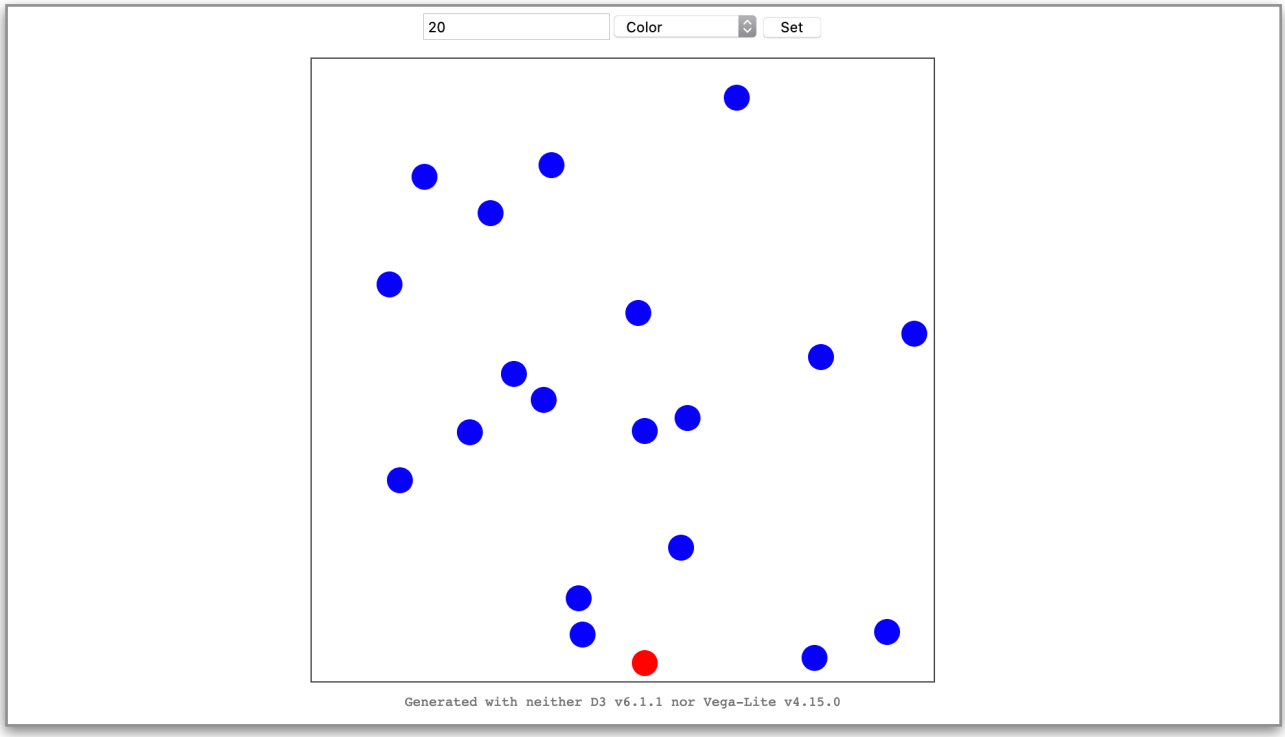
INF552 (2023-2024) - PC s01

Goal: populate an SVG canvas with simple shapes.

This exercise illustrates that conjunctive visual search is not pre-attentive (as we will see in session #02). Its more practical goal is to get you started with a proper Web development environment, and to gain some familiarity with low-level concepts and DOM API calls.



1. Setup

This will apply to all exercises, throughout all 9 sessions.

Edit `.js` and `.html` files in a text editor (such as, *e.g.*, Atom)

Depending on your browser settings, you might not be able to load external resources (data files, etc.) if you use the `file://` protocol. Thus, you are (strongly) encouraged to use a (local) Web server.

