multimedia contents
  - share it with others
  - export the data across tools and other services
- However we may have requirements that do not exactly match an existing service
- In that case we need to code our own web service
- The task is simplified by the existence of a powerful open source library, *leaflet*
- In this concluding tutorial we sketch the structure of this tool to understand its potential

Make your own GIS service

- We have seen how open services provide many functionalities to
  - produce a *live* map, with web links and multimedia contents
  - share it with others
  - export the data across tools and other services
- However we may have requirements that do not exactly match an existing service
- In that case we need to code our own web service
- The task is simplified by the existence of a powerful open source library, *leaflet*
- In this concluding tutorial we scratch the surface of this tool to understand its potential

Make your own GIS service

- We have seen how open services provide many functionalities to
  - produce a *live* map, with web links and multimedia contents
  - share it with others
  - export the data across tools and other services
- However we may have requirements that do not exactly match an existing service
- In that case we need to code our own web service
- The task is simplified by the existence of a powerful open source library, *leaflet*
- In this concluding tutorial we scratch the surface of this tool to understand its potential

Make your own GIS service

- We have seen how open services provide many functionalities to
  - produce a *live* map, with web links and multimedia contents
  - share it with others
  - export the data across tools and other services
- However we may have requirements that do not exactly match an existing service
- In that case we need to code our own web service
- The task is simplified by the existence of a powerful open source library, *leaflet*
- In this concluding tutorial we scratch the surface of this tool to understand its potential

Make your own GIS service

- We have seen how open services provide many functionalities to
  - produce a *live* map, with web links and multimedia contents
  - share it with others
  - export the data across tools and other services
- However we may have requirements that do not exactly match an existing service
- In that case we need to code our own web service
- The task is simplified by the existence of a powerful open source library, *leaflet*
- In this concluding tutorial we scratch the surface of this tool to understand its potential

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
  - Stores the data in the cloud (until September 2020)
- The tool we are going to use 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
  - Stores the layer in the cloud (until September 2025)
- The tool we are going to use to create 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
  - Stores the layer in the cloud (until September 2025)
- 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
  - Stores the layer in the cloud (until September 2025)
- 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
  - Stores the layer in the cloud (until September 2025)
- 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
  - Stores the layer in the cloud (until September 2025)
- 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
  - Stores the layer in the cloud (until September 2025)
- 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
  - Stores the layer in the cloud (until September 2025)
- 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
  - The
  - The
  - The
  - The
- 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
  - The
  - The
  - The
- 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 important
- 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 interesting
- 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 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:

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:
  - 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:

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 central instruction is `L.map('mapid')`
  - 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
  - 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 starts 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 captal 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
    - the id of the DOM element containing the map (our `mapid`)
    - A JavaScript object with the options: position (lat/lon 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:

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 function that receives the event description as

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 lon fields in a 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 a field containing data extracted from the event descriptor e
    - the event descriptor is an object
    - we extract the lat and lon fields in a 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 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:

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
  - 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
    - the position is computed using the `latLng` of 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:

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 [illegible]
  - The [illegible]
  - The [illegible]
  - We [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
  - The event callback increments the variable each time it is run
  - The value of n is displayed on each line of 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

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)

