Prerequisites

Previously you learned about how to use SVGs within HTML documents to produce visual outputs, how to select and manipulate elements via JavaScript and D3, and how to load and prepare data. We also discussed scales, axes, and the enter-update-exit pattern. You will need this knowledge to participate in this tutorial.

Feel free to look up what we did in the last tutorials at any time:

http://vda.univie.ac.at/Teaching/Vis/16s/Tutorials/D3_Tutorial_1.pdf

http://vda.univie.ac.at/Teaching/Vis/16s/Tutorials/D3_Tutorial_2.pdf

Mouse Events

Often we want more interactivity than just manipulating our visualization via HTML GUI elements. For this D3 provides the *.on(type, listener)* function, which can be called on selections. The *type* is a string specifying which kind of event is listened for. Examples would be "click", "mouseover", "keydown". The second argument, the *listener*, is an anonymous function defining what to do when an event is caught. It gets passed the current datum, *d*, and its index, *i*. Moreover, you can access the element on which the event was triggered with *d3.select(this)* within the anonymous function.

```
d3.selectAll("circle").on("click", function(d) {
        console.log(d);
 });
```

More about the *.on()* method can be found here: https://github.com/mbostock/d3/wiki/Selections#on

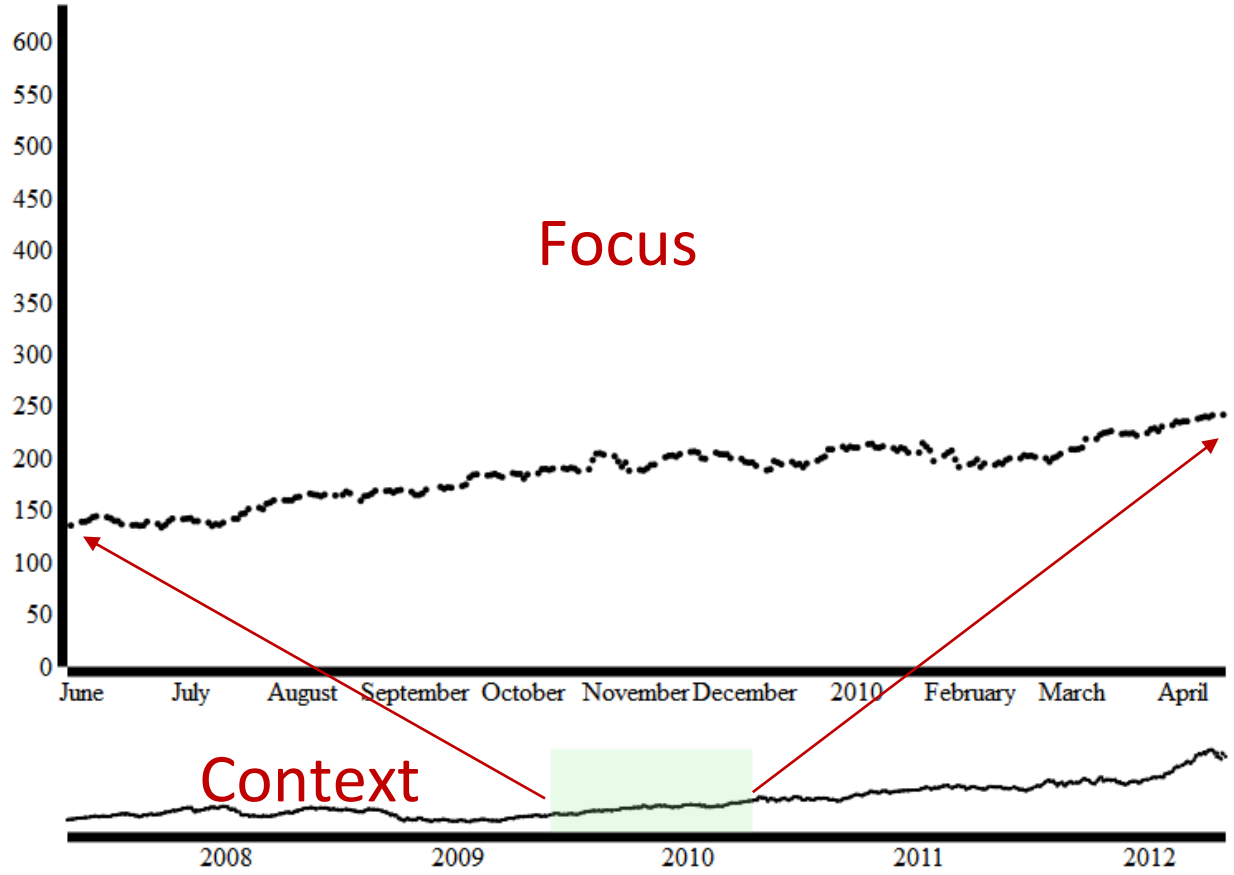
A list of possible event types can be found here: https://developer.mozilla.org/en-US/docs/Web/Events#Standard_events

Exercise

- Get the code from http://vda.univie.ac.at/Teaching/Vis/16s/Tutorials/src/d3tutorial3_template.html
- Download the data http://vda.univie.ac.at/Teaching/Vis/16s/Tutorials/data/sp500.csv
- Make yourself familiar with the code
- Make the points change their color when clicked.

