Learning D3.js Mapping

Build stunning maps and visualizations using D3.js

Thomas Newton
Oscar Villarreal
In this package, you will find:

- The authors biography
- A preview chapter from the book, Chapter 1 “Gather Your Cartographer’s Toolbox”
- A synopsis of the book’s content
- More information on Learning D3.js Mapping

About the Authors

**Thomas Newton** has 20 years of experience in the technical industry, working on everything from low-level system designs and data visualization to software design and architecture. Currently, he is creating data visualizations to solve analytical problems for clients. When he is free, he spends as much time as possible with his family, ideally with a skateboard involved.

**Oscar Villarreal** has been developing interfaces for the past 10 years, and most recently, he has been focusing on data visualization and web applications. In his spare time, he loves to write on his blog, oscarvillarreal.com, and go rock climbing somewhere in the Rockies. He also loves to spend time with his wife and his first child.
Learning D3.js Mapping

This book explores the JavaScript library, D3.js, and its ability to help us create maps and amazing visualizations. You will no longer be confined to third-party tools in order to get a nice looking map. With D3.js, you can build your own maps and customize them as you please. This book will go from the basics of SVG and JavaScript to data trimming and modification with TopoJSON. Using D3.js to glue together these three key ingredients, we will create very attractive maps that will cover many common use cases for maps, such as choropleths, data overlays on maps, and interactivity.

What This Book Covers

Chapter 1, Gather Your Cartographer's Toolbox, covers the tools you need to install to build maps effectively with D3. We will cover the instructions to add NodeJS, npm, and TopoJSON to your system. This is needed in order to quickly bootstrap a local web server and to have the ability to manipulate map data.

Chapter 2, Creating Images from Simple Text, describes the basics of creating multiple shapes and designs with Scalable Vector Graphics (SVG).

Chapter 3, Producing Graphics from Data – the Foundations of D3, will deep dive into the main library of this book, D3.js. Concepts such as enter, update, and exit will be explained in detail with the use of some clever experiments and examples.

Chapter 4, Creating a Map, takes you through every step required to build a map with D3. Asynchronous loading of data, bounding box and projection creation, and TopoJSON element modification and data binding will be the bread and butter of this chapter.

Chapter 5, Click-click Boom! Applying Interactivity to Your Map, covers hover, panning, zooming, and their equivalent in touch events. Also, we'll dive into some nifty 3D projections, and a la par, we will learn transitions via experimentation.

Chapter 6, Finding and Working with Geographic Data, will take you through the process of acquiring data and optimizing it to your needs. This chapter covers one of the most important aspects of map visualization in detail via the use of TopoJSON.

Chapter 7, Testing, explains how to structure your codebase in order to have reusable chart components that are easily unit tested and primed for reuse in future projects.
Welcome to the world of cartography with D3. In this chapter, you will be given all the tools you need to create a map using D3. These tools exist freely and openly, thanks to the wonderful world of open source. Given that we are going to be speaking in terms of the Web, our languages will be HTML, CSS, and JavaScript. After reading this book, you will be able to use all three languages effectively in order to create maps on your own.

When creating maps in D3, your toolbox is extraordinarily light. The goal is to focus on creating data visualizations and remove the burden of heavy IDEs and map-making software. The building blocks are as follows:

- **Quick bootstrap**: We will cover installing Node.js, npm, and a lightweight server.
- **TopoJSON**: We will see the working of this tool that is used to manage and optimize geographic information.
- **Web browser as a development tool**: We will learn how to use a modern web browser, capable of rendering SVG files with built-in development tools. Some examples of common web browsers are Chrome, Firefox, Safari, and IE 9+.

   All screenshots and commands in this book have been done using Chrome. Therefore, we strongly recommend you to use this browser.

- **Installing sample code**: We will take the first steps in getting to grips with coding.
- **Working with developer tools**: We will get familiar with developer tools.
Quick bootstrap

If you’re already familiar with working in D3 or web development, go ahead and set up your workstation based on the following commands by jumping directly to the Step-by-step bootstrap section.

A detailed explanation of this chapter’s concepts will be covered if further instructions are needed. By the end of the chapter, you should feel comfortable in using the Quick bootstrap section as a convention throughout the book. The following instructions assume that Node.js and npm are already installed on your system.

Type in the following in the command line:

```bash
# Install a light webserver
npm install -g http-server

# Install topojson
npm install -g topojson

# Clone the sample code with included libraries
git clone --depth=1 git@github.com:climboid/d3jsMaps.git

# start the project
cd d3jsMaps
http-server
```

Now, open your web browser to http://localhost:8080/chapter-1/example-1.html, and you should see the following map:
Step-by-step bootstrap
The next section covers detailed instructions to set up your development environment to rapidly create visualizations in later chapters. By the end of the chapter, you will have a working environment for the rest of the book (an example of a map running and an initial look at tools used to create visualizations).

A lightweight web server
Technically, most of the content we will craft can render directly in the browser without the use of a web server. However, we highly recommend you not to go ahead with this approach. Running a web server in your local development environment is extremely easy and provides several benefits:

- Geographic information, statistical data, and visualization code can be clearly separated into isolated files
- API calls can be stubbed and simulated, allowing easy integration into a full-stack application in the future
- It will prevent common mistakes when making AJAX calls to fetch geographic and statistical data (for example, the same-origin policy)
Gather Your Cartographer's Toolbox

For our choice of the web server and other tools in our toolbox, we will rely on a Node.js package named http-server. Node.js is a platform built on Chrome’s JavaScript runtime, which is used to build fast, scalable network applications. The platform includes Node Package Manager (npm), which was created by other members of the vibrant Node.js community and allows the developer to quickly install packages of prebuilt software.

