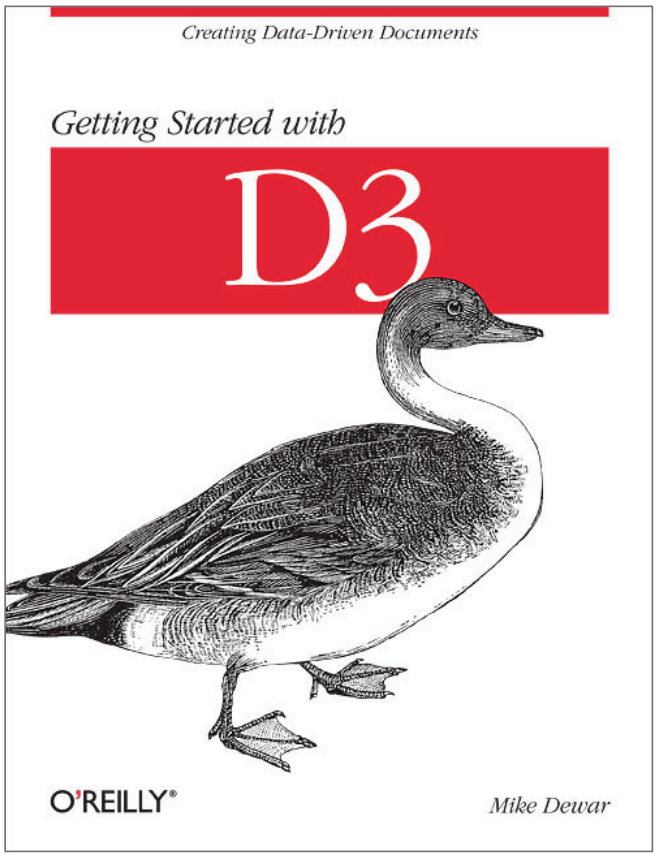

Hanbit eBook
Realtime 08
자바스크립트와 SVG로 쉽게 만드는
웹 기반 데이터 비주얼라이제이션 D3
Getting Started with D3
마이크 드워 지음 / 양성일 옮김
O'REILLY
한빛미디어
Hanbit Media, Inc.



이 도서는 O'REILLY의
Getting Started with D3의
번역서입니다.

자바스크립트와 SVG로 쉽게 만드는

웹 기반 데이터 비주얼라이제이션 D3

지은이_ **마이크 드워**
마이크 드워는 단축 URL 주소를 만드는 뉴욕의 기술 회사 Bitly의 데이터 과학자다. 영국 셰필드 대학교University of Sheffield에서 데이터의 동적 시스템 모델링 분야의 박사 과정을 수료하였고, 에든버러 대학과 콜롬비아 대학에서 기계 학습의 초빙 연구원으로 일했다.

옮긴이_ **양성일**
양성일은 13년간 자바 프로그래머로 기업 솔루션, 홈네트워크, 전자상거래, 검색엔진, 보안, 스마트 폰 앱 등 다양한 분야의 소프트웨어를 개발하였으며, 2010년 11월에 허그루를 창업하여 현재 소프트웨어 프리랜서 활동하고 있다. ADK, 아두이노 보드를 기반으로 한 오픈소스 하드웨어 플랫폼 개발에 전념하고 있으며 관련 지식과 정보를 여러 사람들과 공유하기 위한 하드로이드 커뮤니티(http://www.hardroid.net)를 운영하고 있다.