Linking & Brushing

In more complex visualizations it is often required that the user is able to select a subset of the data – *brushing* – and these selections to be reflected in other views – *linking*. The case where there is one visualization acting as a detail view, and one as an overview, is often called Focus+Context.



Brushing is conveniently implemented in D3 via the *brush* component. This creates an element that works similar to an axis and allows the user to select subsets of the data with their mouse.

```
var brush = d3.svg.brush()
      .x(myContextScaleX)
      .on("brushend", brushed);
```

The brush can react to three events, again caught with the *.on()* method: "*brushstart*" is triggered on mousedown, "*brush*" on mousemove after brushstart, and "*brushend*" on mouseup. In the code example "brushed" is the name of a function implemented elsewhere in your code.

After the user triggered an event, the function passed to the .on() method is called. What is usually done here is that the domain of the focus view's x-axis is adapted to fit the domain selected by the user vie the brush. For this purpose, D3 provides the brush.extent() method that returns the selected region.

```
function brushed() {
        if (brush.empty() == true) {
                myFocusScaleX.domain(myContextScaleX.domain());
        }else{
                myFocusScaleX.domain(brush.extent());
        }
        //Update Focus visualization here
}
```

This example adapts the domain of the scale used for the x-axis of the focus plot to the selected region of the brush. If the user made no selection, it sets it to the domain to be equal to the one used by the context visualization – containing all data points.

Now the brush has to be added to an SVG or SVG group, just like an axis. Additionally, we add a rectangle, indicating the user's selection.

```
myGroup.append("g")
        .call(brush)
        .selectAll("rect")
        .attr("y", yOffset)
        .attr("height", height)
        .style("fill","lightgreen")
        .style("fill-opacity", ".2");
```

More about brushing here: https://github.com/mbostock/d3/wiki/SVG-Controls

Exercise

- Reuse your previous code
- Implement a brush for the x axis. (Hint: You can place it in the space below the main visualization – in variable names denoted with 'context'.) The result should look like the image above. Remember, for the brush you need :
    - A group containing:
        - A brush component
        - An extra x-axis (like in the focus view)
        - An extra y-scale
        - A plot of the data-elements (like in the focus view)
        - Call the brush and the axis on their own a SVG groups
    - A function handling what happens once a brush event is triggered
        - Adapting the focus-scale
        - Redrawing the elements shown in the focus view

CSS

Often you want to set the style of your components for multiple parts of your visualization project at once. For this purpose you can use CSS style sheets. These are separate documents with the .css file extension. In them you can define style properties via the selectors we discussed in the first tutorial – "elementName", ".elemenClass", "#elementID".

As an example:

```
div {
      font-size: 11px;
      background-color: red;
}
```

In you HTML file, a css style sheet can be included via the following tag:
`<link rel="stylesheet" href="myStyleSheet.css">`

More about CSS here: http://www.w3schools.com/css/

Exercise

- Remove all styles in your file and replace them with a CSS style sheet.
- Make your visualization more appealing via CSS. One idea would be to improve the axes: http://alignedleft.com/tutorials/d3/axes

General Tips:

- If you are creating larger visualization systems, don't forget to think object-oriented. There are ways in JS that allow for object oriented behavior, and it is generally a good idea to separate chunks of your architecture into their own, self-contained files.
- To set styles your visualization components, usually not the .style() method is used, but as much of the style information as possible is defined in dedicated .css style sheets.
- Often D3 projects feature subfolders like /js, /data, and /css, that contain different parts of the solution. This helps organizing your files.
- For creating larger web and mobile projects it is useful to utilize additional frameworks to structure your solution. One popular framework is bootstrap: http://getbootstrap.com/
- This tutorial-series was held simple and to the point – if you are implementing your course project in D3, look out for more elegant, efficient, or powerful solution; there is plenty out there. (But remember the academic honesty guidelines.)

- It is generally a good idea to define your layout in terms of offsets, heights, and widths at the top of your code, so you have it all organized and at the same spot.

Acknowledgements

Many thanks to Michael Opperman for all the help and material he provided.