Universität Konstanz

Hands-on with D3

Prof. Dr. Daniel A. Keim
University of Konstanz, Germany

2nd ACM Europe Summer School in Data Science 12 – 18 July 2018, Athens, Greece

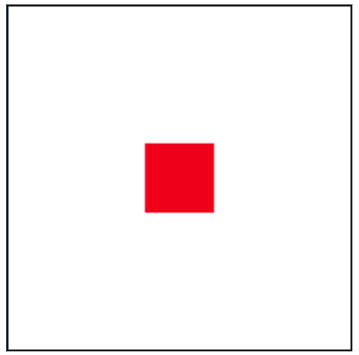
What is D3.js?

- *D3 brings data to life using HTML, SVG, and CSS*
- D3 builds on web standards and gives you the full capabilities of modern browsers
- D3 combines powerful visualization components and a data-driven approach to DOM manipulation
- URL: https://d3js.org/, Examples: https://bl.ocks.org/

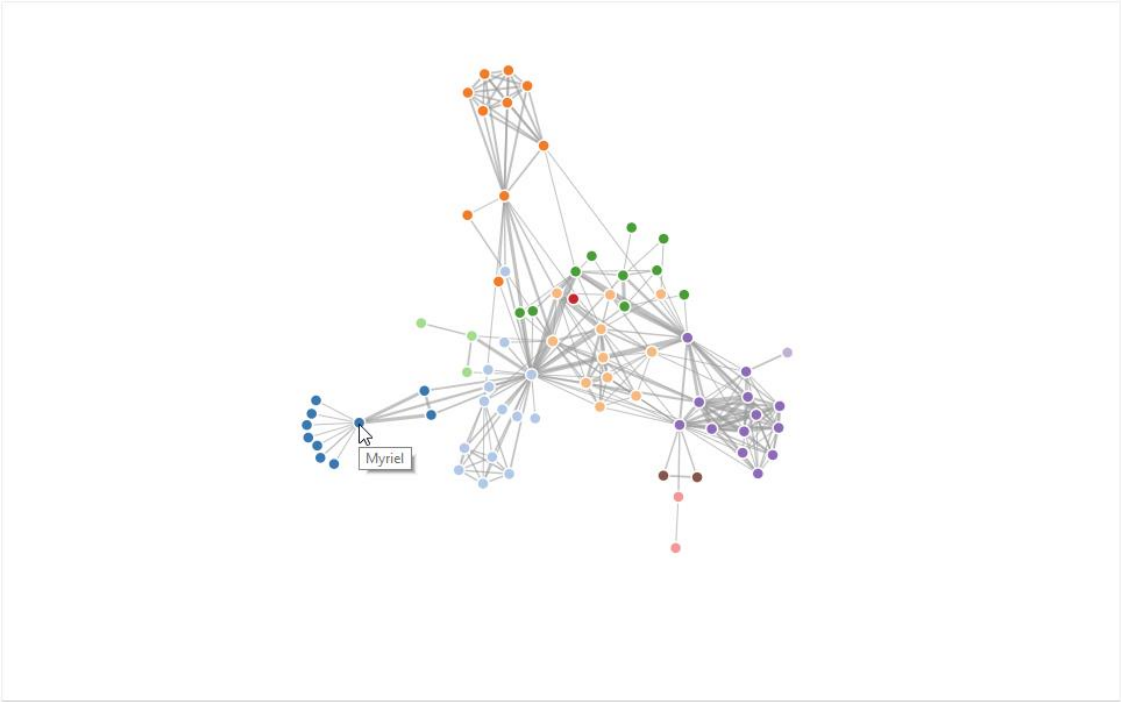
D3 is based on SVG

- SVG: *Scalabe Vector Graphics*, first specification in 2001.
- Can be styled with CSS.
- Supports: shapes, text, color, effects, interactions, …

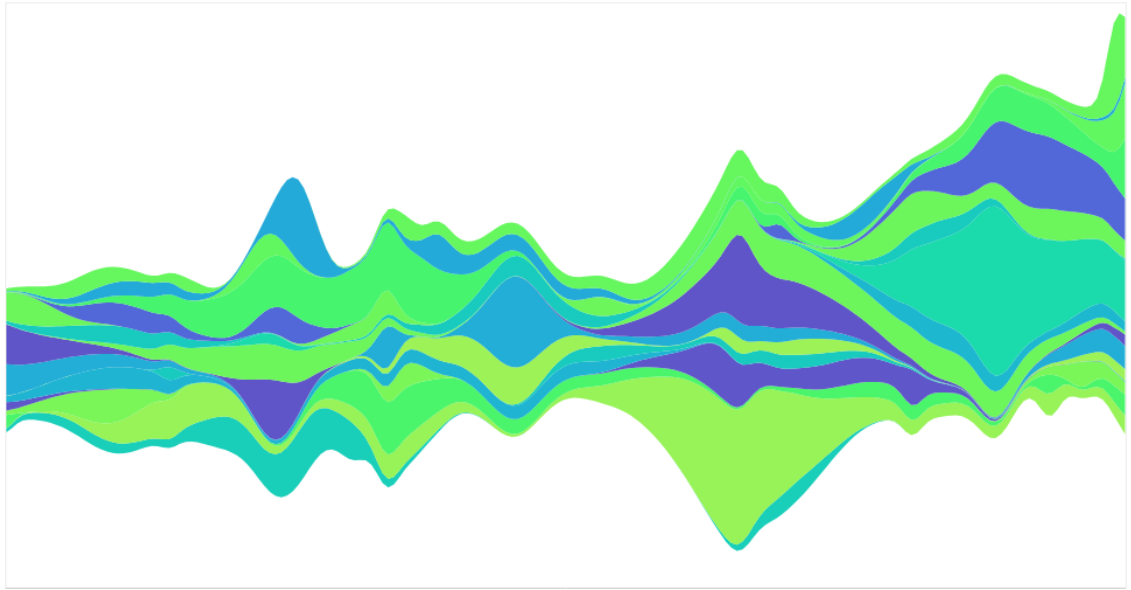
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg xmlns="http://www.w3.org/2000/svg" xmlns:svg="http://www.w3.org/2000/svg"
  width="100" height="100" version="1.1">
    <rect style="stroke: black; fill: none"
       x="0" y="0"
       width="100" height="100"/>
    <rect style="fill:red"
      x="40" y="40"
      width="20" height="20"/>
</svg>
```



Examples

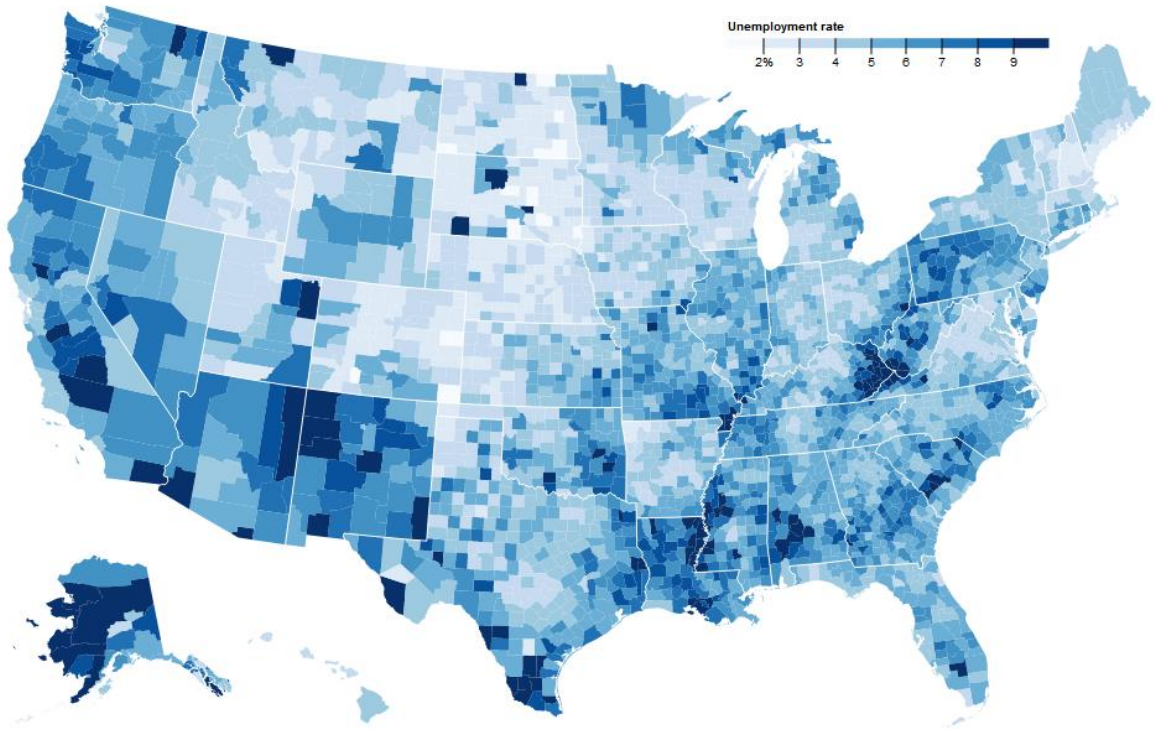


Force-Directed Graph
https://bl.ocks.org/mbostock/4062045



Streamgraph
https://bl.ocks.org/mbostock/4060954

Examples



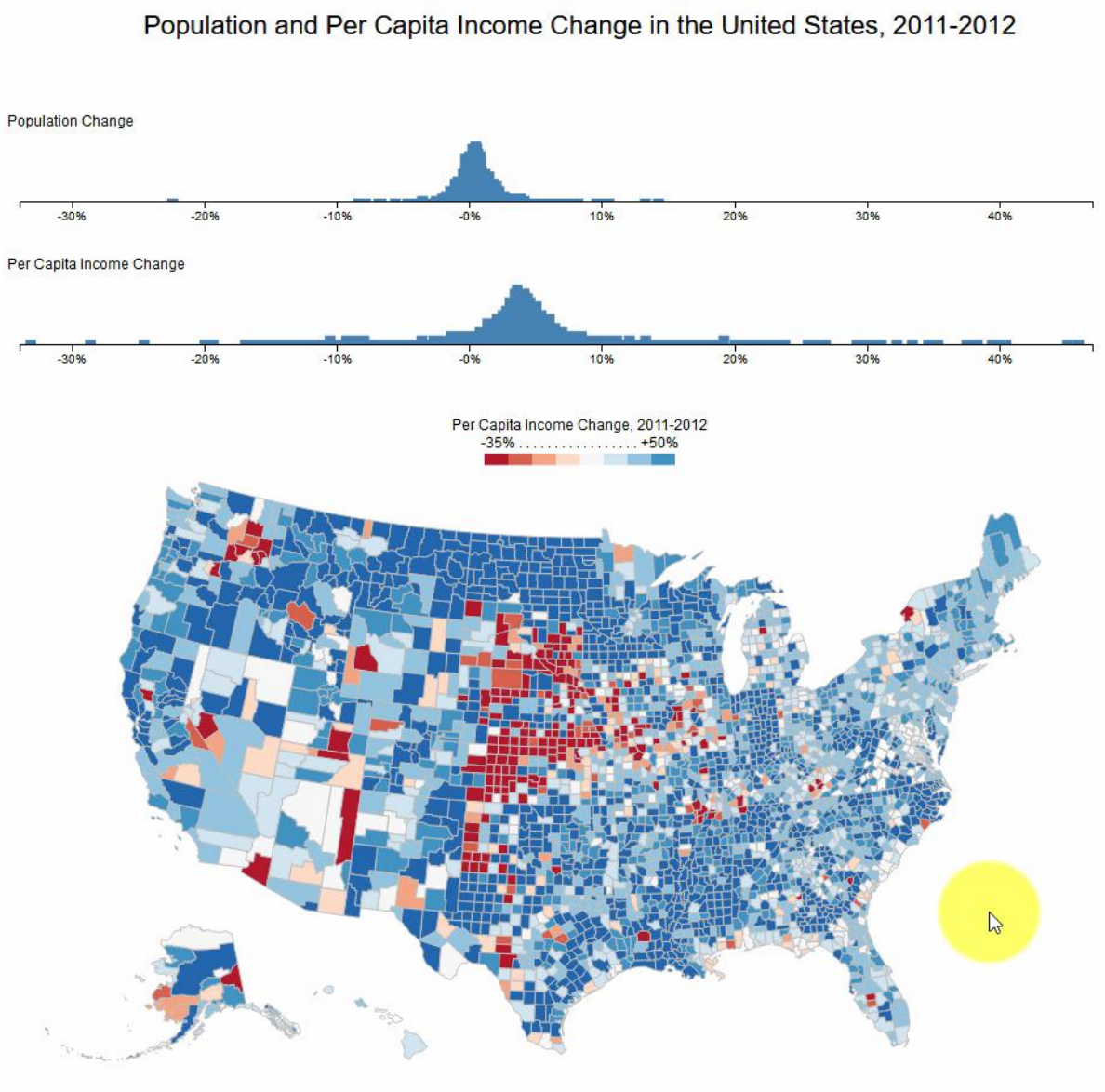
Choropleth
https://bl.ocks.org/mbostock/4060606