자바스크립트와 SVG로 쉽게 만드는 **웹 기반 데이터 비주얼라이제이션 D3**

초판발행 2012년 11월 16일

지은이 마이크 드워 / **옮긴이** 양성일 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-7914-977-7 15560 / **정가** 9,900원

책임편집 배용석 / **기획** 김창수 / **편집** 김병희, 안선화
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 김현미
영업 김형진, 김진불, 조유미 / **마케팅** 박상용, 박주훈, 정민하

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanb.co.kr / **이메일** ask@hanb.co.kr

지금 하지 않으면 할 수 없는 일이 있습니다.
책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanb.co.kr)로 보내주세요.
한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

저자 서문

자바스크립트 라이브러리 D3는 웹 브라우저 기반의 양방향성 시각화를 만들 수 있도록 지원한다. D3를 이용하면 데이터 집합의 문맥을 웹 페이지의 기본 요소로 사용하여 시각화를 완벽하게 구현할 수 있다. 이 책의 목적은 이러한 D3의 기능을 독자가 쉽고 빠르게 학습하여 활용할 수 있도록 돕는 것이다.

데이터 집합을 시각화하면서, D3의 새로운 개념을 설명할 것이다. 책에서 언급하는 예제는 뉴욕교통청과 뉴욕 운송 시스템에서 공개한 이력 구성 표와 데이터 스트림, 지역 정보를 통해 수집한 실제 자료를 이용했다. 책의 말미에는 D3의 핵심 관점 몇 가지를 살펴보고, 웹에서 상호작용하는 데이터 시각화를 적절하게 구축해 볼 것이다.

저자 **마이크 드워**

감사의 글

먼저 도움과 의견을 주고 훌륭한 라이브러리를 사용할 수 있게 해준 마이크 보스탁에게 감사드린다. 기술적인 조언과 함께 영국 영어를 미국 영어로 변경하는 데 큰 도움을 준 나의 훌륭한 친구들을 비롯하여, 동료인 브라이언 에오프, 존 마일스 화이트, 드류 콘웨이, 맥스 쉬론, 가브리엘 가스터에게 감사드린다. 그리고 고함 없이 이 책을 낼 수 있도록 도움을 준 매우 유능한 편집자인 메건 블란케테에게도 감사드린다. 무엇보다 사랑과 인내 그리고 아낌없는 지원을 해준 나의 약혼자 모니카 배킬에게도 고맙다는 말을 전하고 싶다.

역자 서문

지금까지 많은 기업이 수집한 데이터를 어떻게 활용할 것인지 고민해왔다. 그 과정에서 데이터 웨어하우스, 데이터 마이닝, BI 등 많은 솔루션이 등장하였다. 바야흐로 사람이 감당할 수 없을 정도의 정형 혹은 비정형 데이터가 홍수처럼 쏟아지는 시대가 도래한 만큼 기업들은 하둡, 맵리듀스 등 연관 솔루션을 활용하여 기업이 필요로 하는 정보를 추출해서 비즈니스에 적용하기 위해 노력하고 있다.

빅 데이터는 다양한 종류의 대규모 데이터 생성, 수집, 분석, 표현으로 분류되어 발전하고 있다. D3는 빅 데이터 표현 기술에 대한 것이다. 물론 이전에도 ActiveX, Flash, Java를 이용한 시각화 솔루션들이 있었지만, 별도의 플러그인이 필요한 비표준 방식이었다. HTML5가 나오면서 Canvas, SVG를 모든 메이저 브라우저에서 지원하고 있는 현시점에, 자바스크립트 라이브러리 D3는 별도의 플러그인 없이 다양한 시각화를 웹 환경에서 표현할 수 있는 좋은 도구다.

이 책은 D3가 무엇이고 어떻게 사용하는지를 소개한다. 예제를 따라 하다 보면 시각화 표현 도구로서 D3가 얼마나 유용하고 익히기 쉬운지를 알게 될 것이며, D3를 실질적으로 프로젝트에 도입하여 응용하는 데도 많은 도움이 될 것이다.

이렇게 좋은 도서를 번역할 수 있게 기회를 준 한빛미디어 관계자분들에게 감사드리며, 어려움 속에서도 언제나 힘을 주는 사랑하는 아내 홍영미에게 감사하다는 인사말을 남기고 싶다.