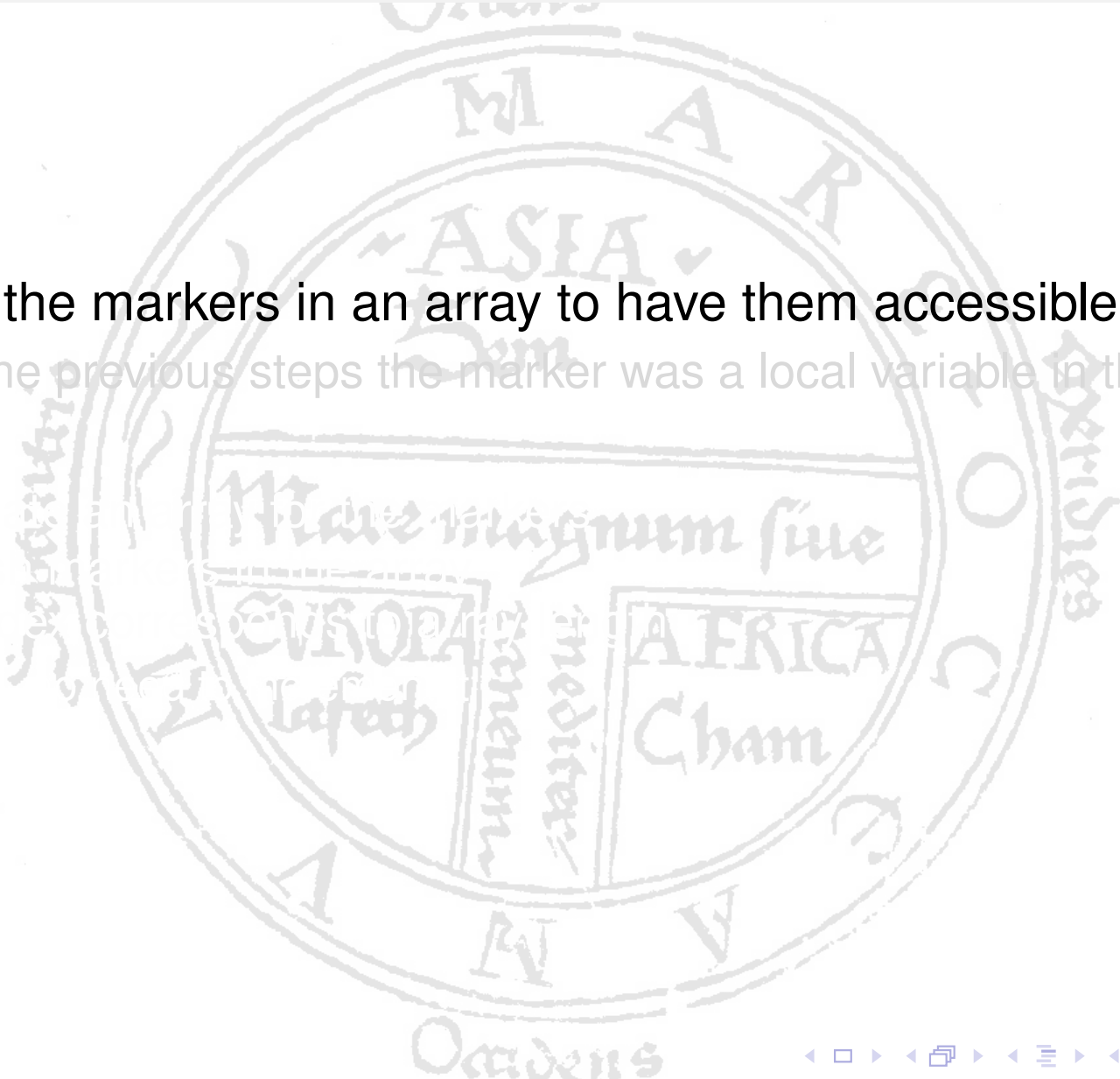
- A progressive index is assigned to each new point
- The index is shown in the list and added as a *title* field in the marker definition
  - the title field is automatically displayed when the mouse hovers on the marker
- How to:
  - HTML is the same
  - We add a new global variable `n` in the JavaScript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



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

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

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

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 to the map to toggle layer visibility
  - The coordinates are the two project arguments
  - See the effect on the layers button (top right) of 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 with the base layers
    - One with the overlays
  - See the effect on the layers button top right of 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 buttons)
    - One for the overlay layers (multiple choice)
  - See the effect on the layers button top right of 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
  - alas, in this way we lose the styling
- The `stringify` method serializes the object as a String object
- The string is finally recorded in the log

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 in the log

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 in the log

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 in the log

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 in the log

Step 7: Lab activity

- Is there any way to record the *title* field in the JSON string?
- Study the geoJSON format in the console and find a solution
- If needed see :
  - https://geojson.org/ for geojson syntax
  - https://leafletjs.com/reference-1.7.1.html#marker for the toGeoJSON method

Step 8-10: a map in the cloud

- We want to store our markers in the cloud
- The simplest option is to use a Key-Value service
  - a basic one is the one I implemented on MongoDB Atlas (just demonstration, not for public use)
- A *New* button in the interface allows the user to acquire a reserved key (step 8)

(project)

- A *Save* button allows to update the cloud record (after filling the Key box) (step 9)

(project)

- A *Load* button allows to download the cloud record (after filling the Key box) (step 10)

(project)

Step 8-10: a map in the cloud

- We want to store our markers in the cloud
- The simplest option is to use a Key-Value service
  - a basic one is the one I implemented on MongoDB Atlas (just demonstration, not for public use)
- A *New* button in the interface allows the user to acquire a reserved key (step 8)

(project)

- A *Save* button allows to update the cloud record (after filling the Key box) (step 9)

(project)

- A *Load* button allows to download the cloud record (after filling the Key box) (step 10)

(project)

Step 8-10: a map in the cloud

- We want to store our markers in the cloud
- The simplest option is to use a Key-Value service
  - a basic one is the one I implemented on MongoDB Atlas (just demonstration, not for public use)
- A *New* button in the interface allows the user to acquire a reserved key (step 8)