```
svg.append("g")
    .attr("class", "counties")
  .selectAll("path")
  .data(topojson.feature(us, us.objects.counties).features)
  .enter().append("path")
    .attr("fill", function(d) { return color(d.rate = unemployment.get(d.id)); })
    .attr("d", path)
  .append("title")
    .text(function(d) { return d.rate + "%"; });

svg.append("path")
    .datum(topojson.mesh(us, us.objects.states, function(a, b) { return a !== b; }))
    .attr("class", "states")
    .attr("d", path);
```

Interactive Examples

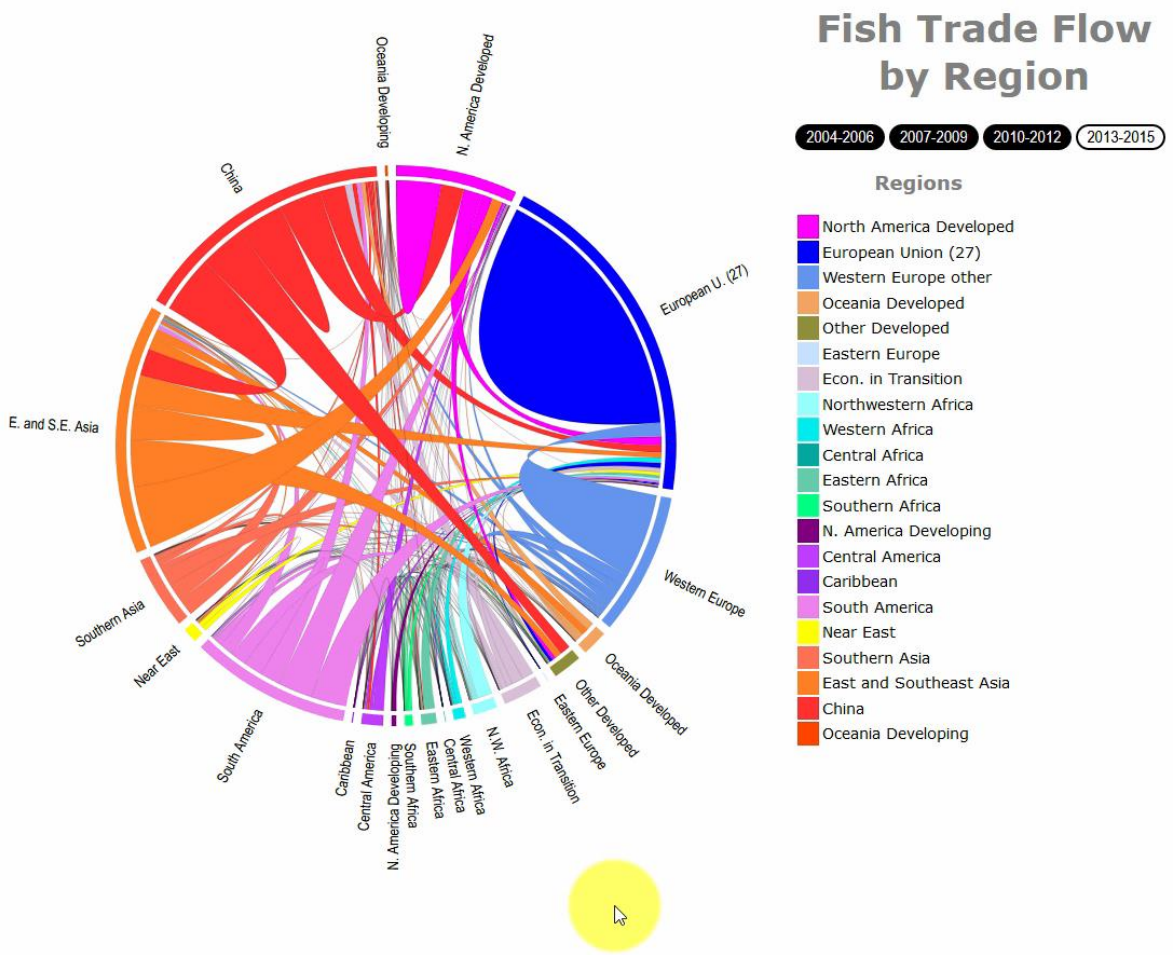
- Choropleth map.
- Drill down via tooltip.
- Three linked views:
  - histogram of population change,
  - histogram of income change,
  - map.



Interactive Choropleth
http://bl.ocks.org/pmia2/ecf6cfd7239aad7ab987cee0b7b12281/ed908f67ec3ded77019add1eb679cd19489819af

Interactive Examples

- Chord diagram.
- Drill down via tooltips.
- Shows trading volumes between geographic regions.
- Selection over time with animated transition.



Chord diagram
http://bl.ocks.org/databayou/c7ac49a23c275f0dd7548669595b8017/d3c81c0153000595b0f76347952ba9510800a2f8

D3 Principles – Initialization

- Everything is a combination of HTML, JavaScript and SVG.