번역을 마치며

양성일

대상 독자

초급 | 초중급 | **중급** | 중고급 | 고급

이 책은 웹 브라우저에서 지원하는 기능을 활용하여 데이터를 시각화하고 싶은 개발자를 대상으로 한다. 뿐만 아니라 인쇄된 기사의 한계에서 벗어나고 싶은 대학생, 회사에서 제공한 자료 중 인상 깊었던 결과들을 공유하고 싶은 개발자, 인터넷에 널리 퍼져 있는 외부 정보에서 유용한 정보를 얻기 원하는 디자이너들에게도 도움이 될 것이다.

때문에 독자가 이미 데이터를 만들고 분석하는 방법에 익숙하다고 가정한다. 즉, 데이터를 분석하고 모델링하는 것과 기본적인 안목을 충족시키는 미학적인 관점은 논하지 않고, 오직 자바스크립트와 SVG를 이용하여 시각화하는 데만 중점을 둘 것이다.

책에서 설명하는 토픽에 대해 자세히 알고 싶으면 드류 콘웨이와 존 마일즈 화이트가 쓴 『Machine Learning for Hackers』(오라일리, 2012년), 더글라스 크락포드가 쓴 『더글라스 크락포드의 자바스크립트 핵심 가이드(JavaScript: The Good Parts)』(한빛미디어, 2008년), 데이비드 아이젠버그가 쓴 『SVG Essentials』(오라일리, 2002년), 벤 프라이가 쓴 『Visualizing Data』(오라일리, 2007년)를 참조하기 바란다.

예제 파일

이 책에서 설명하는 예제들은 실무에 적용할 수 있도록 구성하였기 때문에, 소스 코드를 프로그래밍과 문서에 바로 적용할 수 있습니다. 예제 파일은 “http://examples.oreilly.com/0636920025429/”에서 받을 수 있습니다.

한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook 입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1. **eBook Only - 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.**

   500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내시는 선배, 전문가, 고수분에게는 보다 쉽게 집필하실 기회가 되리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 준비 중이며, 조만간 선보일 예정입니다.

2. **무료로 업데이트되는, 전자책 전용 서비스입니다.**

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정한 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나, 저자(역자)와 독자가 소통하면서 보완되고 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3. 독자의 편의를 위하여, DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT기기에서 자유롭게 활용하실 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해, 독자 여러분이 언제 어디서 어떤 기기를 사용하시더라도 편리하게 전자책을 보실 수 있도록 하기 위함입니다.

4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

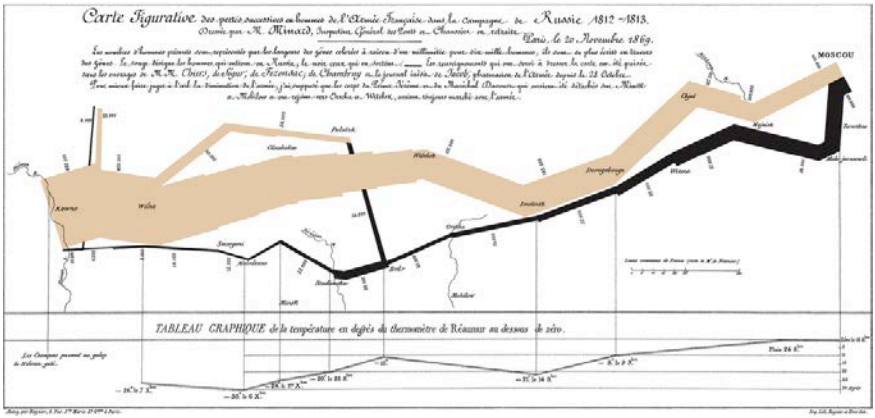
앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

차례

1 | D3 소개

수집한 데이터를 시각화하는 것은 이미 오래된 일이다. 한 예로, 143년 전 미나드Minard가 작성한 그림 1-1의 흐름도는 나폴레옹 군의 3월 행적을 보여준다. 이처럼 우리는 오랫동안 어떤 방식으로든 데이터를 수집하여 시각화하고 분석해왔다. 최근에는 우리가 수집할 수 있는 한계치를 넘을 정도로 데이터가 한없이 증가함에 따라 대용량 데이터를 수집하여 분석할 수 있도록 능력을 개발하는 작업이 계속되고 있다. 최신 웹 브라우저와 결합된 인터넷은 실시간으로 수백만 명의 사용자와 상호작용하는 그래픽을 제공하는 차세대 시각화 기술을 통해 아주 환상적인 사용자 경험을 제공한다.