To install Node.js, simply perform the following steps:

2. Click on the INSTALL button.
3. Open the downloaded package and follow the defaults.

To test the installation, type in the following in the command line:

```
node –v
```

v0.10.26 # or something similar should return

Now that Node.js and npm are installed, we can proceed to install the web server. When the web server is installed globally with the –g command, it becomes easily accessible throughout the system.

Once on the command line, enter the following:

```
npm install -g http-server
```

You can quickly verify the package was installed by typing the following:

```
http-server
```

Then open your web browser to http://localhost:8080. The browser should display a list of all the files in the directory from where you initiated the command.

Optimizing geographic data files with TopoJSON

The topojson is a command-line utility used to create files in the TopoJSON-serialized format. The TopoJSON format will be discussed in detail in Chapter 6, Finding and Working with Geographic Data. The topojson utility is also installed via npm.

Note that we will use the syntax TopoJSON to refer to the JSON format and topojson to indicate the command-line utility.
We already have installed Node.JS and npm, so enter the following on the command line:

```
npm install -g topojson
```

Once the installation is complete, you should check the version of topojson installed on your machine just as we did with Node.js:

```
topojson -version
```

If you see Version 1.x, it means you have successfully installed topojson.

topojson uses node-gyp that has several dependencies based on the operating system. Please go to http://github.com/TooTallNate/node-gyp for details.

If you're using Windows, the basic steps are as follows:
- Install Python 2.x (3.x not supported at the time of writing this book)
- Install Microsoft Visual Studio C++ 2012 for Desktop (Express)

### Using the web browser as a development tool

Although any modern browser supports SVG and has some kind of console, we strongly recommend you to use Google Chrome for these examples. It comes bundled with developer tools that will allow you to very easily open, explore, and modify the code. If you are not using Google Chrome, please go to http://www.google.com/chrome and install Google Chrome.

### Installing the sample code

Go to https://github.com/climboid/d3jsMaps and either clone the repo, if you are familiar with Git cloning, or simply download the zipped version. Once it is downloaded, make sure to extract the file if you have it zipped.

Use the command prompt or terminal to go to the directory where you downloaded your file. For instance, if you downloaded the file to your desktop, type in the following:

```
cd ~/Desktop/d3jsMaps
http-server
```
The last command will launch the simple server we installed previously for the supplied sample code. This means that, if you open your browser and go to http://localhost:8080/chapter-1/example-1.html, you should see a map of Europe, similar to the one shown earlier.

**Working with the developer tools**

It's time to open the developer tools. On the top-right corner of the browser, you will see the icon as shown in the following screenshot:

![Developer Tools Icon](image)

This icon opens a submenu. Click on **Tools**. Then click on **Developer tools**. A panel will open at the bottom of the browser, containing all the developer tools at your disposal.

The option names mentioned here might differ according to the version of Chrome you are using.
For quick access to developer tools on the Mac, use `alt + command + I`; for Windows PCs, use `Ctrl + Shift + I`.

Within developer tools, you have a series of tabs (Elements, Network, Sources, and so on). These tools are extremely valuable and will allow you to inspect different aspects of your code. For more information on the Chrome developer tools, please go to the link https://developer.chrome.com/devtools/docs/authoring-development-workflow.

Since we are going to focus on the Elements tab, click on it if it is not already selected.

You should see something similar to the preceding screenshot; it will have the following code statement:

```
<svg width="812" height="584">
```
If you click on the SVG item, you should see it expand and display the path tag. The path tag will have several numbers and characters tied to a `d` attribute. These numbers are control points that draw the path. We will cover how the path is drawn in the next chapter and how path tags are used to create maps in Chapter 4, *Creating a Map* and Chapter 5, *Click-click Boom! Applying Interactivity to Your Map*.

We also want to draw your attention to how the HTML5 application loads the D3 library. Again in the **Elements** tag, after the SVG tag, you should see the `<script>` tag pointing to D3.js and topojson:

```html
<script src="http://d3js.org/d3.v3.min.js" charset="utf-8"></script>
<script src="http://d3js.org/topojson.v1.min.js"></script>
```

If you click on the path located inside the SVG tag, you will see a new panel called the CSS inspector or the styles inspector. It shows and controls all the styles that are applied to a selected element—in this case the path element.

These three components create a D3 visualization:

- HTML5 (the SVG and path elements)
- JavaScript (the D3.js library and map code)
- CSS (the styling of the HTML5 elements)

Creating maps and visualizations using these three components will be discussed and analyzed throughout the book.

**Summary**

This chapter reveals a quick glimpse of the steps for basic setup in order to have a well-organized code base to create maps with D3. You should become familiar with this setup because we will be using this convention throughout the book.

The remaining chapters will focus on creating detailed maps and achieving realistic visualizations through HTML, JavaScript, and CSS.

Let's jump right in!
Where to buy this book

You can buy Learning D3.js Mapping from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.