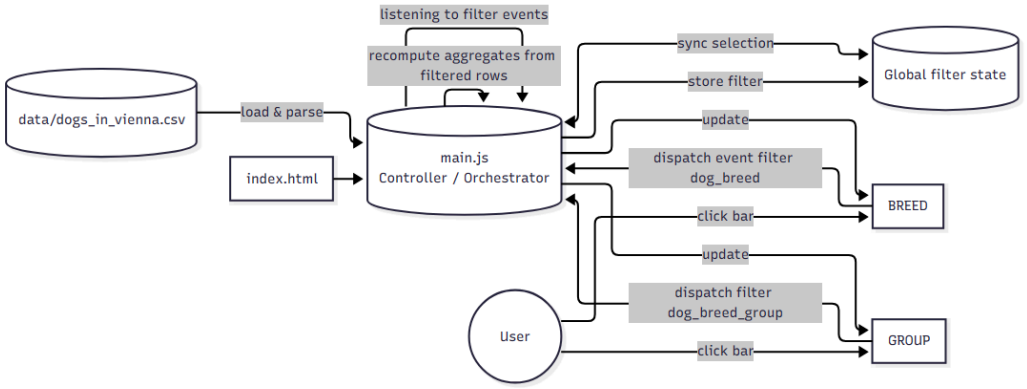
Checkpoint III: First Prototype

Group: 21
Date: 2025/10/09

Prototype Architecture



[Link to Graph]

Modules and Files

index.html

- Hosts two empty containers (e.g., #chart2, #chart3) and loads D3 + ES modules.
- No logic; just bootstraps the app.

script.js (controller/orchestrator)

- Loads the CSV and keeps the raw rows in memory.
- Holds the single source of truth for filter state (selected group / selected breed).
- Computes all derived aggregates (rollups for breeds and groups).
- Instantiates charts and wires their events.
- On any chart's "filter" event: updates global state → recomputes aggregates → calls update(...) on every chart.

breedChart.js (view component)

- Renders the Top-10 breeds bar chart (horizontal).
- Exposes a small public API: on(event), update(data, state), highlightBreed(), highlightByGroup(), hoverByGroup().
- Emits a semantic "filter" event with the selected breed (or null to clear).
- No data loading; only renders what it's given.

groupChart.js (view component)

- Renders the breed-groups bar chart (horizontal).
- Exposes a small public API: on(event), update(data, state), highlightGroup().
- Emits a semantic “filter” event with the selected group (or null to clear).
- Emits a semantic “hover” event with the selected group (or null to clear).
- No data loading; only renders what it’s given.

data/dogs_in_vienna.csv (data source)

- Raw input used by script.js.

style.css

- styles and organizes display of the different idioms.