```html
<div class="main">
    <div id="visualization"></div>
</div>
```

1: HTML

```javascript
//create the svg element
this.d3svg = d3
    .select("#visualization")
    .append("svg")
        .attr("id", "d3-vis")
        .attr("width", 1000)
        .attr("height", 1000);

//create the background group
this.bg = this.d3svg
    .append("g")
        .attr("id", "vis-background");

//add rect to background
this.bg
    .append("rect")
        .attr("x", 0)
        .attr("y", 0)
        .attr("width", 1000)
        .attr("height", 1000)
        .attr("fill", "none")
        .attr("stroke", "gray")
        .attr("stroke-width", 1)
```

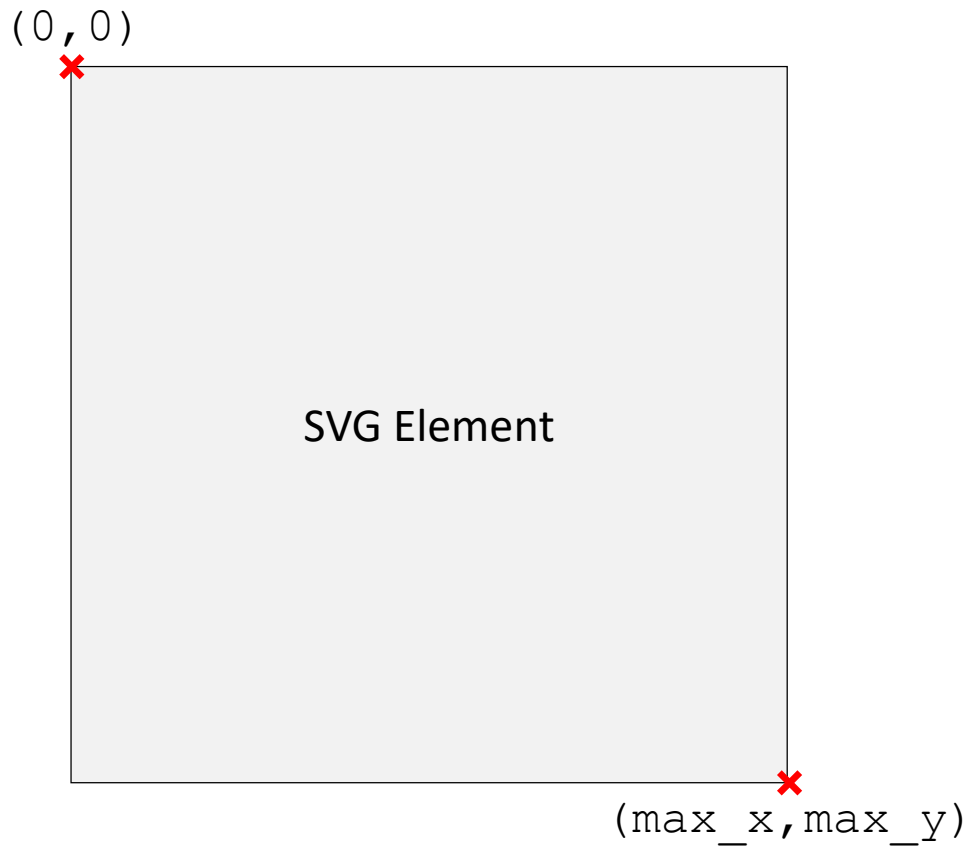
2: JavaScript/D3.js

