

About

This prototype demonstrates that the data in `volume1-poster.pdf` can be loaded and manipulated by `d3.js`.

A vertical arc diagram (right) was chosen as the display format to quickly test that all people and links were present and correct.

The final result is the graphic on the right, showing people and color-coded relationships. Friendly relationships are muted, while hostile relationships stand out in bright red.

This prototype is a good step to more design-focused prototype, because the data tasks, like parsing, organizing, loading, and manipulating the data in `d3.js`, have already been figured out.



How it was done

(1) The `volume1-poster` data was converted to a single JSON file so it could easily be loaded by `d3.js`.

(2) People came from `volume1-poster.csv`, parsed by a simple nodejs script (see `get_people`).

Links came from hard-coded relationships in `volume1-poster.py` because they seemed to match the links in `volume1-poster.pdf` better than the link data in `volume1-poster.csv`.

(3) A `d3.js` script was loaded into Chrome from a simple local static web server. The `d3.js` script first made a request to the web server to load the JSON file containing the data.

(4) Then it created an SVG element in the DOM of the webpage.

(5) Finally the `d3.js` script inserted arcs and text elements for each person and link (code for arcs shown at right).

```
"people": [
  {
    "first": "Moritz",
    "last": "Lazarus",
    "born": "1824",
    "died": "1903",
    "profession": "unknown",
    "key": ""
  },
  {
    "first": "Heymann",
    "last": "Steinthal",
    "born": "1823",
    "died": "1899",
    "profession": "anthropologist",
    "key": ""
  }
],
"links": [
  {
    "from": "Herder",
    "to": "Fichte",
    "type": "teacher"
  },
  {
    "from": "Kant",
    "to": "Fichte",
    "type": "teacher"
  },
  {
    "from": "Langfeld",
    "to": "Allport",
    "type": "teacher"
  }
],
```

```
function get_people(file) {
  // Array to return
  let people = [];

  // Read the file into memory and split by lines
  // (This prototype assumes a small file size)
  let lines = fs.readFileSync(file, 'utf-8').split('\n');

  // Parse the person information from each line
  lines.forEach(function(line) {
    // Split the line by commas (csv)
    let parts = line.split(',');

    // Skip lines with not enough data. We need at
    // least ("P", first, last, born, died)
    if (parts.length < 5)
      return;

    // Skip lines that don't start with "P"
    // (Matches original/volume1-poster.py)
    if (parts[0] !== "P")
      return;

    // Create the person structure with defaults for
    // profession and key
    let person = {
      first: parts[1],
      last: parts[2],
      born: parts[3],
      died: parts[4],
      profession: "linguist",
      key: ""
    };

    // Set the profession if one is available
    if (parts.length >= 7)
      person.profession = parts[6];

    // Replace empty professions with "unknown"
    if (person.profession.trim() === "")
      person.profession = "unknown";

    // Set the key if one is available
    if (parts.length >= 8)
      person.key = parts[7];

    // Add to the output array
    people.push(person);
  });

  return people;
}
```

```
// Load and parse the json data
d3.json("data/all.json").then(function(genealogy) {
```

```
// Create the SVG
var svg = d3.select("#svg-container").append("svg")
  .attr("width", width)
  .attr("height", height);
```

```
// Make the arcs
const path = svg.insert("g", "*")
  .attr("fill", "none")
  .attr("stroke-opacity", 0.6)
  .attr("stroke-width", 1.5)
  .selectAll("path")
  .data(graph.links)
  .join("path")
  .attr("stroke", function(d) {
    switch (d.type) {
      case "teacher":
        return "#b2c2bd";
      case "influence":
        return "#6d93c7";
      case "postDoc":
        return "#65dbb7";
      case "hostile":
        return "#ff0000";
      default:
        return "#aaa";
    }
  })
  .attr("d", make_arc);
```