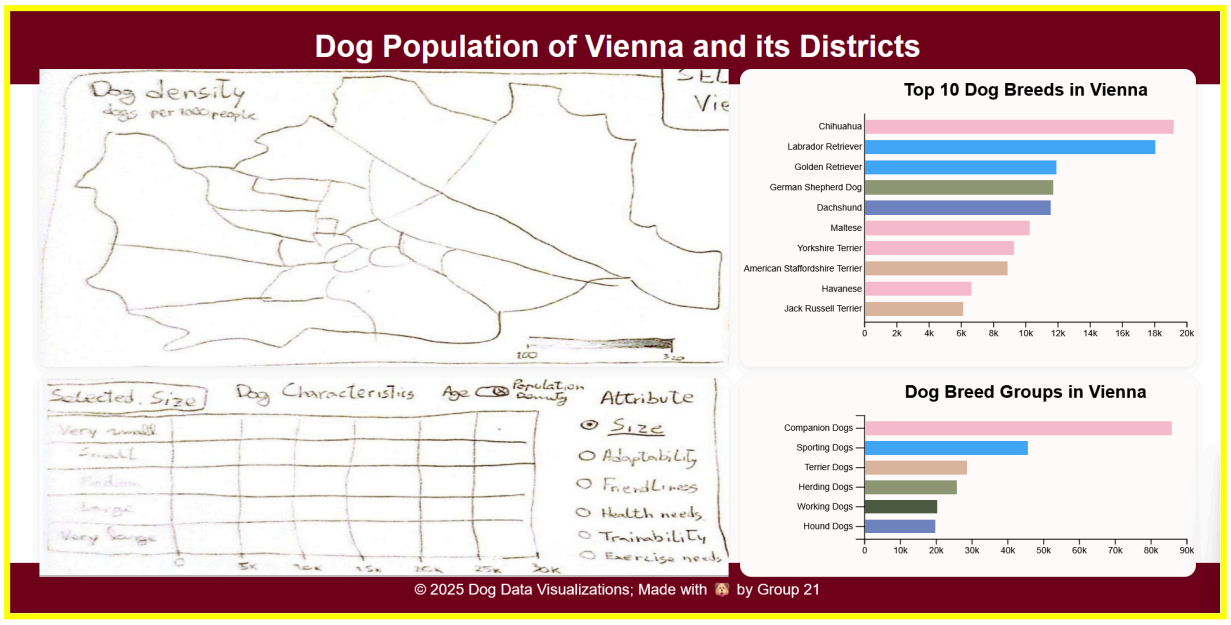
colors.js

- for better testing/asserting value of different color palettes.

Interaction Patterns

- ES modules with factory functions per chart returning a minimal API.
- Unidirectional data flow: main.js owns the state and pushes data down via update(...).
- Event-driven communication: charts publish semantic “filter” events via an internal dispatcher; main.js listens and reacts.
- Stable data joins in charts (keyed by breed / group) so updates are smooth.
- Charts are stateless with respect to data (state only reflected through the state argument on update).

Dashboard Layout



The dashboard consists of four main panels. On the **right side** of the dashboard, the **top-right panel** shows the top ten dog breeds, and the **bottom-right panel** shows the main dog breed groups. Each group and its breeds share a hue reflecting their function—for example, terriers are light brown, symbolizing their role as “earth dogs” that chased animals from underground. Both bar charts are linked and form the focus of this week’s checkpoint.

On the left side, the top panel will show an interactive map of Vienna’s districts, updating the right-hand charts when a district is selected. The bottom panel will feature a tile map comparing

human demographics (population density or average age) with dog attributes such as size or adaptability.

This layout connects spatial data, demographic factors, and breed distributions in one view, making it easier to explore Vienna's dog population from multiple perspectives.

Data Processing

In order to load our charts, there are three essential steps:

- We roll up the dataset by breed, or by group, for the breed bar chart and group bar chart respectively;
- We select only relevant attributes, which are dog_breed_group and dog_count for both charts, and also dog_breed for the first chart;
- We sort the resulting data and slice it, using 10 values for the breed chart and 6 for the group;

For the future tile heatmap, we will also need other attributes, which we will get in the second stage of this process and keep in loaded data.

After the initial loading, data can be filtered by breed, group or district, but the initial result of this process is kept in the main script, while the new filtered data will be sent to each chart.