```html
<div class="main">
    <div id="visualization">
        <svg width="1000" height="1000">
            <g id="vis-background">
                <rect x="0" y="0" width="1000" height="1000"
                    fill="none" stroke="gray" stroke-width="1"></rect>
            </g>
        </svg>
    </div>
</div>
```

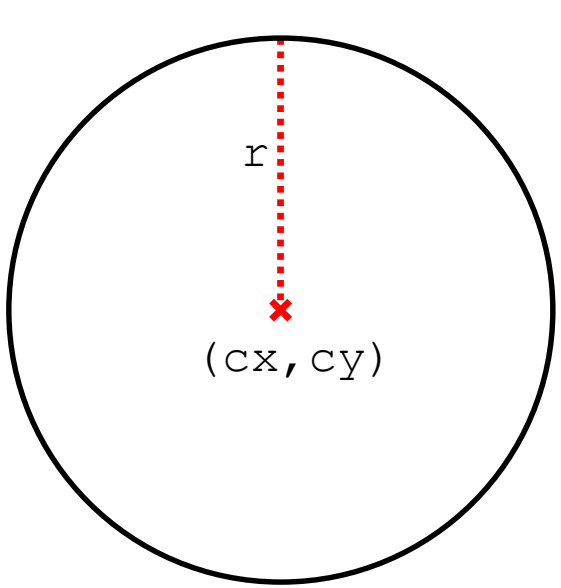
3: HTML & SVG

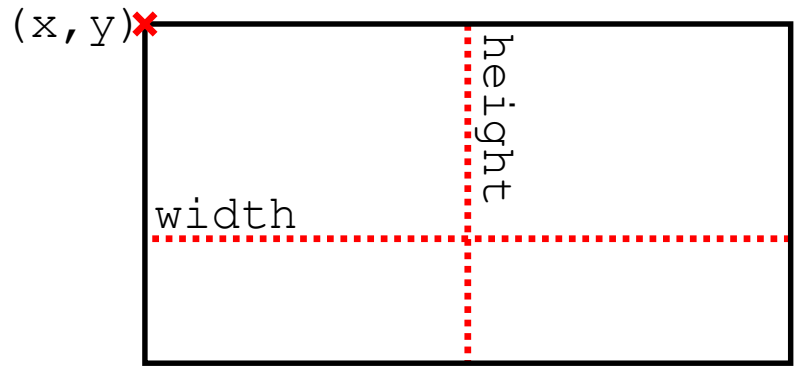
D3 Principles – Coordinates

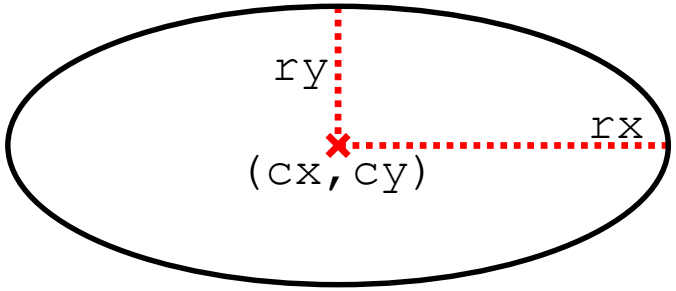
- Coordinates are given in `x, y` order.
- Origin (`0, 0`) of coordinates is on the top left.

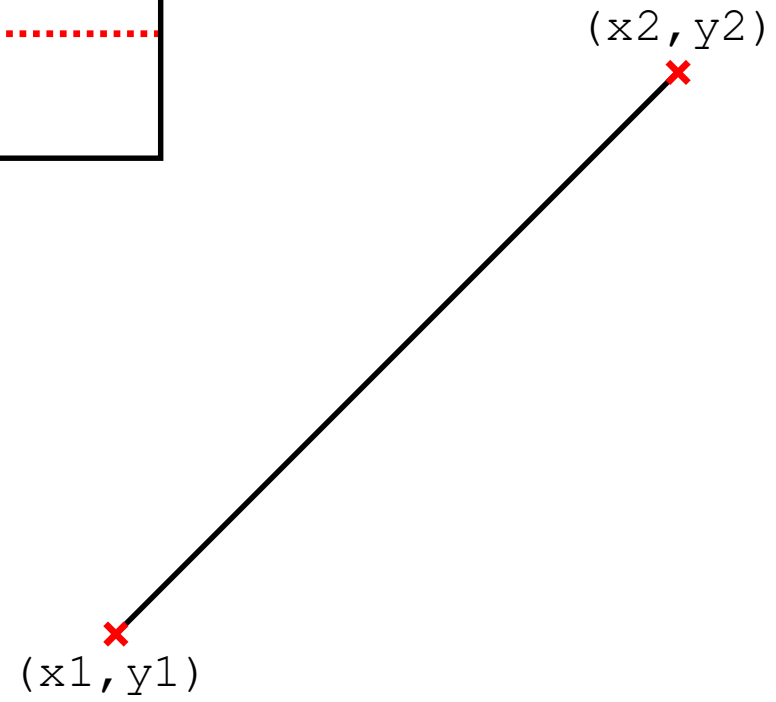


D3 Principles – Basic Shapes

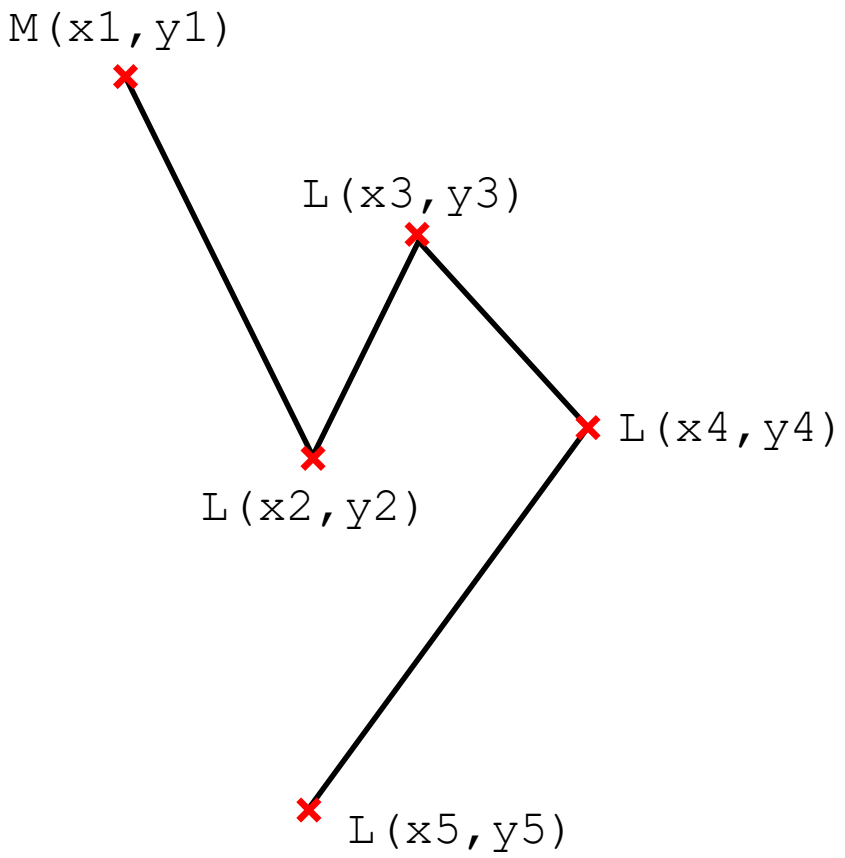








D3 Principles – Generic Shapes



Instruction	Attributes	Description
M	`x,y`	moves to x,y without drawing a line
L	`x,y`	adds a straight segment to x,y
H	`x`	adds a horizontal segment to x
V	`y`	adds a vertical segment to y
C	`x1,y1,x2, y2,x,y`	adds a cubic Bézier segment to x,y with control points x1,y1 and x2,y2
Z	–	closes a path

Some instructions of a path

D3 Principles – Attributes

- Attributes define properties of elements.

  Set: `d3.select("selector").attr("ATTRIBUTE", VALUE)`

  Get: `d3.select("selector").attr("ATTRIBUTE")`

- Different types of shapes have different attributes, e.g.:
  - `stroke`: stroke color (`red`, `#ff0000`),
  - `stroke-width`: width of a stroke (with or without unit, `1` vs `1px`),
  - `style`: CSS instructions for styling,
  - and many more.
- Attributes are element specific!

D3 Principles – Selections

- D3 operates on sets of nodes, which are returned by a selector.
- A selector matches DOM nodes(s) (HTML/SVG).

Selector	Match	D3 Code	Description
`#vis`	`<div id="vis"> </div>`	`d3.select("#vis")`	selects the element with id `vis`
`svg`	`<svg>…</svg>`	`d3.select("svg")`	selects all `svg` elements
`.rect`	`<rectangle class="rect"> </rectangle>`	`d3.select(".rect")`	selects all elements with class `rect`
`circle[r="100"]`	`<circle id="c" r="100"> </circle>`	`d3.select( "circle[r='100']" )`	selects all `circles` having an attribute `r` of value `100`

D3 Principles – Adding a Shape

select SVG

```
d3.select("svg")
```

add element of type line

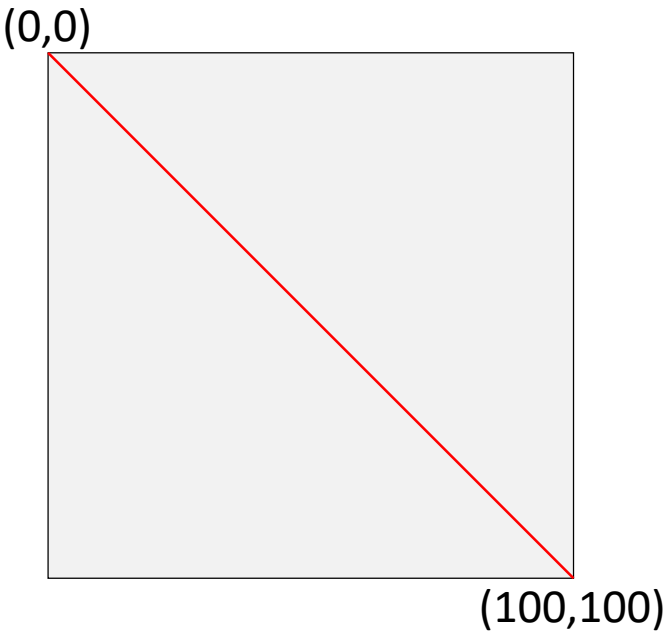
```
.append("line")
```

line attributes

```
.attr("x1", 0)
.attr("y1", 0)
.attr("x2", 100)
.attr("y2", 100)
```

set stroke color

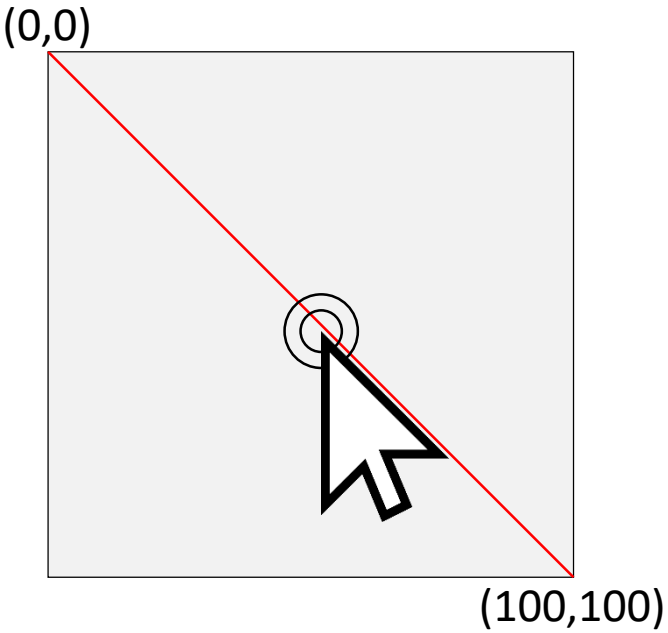
```
.attr("stroke", "red")
```



D3 Principles – Event Handling