(project)

- A *Save* button allows to update the cloud record (after filling the Key box) (step 9)

(project)

- A *Load* button allows to download the cloud record (after filling the Key box) (step 10)

(project)

Step 8-10: a map in the cloud

- We want to store our markers in the cloud
- The simplest option is to use a Key-Value service
  - a basic one is the one I implemented on MongoDB Atlas (just demonstration, not for public use)
- A *New* button in the interface allows the user to acquire a reserved key (step 8)

(project)

- A *Save* button allows to update the cloud record (after filling the Key box) (step 9)

(project)

- A *Load* button allows to download the cloud record (after filling the Key box) (step 10)

(project)

Step 8-10: a map in the cloud

- We want to store our markers in the cloud
- The simplest option is to use a Key-Value service
  - a basic one is the one I implemented on MongoDB Atlas (just demonstration, not for public use)
- A *New* button in the interface allows the user to acquire a reserved key (step 8)

(project)

- A *Save* button allows to update the cloud record (after filling the Key box) (step 9)

(project)

- A *Load* button allows to download the cloud record (after filling the Key box) (step 10)

(project)

Firebase deployment

- For this you need a Google account
- You need first to access the console of the service at `https://console.firebase.google.com/` and add a new project
  - in the following dialog, do not enable Google Analytics
  - observe the firebase logo in the upper left corner
- In the Stackblitz window click on the firebase logo in the left toolbar
- Click on the name of your project and next "Deploy"
- Finally click on the "Open Site", or visit `<YourProjectName>.firebaseapp.com`
- Your app is now permanently available at that URL

Firebase deployment

- For this you need a Google account
- You need first to access the console of the service at `https://console.firebase.google.com/` and add a new project
  - in the following dialog, do not enable Google Analytics
  - observe the Firebase logo, in the upper left corner
- In the Stackpoltz window click on the Firebase logo in the left toolbar
- Click on the name of your project and then "Deploy"
- Finally click on "Run Deploy" and visit `<YourProjectName>.firebaseapp.com`
- Your app is now permanently available at that URL

Firebase deployment

- For this you need a Google account
- You need first to access the console of the service at `https://console.firebase.google.com/` and add a new project
  - in the following dialog, do not enable Google Analytics
  - observe the firebase logo, in the upper left corner
- In the Stackblitz window click on the firebase logo in the left toolbar
- Click on the name of your project and then "deploy"
- Finally click on the "Open Site" button
- Your app is now permanently available at the URL `<YourProjectName>.firebaseapp.com`

Firebase deployment

- For this you need a Google account
- You need first to access the console of the service at `https://console.firebase.google.com/` and add a new project
  - in the following dialog, do not enable Google Analytics
  - observe the firebase logo, in the upper left corner
- In the Stackblitz window click on the firebase logo in the left toolbar
- Click on the name of your project and next "Deploy"
- Finally click on the "Open Site" or visit `<YourProjectName>.firebaseapp.com`
- Your app is now permanently available at that URL

Firebase deployment

- For this you need a Google account
- You need first to access the console of the service at `https://console.firebase.google.com/` and add a new project
  - in the following dialog, do not enable Google Analytics
  - observe the firebase logo, in the upper left corner
- In the Stackblitz window click on the firebase logo in the left toolbar
- Click on the name of your project and next "Deploy"
- Finally click on the "Open Site", or visit `<YourProjectName>.firebaseapp.com`
- Your app is now permanently available at that URL

Firebase deployment

- For this you need a Google account
- You need first to access the console of the service at `https://console.firebase.google.com/` and add a new project
  - in the following dialog, do not enable Google Analytics
  - observe the firebase logo, in the upper left corner
- In the Stackblitz window click on the firebase logo in the left toolbar
- Click on the name of your project and next "Deploy"
- Finally click on the "Open Site", or visit `<YourProjectName>.firebaseapp.com`
- Your app is now permanently available at that URL

Firebase deployment

- For this you need a Google account
- You need first to access the console of the service at `https://console.firebase.google.com/` and add a new project
  - in the following dialog, do not enable Google Analytics
  - observe the firebase logo, in the upper left corner
- In the Stackblitz window click on the firebase logo in the left toolbar
- Click on the name of your project and next "Deploy"
- Finally click on the "Open Site", or visit `<YourProjectName>.firebaseapp.com`
- Your app is now permanently available at that URL

Firebase deployment

- For this you need a Google account
- You need first to access the console of the service at `https://console.firebase.google.com/` and add a new project
  - in the following dialog, do not enable Google Analytics
  - observe the firebase logo, in the upper left corner
- In the Stackblitz window click on the firebase logo in the left toolbar
- Click on the name of your project and next "Deploy"
- Finally click on the "Open Site", or visit `<YourProjectName>.firebaseapp.com`
- Your app is now permanently available at that URL