Chart Visualization

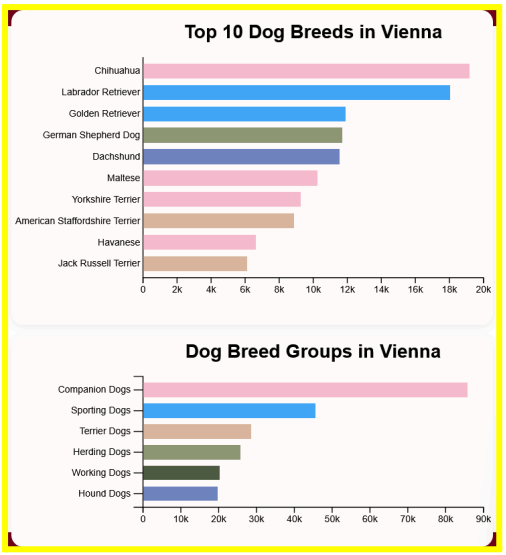


Chart Interaction

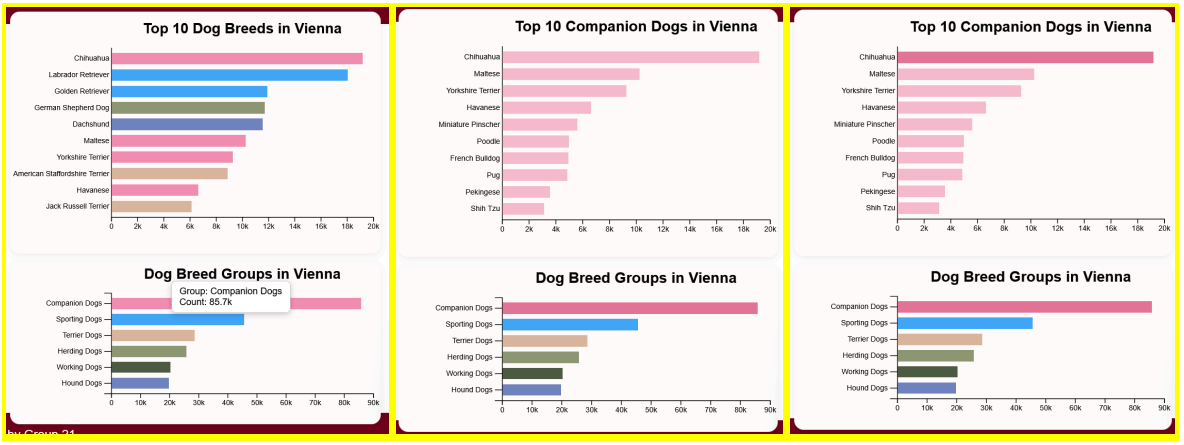
BarChart 1 (Dog Breeds): Hovering over a bar darkens its color slightly, revealing related info in a small tooltip and highlighting corresponding bars in BarChart 2. Clicking a bar applies an app-wide filter for that dog breed and changes its color to a noticeably darker shade.

BarChart 2 (Breed Groups): Hovering over a bar slightly darkens its color, displays related info in a small tooltip, and highlights matching bars in BarChart 1. Clicking a bar applies an app-wide filter for that breed group and darkens its color significantly.

Map: Clicking on a district will filter all data application wide by this district

Tile Map: This visualization allows the user to interactively choose the feature for the x- and y-axis.

For the x-axis Age and Population-Density can be chosen. For the y-axis the user can switch between dog traits, size, adaptability, etc.



Hovering over a dog group highlights its breeds (left image), while hovering over a breed highlights its group. Clicking a group filters all data by that group (middle image), showing only the most common breeds in the selected district—or citywide if none is chosen.

By clicking on a dog breed bar, an equivalent effect happens (right image). To emphasize the activations of the filter, the bars are highlighted and there is a color shift from that breed/group's hue to a considerably darker shade. None of these interactions filter the bar-chart to a single bar.

Chart Integration

When clicking on a dog breed, the breedChart sends the main script a filter event, with the select breed in its filterState. The main script then updates its filterState, and recomputes the data. This is getting the filtered rows and re-doing the processing needed to create the charts. Lastly, the main script calls the update function on the charts, with new filtered data and the new filterState.

To add new charts, they'll need to have this update function and the main script will also call the new chart's update when its re-computing and re-rendering the charts.

The project, as it stands, combines two bar charts to explore Vienna's dog population at both the district and city-wide levels.

The first chart displays the top ten dog breeds within the selected area (either a specific district or the entire city), while the second chart presents the five most popular dog breed groups. The linked interaction between the two charts enables dynamic exploration of the data.

Integration works both ways: highlighting a breed in the breed chart emphasizes it and its group in the group chart, while selecting a group updates the breed chart to show the top ten breeds for that district or city. This cross-filtering enables seamless movement between aggregated and detailed perspectives.

Together, the charts reveal patterns beyond raw counts. Some groups are popular due to a few established breeds, others through many lesser fashionable breeds. Linking both charts allows insights into these dynamics, offering deeper understanding into Vienna's canine population.