```
d3.select("#d3vis")
    .append("line")
    .attr("x1", 0)
    .attr("y1", 0)
    .attr("x2", 100)
    .attr("y2", 100)
    .attr("stroke", "red")
    .on("click", function() {
      console.log(this);
    });
```

mouse click handler



All DOM Events are supported (*mouseout*, *mousover*, …).

D3 Principles – Animation

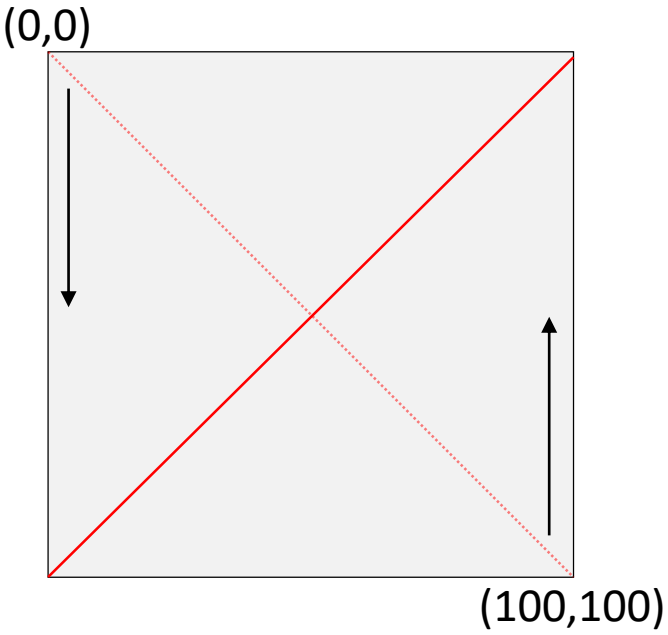
```
d3.select("line")
    .transition()
    .attr("y1", 100)
    .attr("y2", 0)
    .duration(1000);
```

start a transition

attribute `y1` interpolation

attribute `y2` interpolation

transition duration

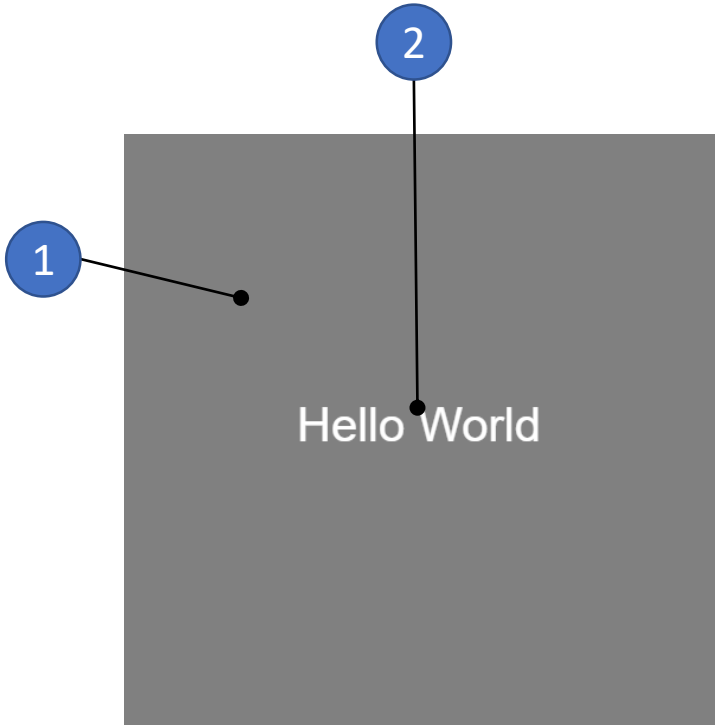


- Transitions interpolate attribute values.
- Multiple transition can be chained.
- Expose a lifecycle (`start`, `end`, `interrupt`).

D3 Principles – Misc

- Insertion order of elements determines drawing order.
- Elements can be grouped into groups (g element).

```
<svg width="500" height="500">
  <g id="background">
    <rect x="0" y="0" width="500" height="500" fill="black">
  </g>
  <g id="foreground">
    <text x="50%" y="50%"
      alignment-baseline="middle" text-anchor="middle"
      fill="white" font-family="Arial" font-size="40">Hello World</text>
  </g>
</svg>
```



D3 Principles – Misc

- There are a number of utilities for data handling.
  - aggregation, min/max computations, scaling, …
- D3 provides a data-driven update pattern:
  1. `data:` join data and graphics
  2. `enter`: opens a selection per data record
  3. `exit:` leaves the data record focus

```
this.vis.selectAll()
    //assign data
1   .data(dataPoints)
    //enter record scope
2   .enter()
        //create a circle
        .append("circle")
        //set circle center x
        .attr("cx", d => d.x)
        //set circle center y
        .attr("cy", d => d.y)
        //set circle radius
        .attr("r", 5);
```

3 `exit:` not required, instruction is terminated

JavaScript

- Script language interpreted by internet browsers.
- Available since more than 20 years.
- The driving force of interactive web applications.

```
var Car = function(brand, model) {
    this.brand = brand;
    this.model = model;
}

Car.prototype.drive = function(speed) {
    console.log(this.brand + " " + this.model + " drives " + speed + " mph");
}
```

Prototype-based inheritance

```
class Car {
    constructor(brand, model) {
        this.brand = brand;
        this.model = model;
    }

    drive(speed) {
        console.log(this.brand + " " + this.model + " drives " + speed + " mph");
    }
}
```

Equivalent, “modern“ syntax (ECMAScript 2015+)

```
var beetle = new Car("VW", "Beetle");
beetle.drive(55);

> VW Beetle drives 55 mph
```

Usage example

CSS

- Cascading Style Sheets is a markup language.
- Describes presentation of elements.
- Layout, colors, fonts, animations.

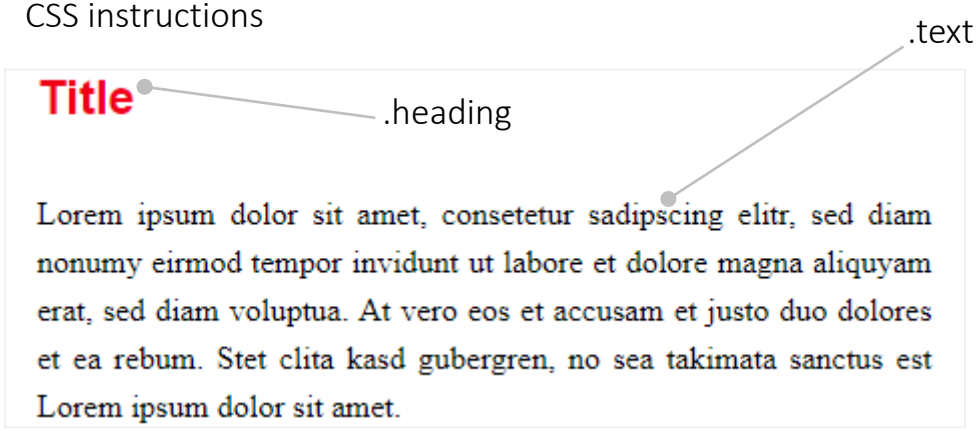
```html
<div class="heading">Title</div>
<p class="text">
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.
</p>
```

HTML code

```css
.heading {
    font-size: 1.5rem;
    font-weight: bold;
    color: red;
    margin-bottom: 2rem;
}