그림 1-1 모스크바 행군에 따른 나폴레옹 군의 규모 감소를 나타낸 미나드의 흐름도(1869년 11월 20일에 다리와 도로 검시 담당관인 미나드가 작성함)



자바스크립트는 현재 대부분의 웹 브라우저에서 기본으로 지원하는 언어다. 즉, 사용하는 컴퓨터에 웹 브라우저가 설치되어 있다면 자바스크립트도 함께 설치되어 있다는 의미다. 또한 플래시가 지원되지 않는 모바일 기기를 포함하여 거의 모든 웹 브라우저는 SVGScalable Vector Graphics를 지원한다(2011년 인터넷 익스플로러 9가

소개된 이후).

자바스크립트와 SVG를 이용하면 인터넷 사용자가 웹에서 볼 수 있는 대부분의 복잡한 차트를 만들 수 있지만, 새로운 요소와 규칙을 익혀야 할뿐더러 이를 이용해 차트를 구현하는 것은 복잡한 작업이다. 자바스크립트와 SVG를 조합하는 이러한 복잡한 작업을 단순하게 구현할 수 있도록 도와주는 것이 바로 D3다.

1.1 D3의 특징

D3는 마이크 보스탁Mike Bostock이 만든 자바스크립트 라이브러리로, 사용이 쉬운 시각화 도구인 프로토비스Protovis를 상속하였다. D3 라이브러리는 데이터 집합의 문맥 안에서 HTML, SVG, Canvas 같은 웹 페이지의 요소를 데이터에 따라 보여주고, 삭제하며, 능숙하게 편집할 수 있도록 지원한다. 예를 들어, 분산 그래프를 만들 때 D3는 SVG circle 요소를 사용한다. 이를 통해서 circle 요소의 중심점인 cx와 cy 속성 값을 자연수 단위로 된 데이터 집합의 요소 x와 y 속성 값으로 적절하게 조정하여, 픽셀 크기의 보기 좋은 그래프를 보여준다.

D3의 최대 장점은 새로운 언어를 습득하지 않고도 웹 브라우저가 지원하는 여러 기술을 사용할 수 있다는 것이다. 즉, CSS 선택자를 사용하는 것처럼 요소들을 선택하고(jQuery 사용자들은 D3와 jQuery가 매우 유사하다는 것을 알게 될 것이다), 일반적인 CSS를 사용하여 스타일을 정의할 수 있다. 설계자는 또한 현재 웹 개발에 사용되는 파이어폭스의 파이어버그[01]와 크롬의 개발자 도구[02] 같은 툴들을 사용하여 쉽게 디버깅할 수 있다.

설계자와 웹 페이지 사이에 래퍼wrapper를 배치하는 전통적인 시각화 툴킷toolkit과 달리, D3는 축 생성, 그래프 표현, 코드 다이어그램을 레이아웃하는 일상적인

01 http://getfirebug.com
02 https://developers.google.com/chrome-developer-tools

작업을 단순화할 수 있도록 관련 작업을 지원하는 함수를 제공하는 데 초점이 맞춰져 있다. 즉, D3를 이용하면 설계자는 다양하고, 현대적 감각에 맞게, 또한 쉽게 데이터 시각화를 표현할 수 있다.

1.2 기본 설정

D3 라이브러리는 "http://d3js.org"에서 다운받을 수 있다. 책에서 설명하는 각 예제는 HTML이 포함된 디렉터리에 d3.js 파일이 있는 것으로 가정한다. 또한 모든 예제는 예제 1-1처럼 일반적인 HTML, 자바스크립트 구조를 따른다.