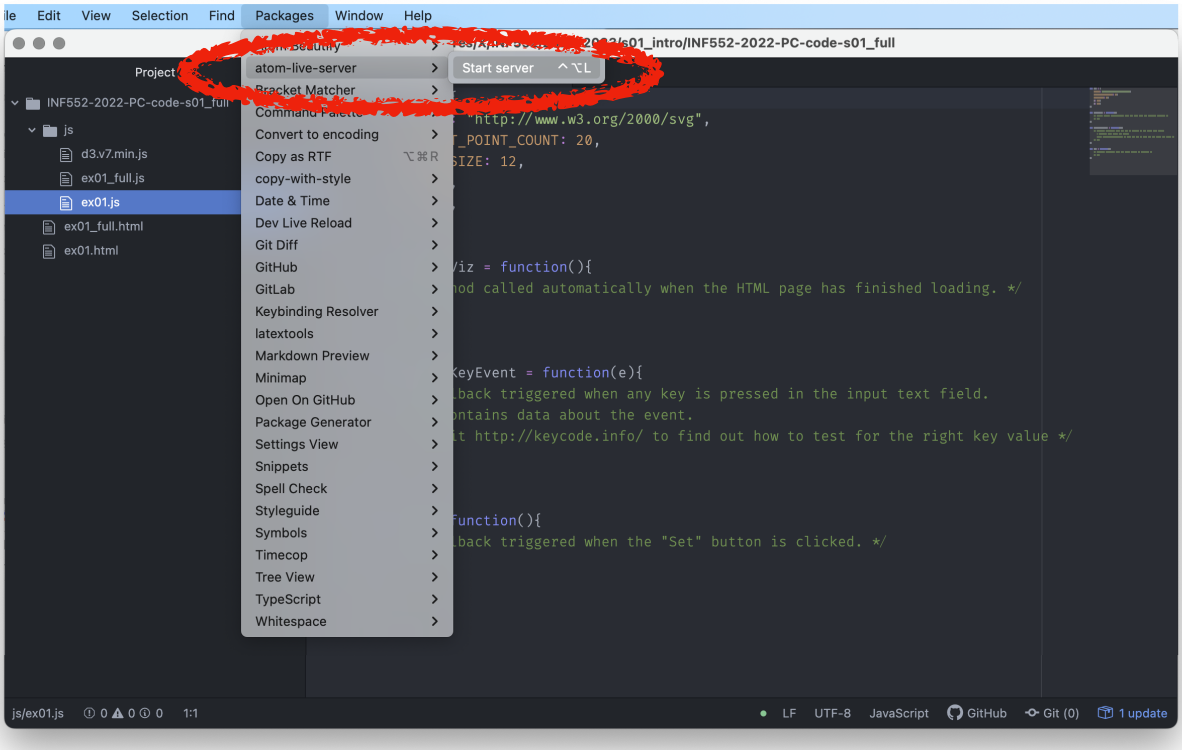
See different options on the next page.

- Start a simple Web server in the relevant directory
  - using, *e.g.*, Python:

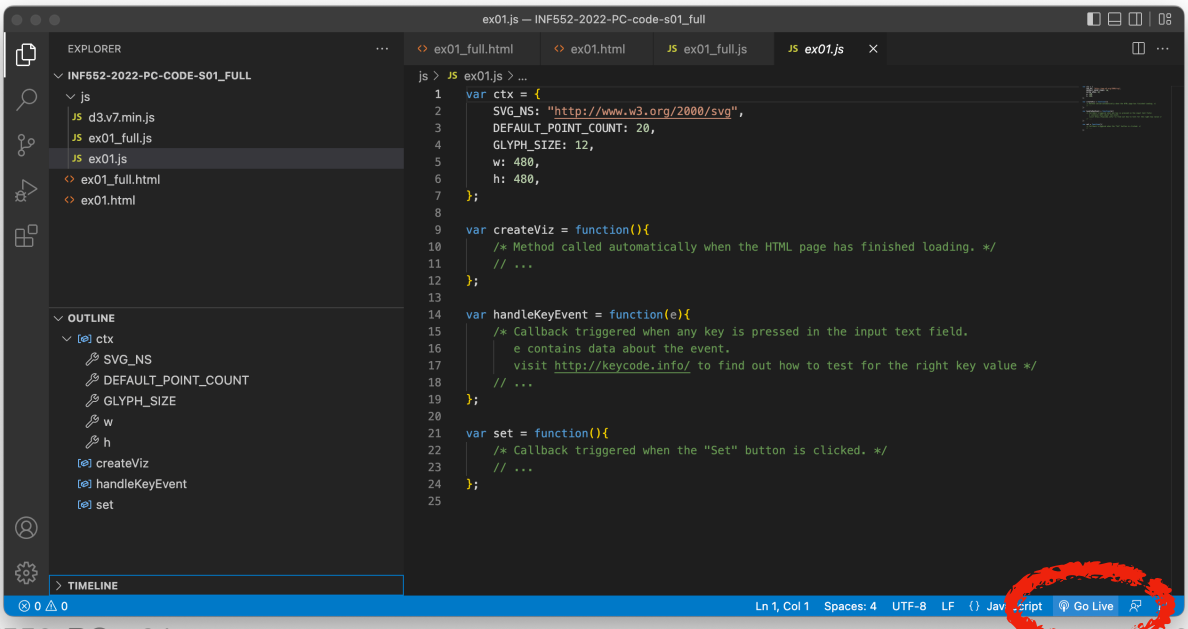
    ```
    cd INF552-2022-code-s01/; python3 -m http.server [portNumber]
    ```