.text {
    font-family: 'Times New Roman', Times, serif;
    line-height: 1.5rem;
    text-align: justify;
}
```

CSS instructions

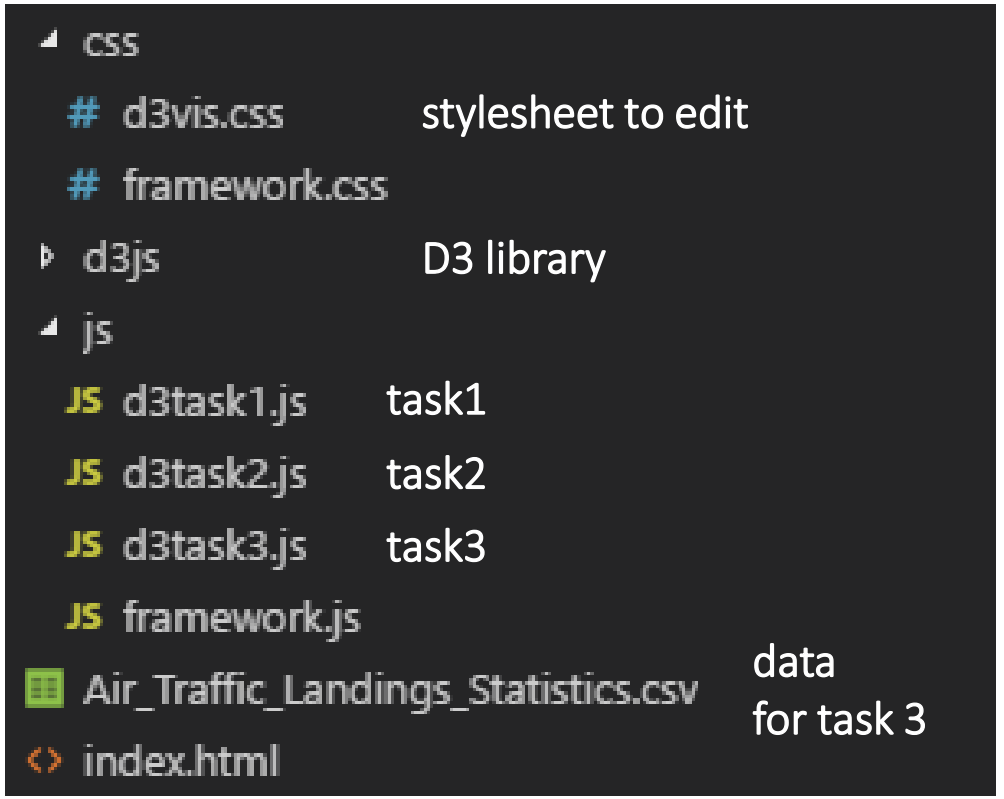


Rendered HTML.

D3 Hands-On

Overview

- Three different tasks.
- Each task is part of a separate JavaScript file (ECMAScript 2016).
- A framework takes care about initialization/control logic.

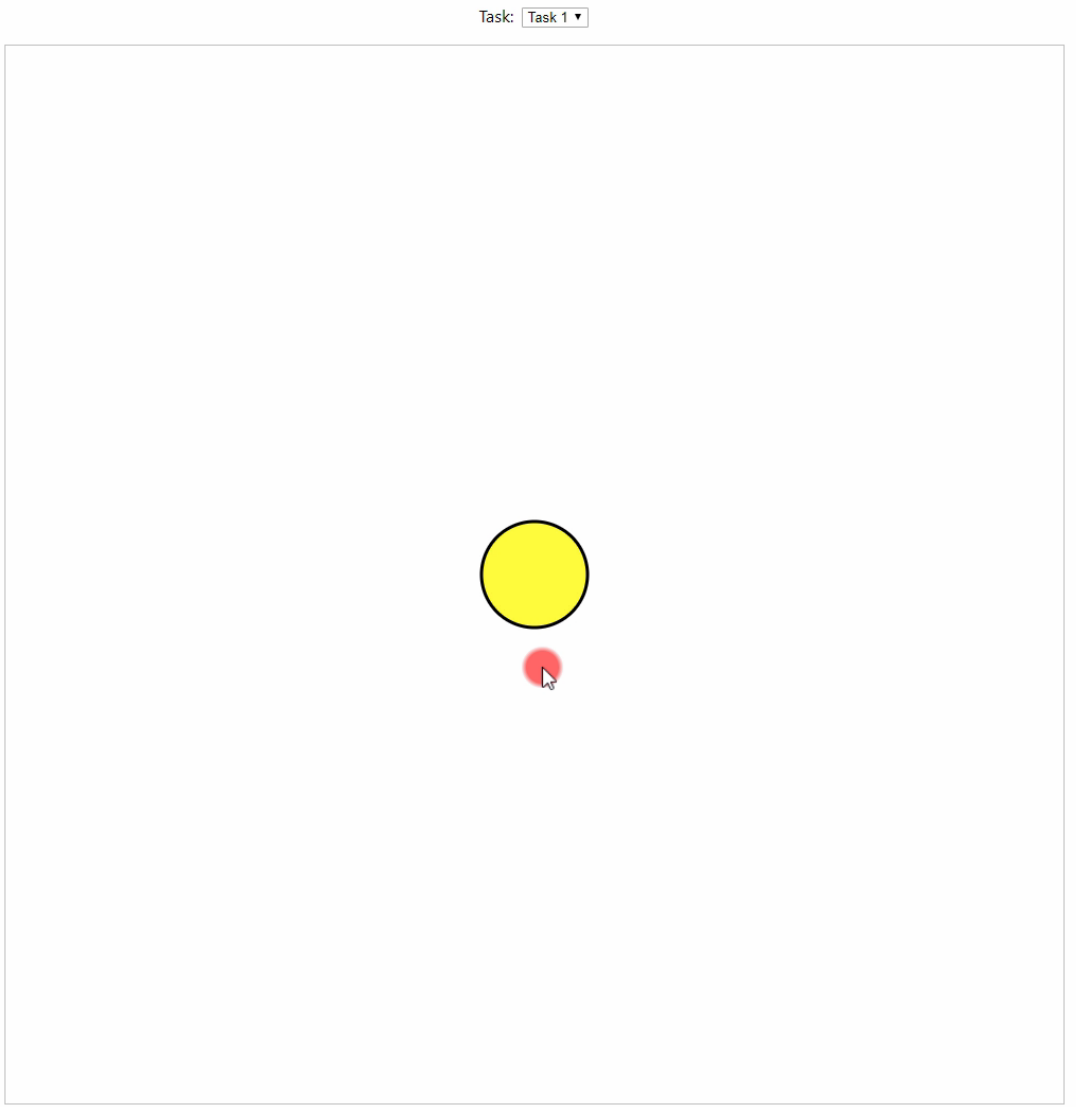


Task 1: Simple Shapes

- Create a shape and style it.
- Add interaction (effect on hover).
- Add some animation.

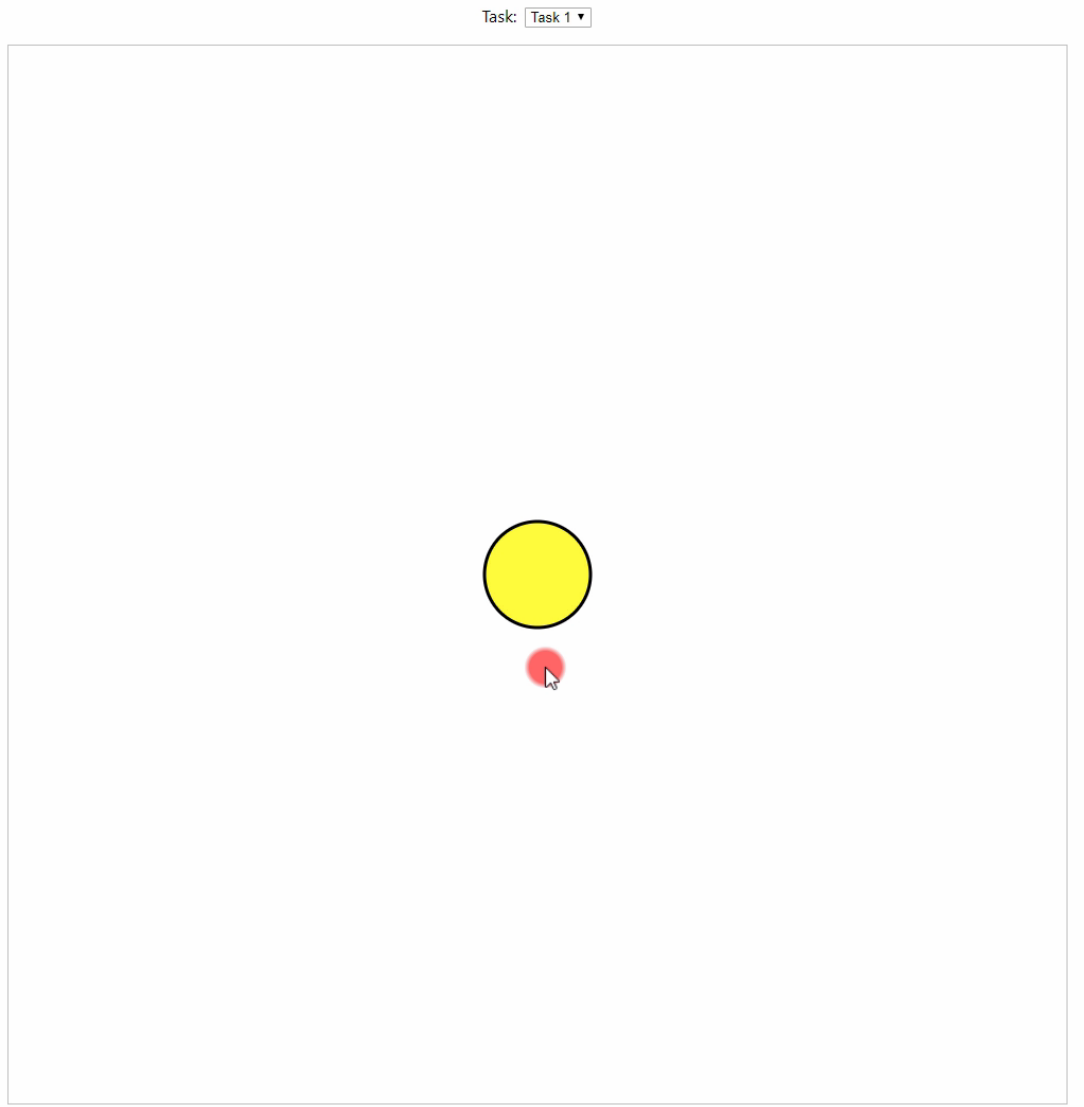
Source code:

CSS: `css/d3vis.css`
JS: `js/d3task1.js`



Task 1: Simple Shapes – Details

1. Append a shape to `this.vis`.
2. Set coordinates of the shape.
3. Style the shape.
4. Add interaction handlers (*`mouseover`*)