예제 1-1 이 책에서 사용하는 기본 페이지 설정

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
    <script src="d3.js"></script>  ❶
    <script>
    function draw(data) {  ❷
      "use strict";  ❸
      // 시각화 소스 코드는 여기에 위치함
    }
    </script>
</head>
<body>
  <script>
    d3.json("data/some_data.json", draw);  ❹
  </script>
</body>
</html>
```

① 시각화를 제공하는 D3 메서드에 접근하기 위해서는 항상 D3 라이브러리를 script 태그로 포함해야 한다.

② 책에서 다루는 예제들은 우리가 매번 호출할 'draw 함수'에 초점이 맞춰질 것이다. 이 함수는 인자를 하나 가지고 있는데, 클라이언트가 데이터를 다운받을 때 데이터가 인자에 할당되어 호출된다. 함수에는 시각화를 생성하는 데 필요한 소스 코드가 포함되어 있다.

③ D3는 형식이 잘 갖춰진 자바스크립트로 작성할 수 있도록, 캐스케이드cascade 자바스크립트 코딩 스타일을 권장한다. "use strict"; 부분은 자바스크립트 규칙을 엄격히 해석strict interpretation하는 strict 모드로 웹 브라우저를 설정한 것으로, 깔끔한 소스 코드를 작성하고 혼란스러운 버그를 피하기 위한 것이다(JSLint 툴[03]을 이용하여 예제에서 사용한 모든 자바스크립트를 검증하였다).

④ d3.json 함수는 첫 번째 인수인 URL(data/some_data.json)을 통해 HTTP GET으로 JSON 파일을 요청하고 데이터를 다운받는다. 다운로드가 완료되면 두 번째 인수로 전달한 함수를 호출한다. 이 책에서는 두 번째 인수인 콜백 함수에 항상 draw를 사용할 것이다. 매개변수로 전달받은 JSON은 객체 혹은 배열로 적절하게 변환하여 시각화하는 데 이용한다. D3는 XML과 CSV도 지원하지만 이 책에서는 JSON으로만 처리한다.

이 책은 독자가 시각화를 구축하는 과정에 관심을 갖도록 기술하였다. 즉, 처음 몇 단계에서는 지저분한 결과가 나오게 했으며 순차적으로 형태를 다듬어 차츰 페이지를 개선하는 것을 보여줄 것이다. 예제에서 사용한 CSS는 시각화 요소들을 알아본 후에 설명한다.

1.3 뉴욕교통청의 데이터 집합

뉴욕은 매우 크고, 항상 많은 사람들로 붐비는 밀집된 도시로, 이곳의 복잡한 교통

03 http://www.jslint.com/

네트워크는 뉴욕교통청(이하 MTA로 표기함)[04]에서 관리하고 있다. MTA는 자치구 다섯 군데와 주변 지역에서 하루에 천백만 명 이상의 유동 인구가 이용하는 로컬 기차, 지하철, 버스, 다리, 터널을 관리한다.

MTA는 교통 네트워크의 흐름과 관련 있는 아주 많은 양의 유용한 데이터를 만들어 공개한다. 이 데이터를 적극적으로 활용하는 개발자 커뮤니티가 형성되어 있으며, 이들은 세금으로 운영되는 교통망을 뉴욕 시민이 좀 더 효과적으로 이용할 수 있도록 교통망 기능 개선을 돕는다.