  - and then access it from `http://localhost:portNumber/`

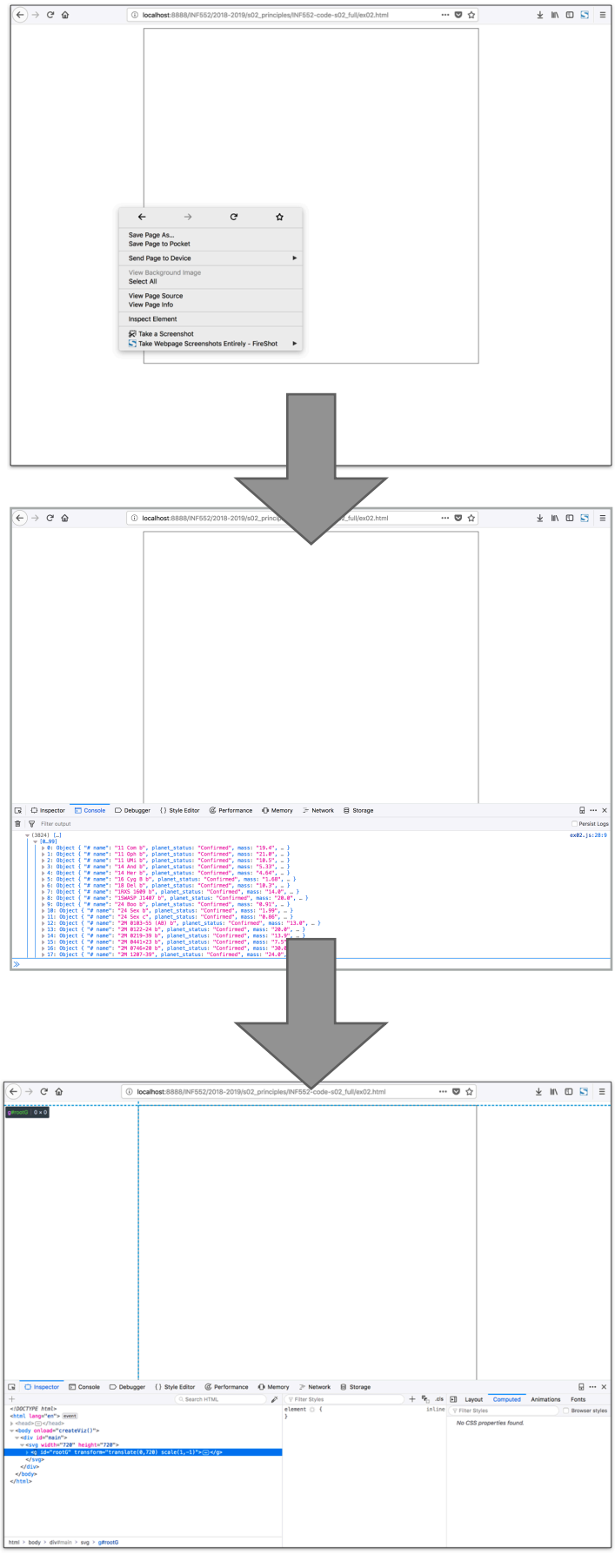
- or use any integrated solution, such as atom-live-server if you edit your code with Atom
  `https://atom.io/packages/atom-live-server`