    1. Connect handler with an action.
5. Add an animation to the shape.

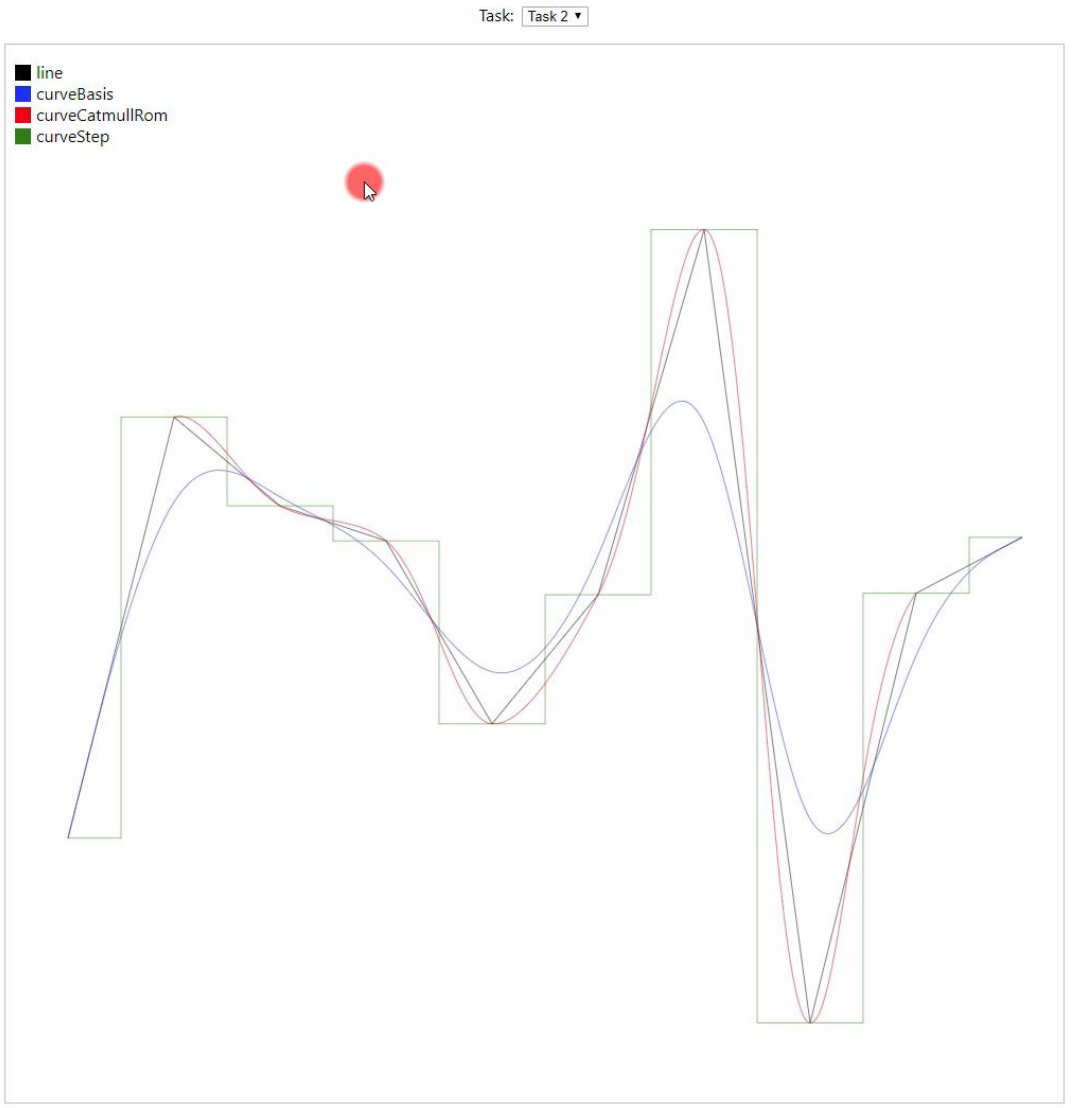


Task 2: Complex Paths

- Create and visualize complex paths (interpolated lines) using D3.
- Add switches to hide/show the paths.

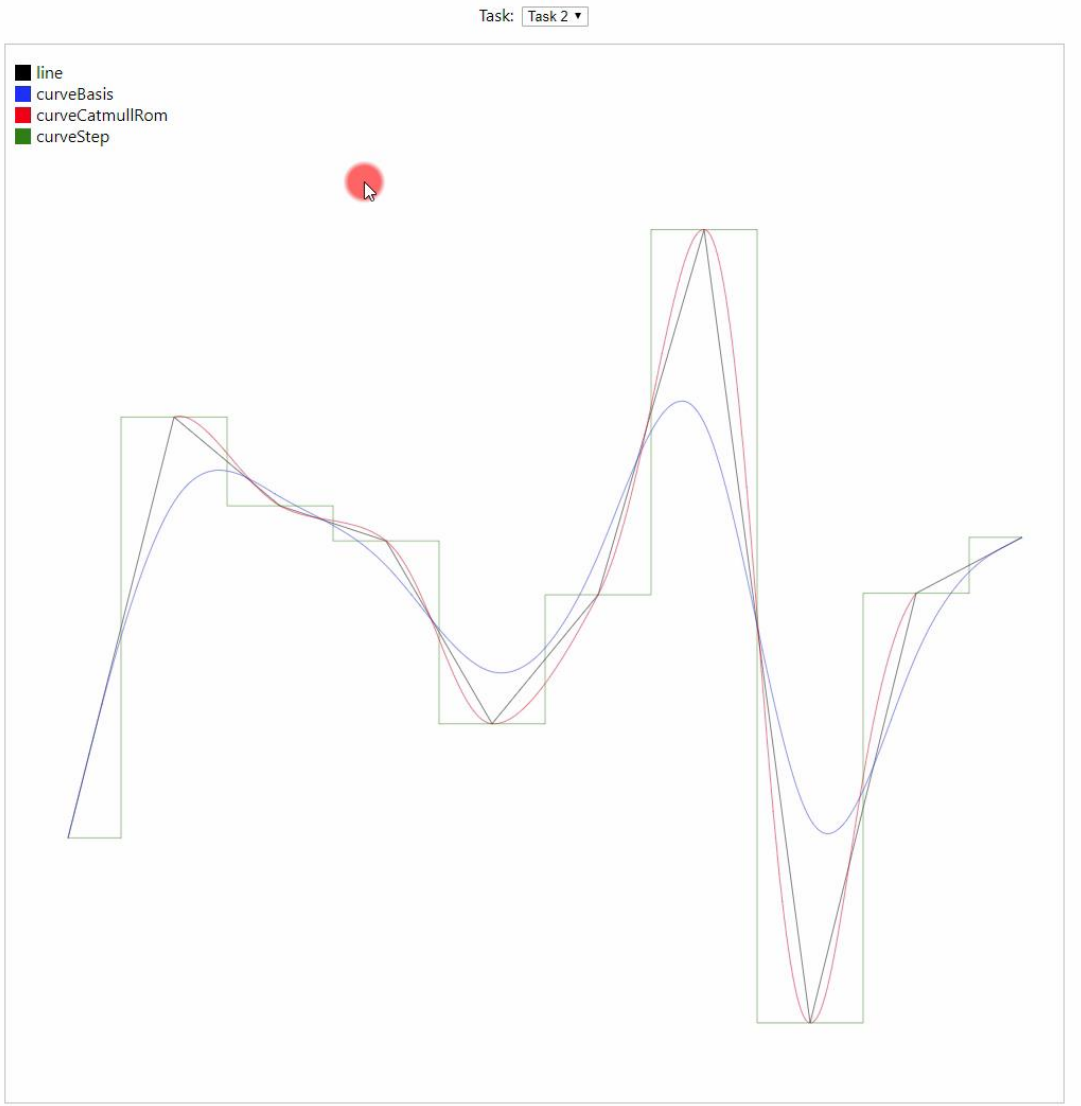
Source code:

CSS: `css/d3vis.css`
JS: `js/d3task2.js`



Task 2: Complex Paths – Details

1. Explore the D3 line/curve API.
2. Draw different interpolated using the given data (`d3.line()`, `this.data`).
3. Add a legend (append to `this.vislegend`).
4. Add interaction to legend (hide/show).

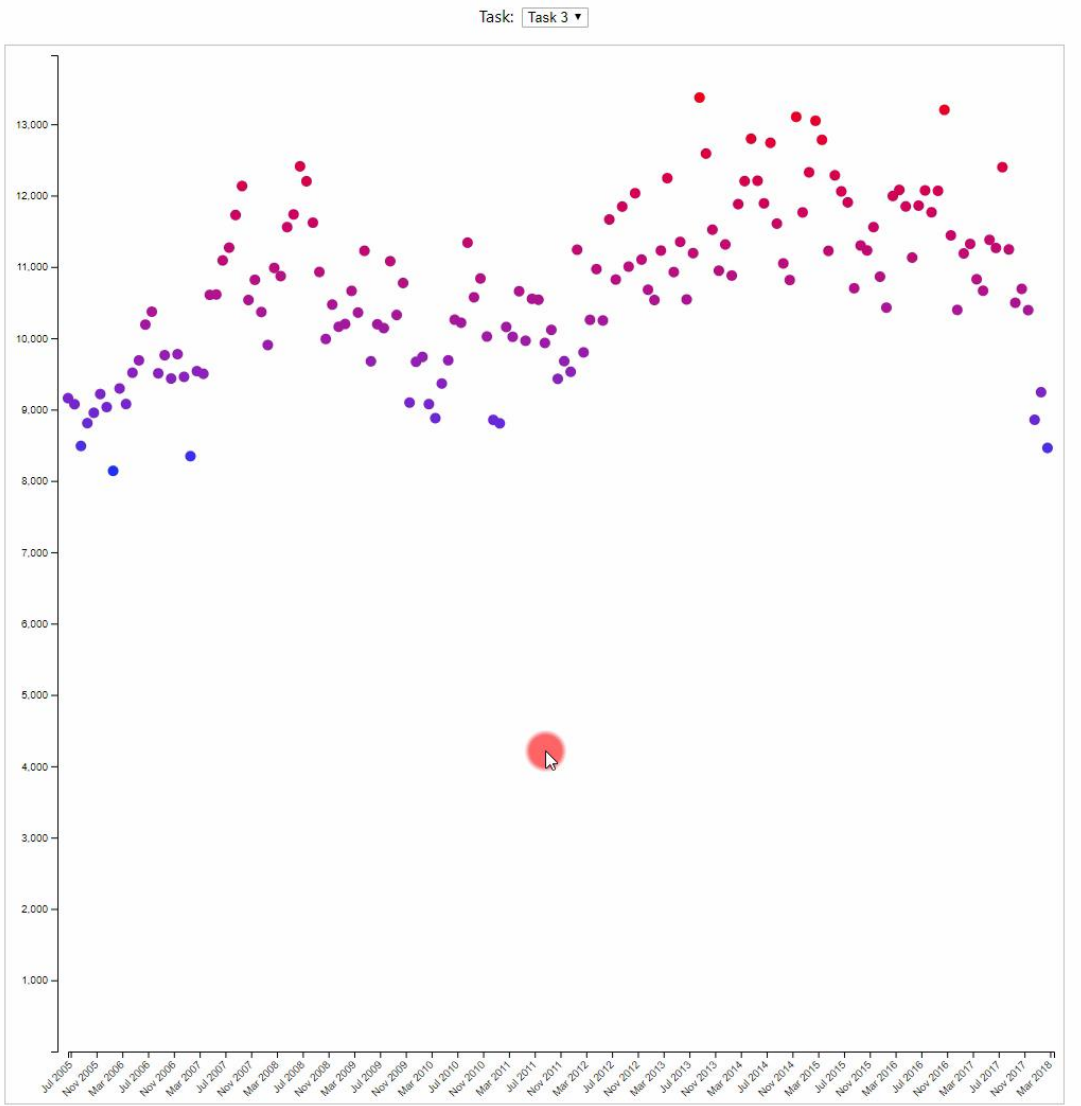


Task 3: Scatter Plot

- Visualize a dataset with a scatter plot.
- Add axis, and color mapping.

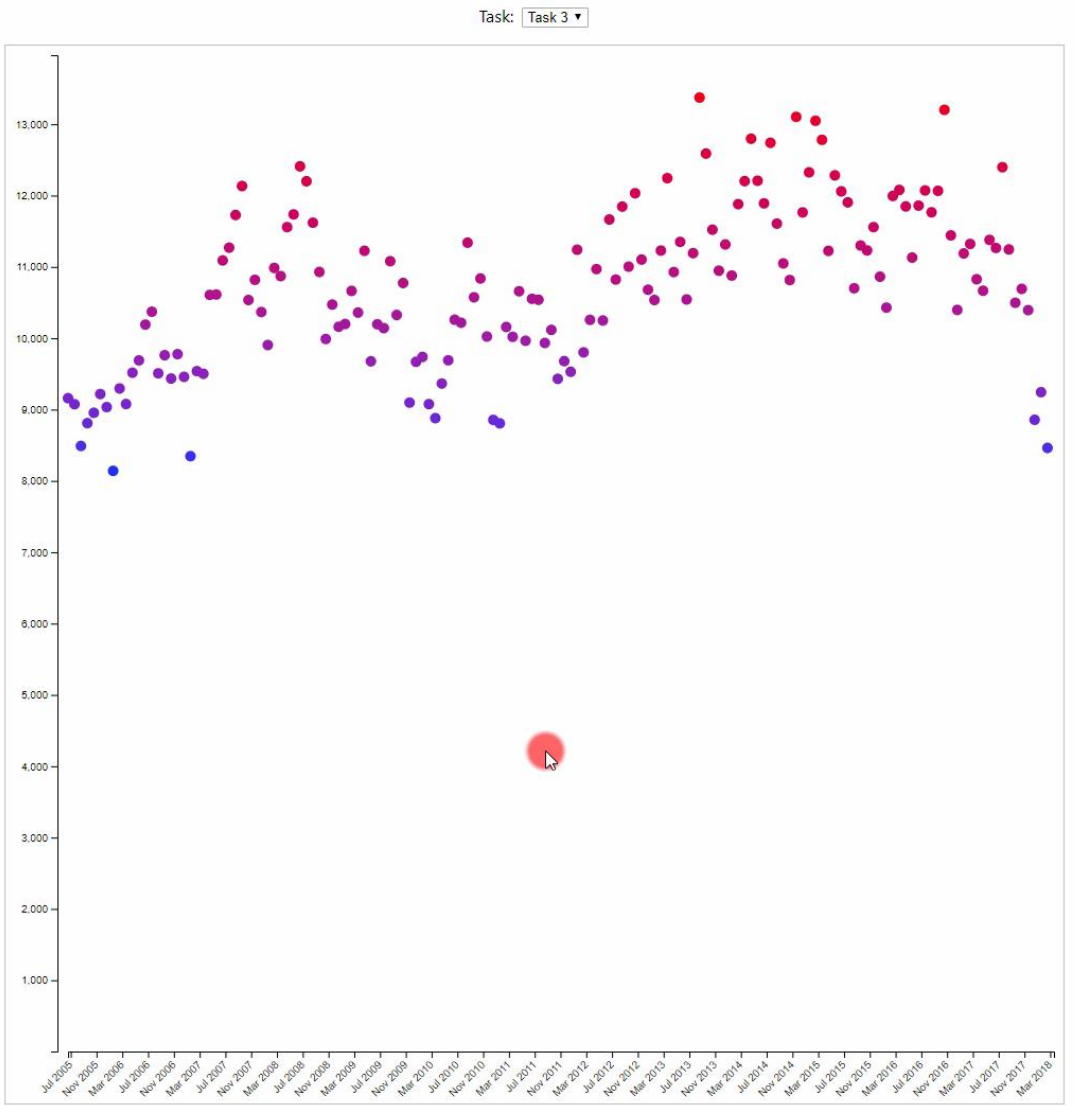
Source code:

CSS: `css/d3vis.css`
JS: `js/d3task2.js`



Task 3: Scatter Plot – Details

1. Complete the data loading logic, if required (`loadData()`).
2. Compute some statistics to visualize.
3. Append the data to the plot (`this.vis`).
4. Add colors to the visualized data.
5. Add a legend.



Some Interactive Examples

- Delaunay Triangulation
  https://bl.ocks.org/mbostock/4341156
- Chord Diagram
  http://bl.ocks.org/databayou/c7ac49a23c275f0dd7548669595b8017/d3c81c0153000595b0f76347952ba9510800a2f8
- Parallel Coordinates
  https://bl.ocks.org/timelyportfolio/e9ea7af871cc11e0f43f
- Choropleth Map
  http://bl.ocks.org/pmia2/ecf6cfd7239aad7ab987cee0b7b12281/ed908f67ec3ded77019add1eb679cd19489819af
- Fantasy Map Generator
  https://azgaar.github.io/Fantasy-Map-Generator/