MTA에서 공개한 이 데이터를 책의 예제로 사용할 것이다. 각 예제마다 MTA가 공개한 데이터 파일을 하나 이상 사용할 것인데, 데이터 종류에 대해서는 "http://www.mta.info/developers/"를 참조하면 된다. 이 웹 사이트에서 받은 파일은 최신 데이터 집합을 제공해줄 뿐만 아니라, 데이터를 통해 정형화된 사용자 그룹을 추출할 수 있는 훌륭한 자원이다.

1.3.1 데이터 정제[05]

예제에 사용한 소스 코드는 두 개의 디렉터리에 저장되어 있다(소스 코드는 책 앞부분의 '예제 파일'을 참조하기 바란다). cleaning_code 디렉터리에는 다양한 포맷의 MTA 데이터를 형식이 잘 갖추어진 JSON으로 변경하는 파이썬Python 소스 코드가 있다. 데이터 처리는 이 책의 논점이 아니며 파이썬 코드를 실행하는 방법을 모르거나 이해하지 않더라도 충분히 따라 할 수 있다. 다만 MTA의 데이터가 실시간으로 공개되면서 포맷이 변경될 수도 있음에 따라, 책을 읽는 시점에는 파이썬 코드가 제대로 동작하지 않을 수 있으니 주의하기 바란다.

---

04 (역자주) 뉴욕교통청의 정확한 명칭은 '메트로폴리탄 교통 기관MTA(Metropolitan Transit Authority)'이다.

05 (역자주) 데이터 정규화, 포맷 변환 등의 작업을 통해 데이터 분석에 사용할 데이터를 만드는 것을 '데이터 정제'라고 한다. 학술적으로 사용하는 용어다.

NOTE_ D3는 데이터 정제Cleaning the Data를 위한 훌륭한 도구가 아니다. 데이터를 정제하는 데 자바스크립트를 사용할 수 있긴 하지만, 일반적으로는 웹 브라우저의 클라이언트[06]에서 처리하는 것이 그리 좋은 방법은 아니다. 파이썬은 XML, CSV, JSON 포맷의 데이터를 파싱하고 정형화하는 많은 툴들을 제공하므로, 이 책에서 사용한 데이터는 파이썬을 이용하여 정제하였다.

visualisations 디렉터리에는 시각화를 위한 HTML 파일들이 있다. 이 디렉터리의 소스 코드에 대해서는 다른 장에서도 다룰 것이다. 정제된 JSON 데이터는 visualisations/data에 저장되어 있다. 이 파일 중 몇 개는 용량이 아주 크니, 텍스트 에디터로 확인할 때 주의하여 열기 바란다.

CAUTION_ 시간을 투자하여 정제하고 잘 구조화한 JSON은 시각화를 구현하는 데 많은 도움을 준다. 그리고 JSON은 최소한 "http://jsonlint.com"에서 수행하는 테스트를 만족해야 한다. 웹 브라우저에서 데이터 정제와 분석을 수행하는 것은 프로그래밍 작업을 좌절시킬 뿐만 아니라 시각화를 제대로 표현하기도 어렵다.

미카의 황금률

뉴욕에서 데이터 과학자로 일하고 있는 미카 고레릭Micha Gorelick은 다음과 같은 규칙을 만들었다.

> JSON 저장 형식의 키 영역에는 데이터를 저장하지 말라[07]

D3는 이 '미카의 황금률'에 따라 데이터를 처리하기를 권장한다. 이 규칙을 따르면 시간을 많이 절약할 수 있다. 예를 들면, 아래와 같은 형식으로 JSON을 만들어서는 절대 안 된다는 것을 의미한다.

06 (역자주) 웹 브라우저에서 실행되는 모든 UI 애플리케이션을 말한다.
07 (역자주) JSON의 'key: value' 형식에서, key 영역에 데이터를 저장하지 않는다는 의미다.