Stackblitz interface

- In the right frame we see the preview of our service
  - the URL on top of the screen is functional (try it...)
- In the center frame there is a code editor
  - try to change the title in the `app.component.html` file
- In the left frame there is the project content and github reference
- The left toolbar contains the general Angular commands
- The top toolbar is for project management

Stackblitz interface

- In the right frame we see the preview of our service
    - the URL on top of the screen is functional (try it...)
- In the center frame there is a code editor
    - try to change the string in line 6 and notice the preview change
- In the left frame there is the project content and github reference
- The left toolbar controls the content of the left column
- The top toolbar is for project management

Stackblitz interface

- In the right frame we see the preview of our service
  - the URL on top of the screen is functional (try it...)
- In the center frame there is a code editor
  - try to change the string in line 6 and notice the preview change
- In the left frame there is the project content and github reference
- The left toolbar controls the content of the left column
- The top toolbar is for project management

Stackblitz interface

- In the right frame we see the preview of our service
  - the URL on top of the screen is functional (try it...)
- In the center frame there is a code editor
  - try to change the string in line 6 and notice the preview change
- In the left frame there is the project content and github reference
- The left toolbar controls the content of the left column
- The top toolbar is for project management

Stackblitz interface

- In the right frame we see the preview of our service
  - the URL on top of the screen is functional (try it...)
- In the center frame there is a code editor
  - try to change the string in line 6 and notice the preview change
- In the left frame there is the project content and github reference
- The left toolbar controls the content of the left column
- The top toolbar is for project management

Stackblitz interface

- In the right frame we see the preview of our service
  - the URL on top of the screen is functional (try it...)
- In the center frame there is a code editor
  - try to change the string in line 6 and notice the preview change
- In the left frame there is the project content and github reference
- The left toolbar controls the content of the left column
- The top toolbar is for project management

Stackblitz interface

- In the right frame we see the preview of our service
  - the URL on top of the screen is functional (try it...)
- In the center frame there is a code editor
  - try to change the string in line 6 and notice the preview change
- In the left frame there is the project content and github reference
- The left toolbar controls the content of the left column
- The top toolbar is for project management

One screen development environment

- The project code resides on the Stackblitz Webserver
- The user get access to the project following a web link
- The page displays an editable
- The user interacts with the browser clicking buttons, filling forms and such
- All this is synthesized in the Stackblitz screen

One screen development environment

- The project code resides on the Stackblitz Webserver
- The user get access to the project following a web link
- The page displays an editable
- The user interacts with the browser clicking buttons, filling forms and such
- All this is synthesized in the Stackblitz screen

One screen development environment

- The project code resides on the Stackblitz Webserver
- The user get access to the project following a web link
- The page displays an editable
- The user interacts with the browser clicking buttons, filling forms and such
- All this is synthesized on the Stackblitz screen

One screen development environment

- The project code resides on the Stackblitz Webserver
- The user get access to the project following a web link
- The page displays an editable
- The user interacts with the browser clicking buttons, filling forms and such
- All this is synthesised in the StackBlitz screen

One screen development environment

- The project code resides on the Stackblitz Webserver
- The user get access to the project following a web link
- The page displays an editable
- The user interacts with the browser clicking buttons, filling forms and such
- All this is synthesised in the StackBlitz screen
  - using a Web server which is appropriate only for development
  - a real deployment may run on Firebase (third icon in the left toolbar)
  - free plan available. Google account needed

One screen development environment

- The project code resides on the Stackblitz Webserver
- The user get access to the project following a web link
- The page displays an editable
- The user interacts with the browser clicking buttons, filling forms and such
- All this is synthesised in the StackBlitz screen
  - using a Web server which is appropriate only for development
  - a real deployment may run on Firebase (third icon in the left toolbar)
  - free plan available. Google account needed

One screen development environment

- The project code resides on the Stackblitz Webserver
- The user get access to the project following a web link
- The page displays an editable
- The user interacts with the browser clicking buttons, filling forms and such
- All this is synthesised in the StackBlitz screen
  - using a Web server which is appropriate only for development
  - a real deployment may run on Firebase (third icon in the left toolbar)
  - free plan available, Google account needed

One screen development environment

- The project code resides on the Stackblitz Webserver
- The user get access to the project following a web link
- The page displays an editable
- The user interacts with the browser clicking buttons, filling forms and such
- All this is synthesised in the StackBlitz screen
  - using a Web server which is appropriate only for development
  - a real deployment may run on Firebase (third icon in the left toolbar)
  - free plan available, Google account needed