- or Live Server if you edit your code with Visual Studio Code
  `https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer`

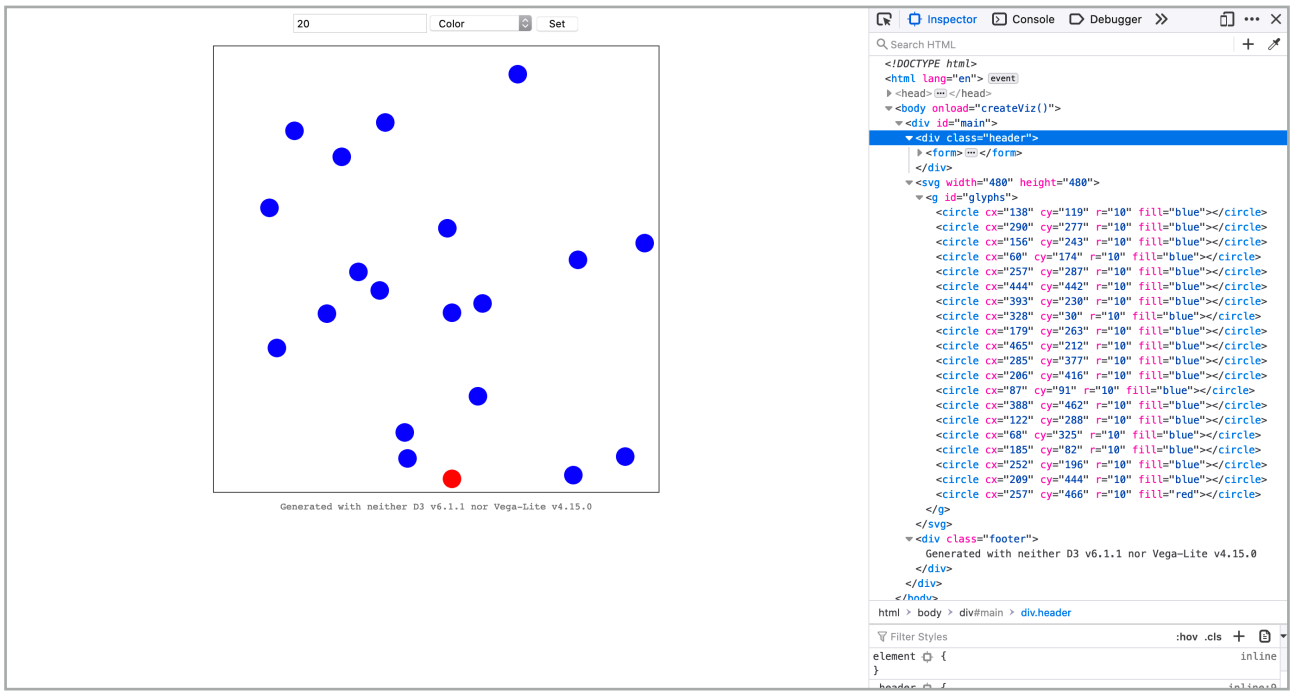


In your code, *always* use *relative* URLs.
Use the developer tools provided by your Web browsers to debug your code. Essentially: the console, and the inspector. Example screenshots using Firefox:



2. Populating the SVG Canvas

The code skeleton already features the form/input elements and the associated (empty) event callbacks.

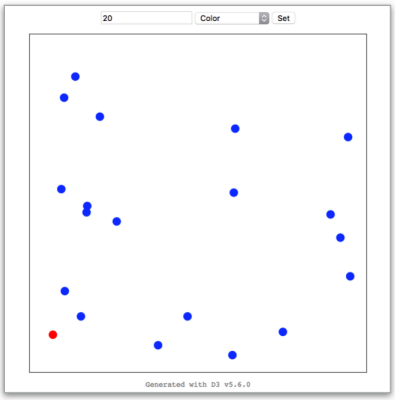


Main steps:

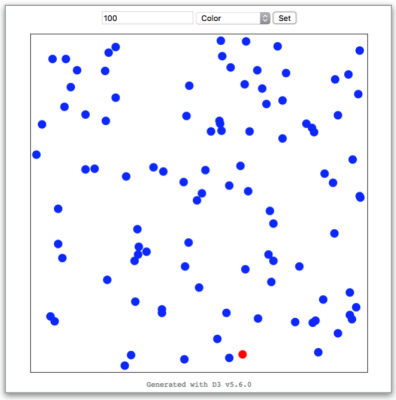
- Using the DOM API, add an SVG element of size 480x480 in `div#main`
- Populate the SVG canvas depending on the parameters provided in the input form, as detailed in Section 2.1
- Populating the canvas should be triggered when hitting button `Set`, or when hitting the `enter` key in the text field. See empty callbacks already provided in `ex01.js`.
  Tip: use http://keycode.info to find out how to test for the right key value.

2.1. Parameters

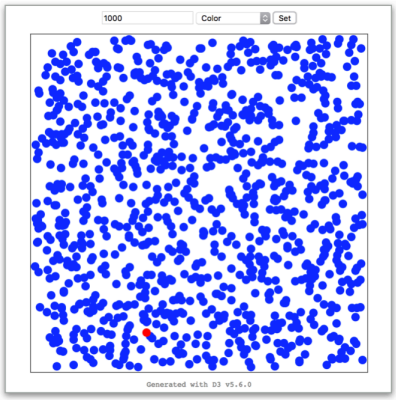
The first parameter is an integer indicating the total number of glyphs:



N=20

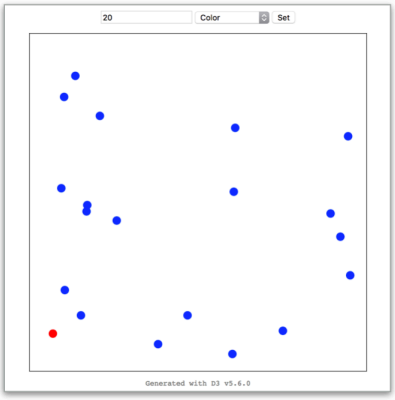


N=100



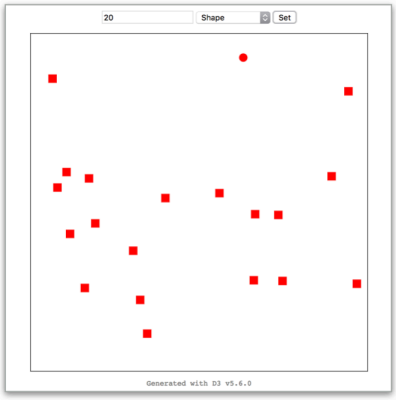
N=1000

The second parameter controls the type of visual search:



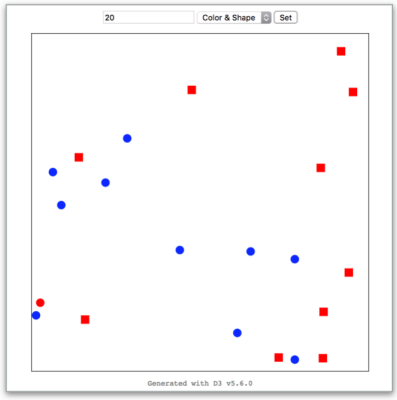
Color

Populate the canvas with:
- N-1 blue circles;
- 1 red circle.



Shape

Populate the canvas with:
- N-1 red rectangles;
- 1 red circle.



Color & Shape

Populate the canvas with:
- N/2 red rectangles;
- N/2 blue circles;
- 1 red circle.

Some general tips:

- Put your circles and rectangles inside a `<g>` element (group), that element being the only child of the `<svg>` element created earlier.
- Do not forget to clear the contents of the SVG canvas before populating it again (just remove the above `<g>` element).
- Be careful about namespace declarations when creating SVG nodes using the DOM API. Use `Document.createElementNS()`

2.2. Fine-tuning

Display D3 and Vega-Lite version numbers below the SVG canvas in a `<div>`. Fetch them programmatically, with `d3.version` and `vegaEmbed.vegaLite.version`

Add CSS styling rules to make the UI look as similar as possible to the one below. Either use the `<style>` attribute already in `<head>` or reference an external stylesheet.