```
{
   "bob": 20,
   "alice": 23,
   "drew": 30
}
```

이 예에서는 키(name)와 값(age) 영역 모두에 데이터가 저장되어 있다. 이는 완벽히 유효한 JSON 형식이지만, 미카의 황금률에는 위배된다. 미카의 황금률에 따르면 다음과 같이 변경할 수 있다.

```
[
   {
      "name": "bob",
      "age": 20
   },
   {
      "name": "alice",
      "age": 23
   },
   {
      "name": "drew",
      "age": 30
   }
]
```

여기서 데이터는 오직 값 영역에만 있으며, 키 영역에는 데이터가 무엇을 의미하는지만을 표현하였다. 만약 미카의 황금률을 위배하는 JSON이 있다면, d3.entries()를 사용해서 변경한 후 이용하길 권한다.

1.3.2 데이터 제공

d3.json()은 HTTP GET 요청을 웹 서버에 전송한다. 그러므로 요청을 받아 JSON을 반환하는 웹 서버가 실행되고 있어야 한다. 파이썬의 SimpleHTTPServer를 사용하여 웹 브라우저에 HTML과 JSON 파일을 제공하면, 웹 서버를 간단히 실행시킬 수 있다. 대부분의 리눅스와 OS X에는 파이썬이 설치되어 있지만, 윈도즈Windows는 그렇지 않으니 "http://python.org"에서 다운받아 설치한다.

서버를 실행하려면 리눅스와 OS X에서는 터미널로, 윈도즈는 명령창에서 visualisations 디렉터리로 이동 후 다음과 같은 명령어를 수행한다.

```
python -m SimpleHTTPServer 8000
```

위의 명령어는 HTTP 서버를 8000번 포트로 실행한다. 웹 브라우저를 열어 주소창에 "http://localhost:8000"을 입력하면 책에서 제공하는 HTML 예제 파일과 'data' 디렉터리에 있는 정제된 JSON 파일들을 볼 수 있다.

> **CAUTION_** HTTP 파일들을 제공하는 방법은 여러 가지가 있는데, 플랫폼에 독립적으로 이용할 때는 파이썬이 사용하기 가장 쉽다.

HTTP 서버가 실행되면, 서버는 우리가 요청한 모든 데이터를 제공한다. 모든 경로를 visualisations 디렉터리에 상대경로로 지정해서, 소스 코드를 다른 상용 서버로 이전해도 수정 없이 그대로 사용할 수 있다.

2 | 엔터 셀렉션

셀렉션selection은 CSS 선택자를 참조하여 만든 D3의 핵심 개념으로, 웹 페이지의 요소를 선택한 후 데이터 집합의 항목을 수정, 첨가, 제거하는 데 이용된다. 이번 장에서는 HTML 요소를 동적으로 추가하고 설정할 수 있는 셀렉션을 사용하여, 단순한 시각화 작업인 리스트와 막대 차트를 만들 것이다.

시각화 과정은 페이지의 body를 선택하여 컨테이너 요소를 추가한 후, 데이터 집합에 대한 각 요소의 값을 시각 요소의 속성property으로 설정하여 추가하는 기본 골격을 공통적으로 따른다. 이는 복잡한 시각화 작업에도 똑같이 반복되니, D3를 학습하는 동안 패턴을 쉽게 숙지할 수 있을 것이다.

2.1 간단한 지하철 상황판 만들기

뉴욕에서 운영하는 지하철은 공사, 스케줄 변경, 연착 등으로 정확한 상황을 알기 어려우며, 때에 따라 상황이 달라지기도 할 것이다. 특히, 평일에는 5백만 명 이상이 이용하기 때문에 러시아워에는 연착이 빈번하다.

MTA는 지하철 노선들의 상황을 매분 갱신하여 실시간 정보를 제공한다. 또한 훌륭한 에코시스템ecosystem이 형성됨에 따라 공개된 정보는 스마트 폰과 웹 기반의 애플리케이션으로 개발되어 있어서, 쉽게 이용할 수 있다. D3를 이용하여 지하철 상황을 보여주는 리스트인 첫 번째 예제도 이러한 에코시스템에 기여할 것이다. 다음은 예제의 처리 순서다.

1. "http://www.mta.info/status/serviceStatus.txt"에서 XML 형식의 데이터를 내려받는다.
2. XML에서 예제에 필요한 데이터만 추출한 후 "data/service_status.json"으로 제공될 JSON으로 변경한다.
3. 예제 1-1의 템플릿에서 d3.json("data/service_status.json", draw)과 같이

상황 데이터를 요청하도록 수정한다.

4. draw 함수를 작성한다.
5. 'python -m SimpleHTTPServer 8000' 명령어를 수행하여 서버가 파일을 제공하게 한다.
6. 웹 브라우저의 주소창에 "http://localhost:8000"을 입력한 후 상황표를 조회한다.

MTA에서 '상황 데이터 다운로드 서비스'를 아주 깔끔하게 제공하고 있으니, 첫 번째 예제에 필요한 데이터만 XML에서 추출하여 JSON으로 변경하는 것은 수월할 것이다. 여기서 지하철 이외의 데이터는 무시하고 JSON으로 변경한다. 그렇게 변경한 파일은 service_status.json으로 data 디렉터리에서 찾을 수 있다. 다음은 파일의 첫 번째 줄의 상황 데이터다.

```
{
  "status": ["GOOD SERVICE"],
  "name": ["123"],
  "url": [null],
  "text": ["..."],
  "plannedworkheadline": [null],
  "Time": [" 7:35AM"],
  "Date": ["12/15/2011"]
}
```

2.1.1 draw 함수

첫 번째 예제의 draw 함수는 뉴욕 지하철의 모든 노선별 상황을 보여주는 리스트를 만드는 간단한 작업을 수행한다. D3의 입장에서 보면, 이것은 데이터 집합의 각 요소에 대해서 노선 이름과 서비스 상황을 내용으로 하는 〈li〉요소를 생성하는 것이 된다.

draw 함수의 소스 코드는 다음과 같다. 여기서는 생략되었지만, 소스 코드는 예제 1-1의 템플릿 안에 작성해야 한다는 것을 유념하기 바란다. 소스 코드를 간단히 살펴보면 body 요소를 선택한 후 리스트를 표시하기 위해 〈ul〉 요소를, 데이터 각각에는 텍스트를 포함하고 있는 〈li〉 요소를 추가하였다.

이 일을 수행하는 아래의 D3 소스 코드가 아직 익숙하지 않을 것이다. 이해를 돕기 위해 한 줄씩 주의 깊게 살펴보겠다.

```
function draw(data) {
   "use strict";
   d3.select("body")
      .append("ul")
      .selectAll("li")
      .data(data)
      .enter()
      .append("li")
      .text(function (d) {
         return d.name + ":" + d.status;
   });
}
```

위와 같은 프로그래밍 스타일을 '메서드 체인method chain' 혹은 '캐스케이드cascade'라고 하는데, 각 메서드는 메서드를 가진 셀렉션을 반환한다. 실제로 하나의 셀렉션은 그 셀렉션 안의 모든 요소에 연산을 수행할 수 있는 특정 메서드를 가진 요소들의 배열이다.

캐스케이드는 d3.select("body")를 사용하는 것부터 시작하며, 이는 페이지의 body 요소를 선택하고 새로운 요소를 추가하기 위해 준비한다. 예제에서는 페이지 안에 〈ul〉 요소를 생성하여 넘버링되지 않는 리스트를 추가하였다. select 메서

드와 마찬가지로 append 메서드도 리스트에 대한 셀렉션을 반환하는데, 이번 경우에는 넘버링되지 않는 리스트를 반환한다.

.selectAll("li")은 약간 이상할 수도 있다. 아직 페이지에 삽입하지도 않았는데, 페이지의 모든 리스트 요소를 선택하는 selectAll을 호출하였다. 이 호출은 시각화가 보여질 새로운 리스트 요소들을 추가하기 전에 선행되어야 한다. 실제로 이것은 어떠한 항목도 가지지 않는 배열인 빈 셀렉션을 생성하지만, 앞으로 데이터를 받아들이는 데 쓸 data 메서드를 갖고 있다.

data 메서드는 데이터 집합의 각 데이터를 빈 셀렉션에 연결시킨다. 결과적으로 앞에서 반환된 셀렉션은 데이터 집합 안의 데이터 개수만큼 항목을 가진 배열이 된다(여기서는 지하철 노선 수를 위한 배열이다). 아직까지는 비어 있는 배열 셀렉션이지만, enter 메서드를 가지고 있으므로 이를 이용해 데이터를 추가할 수 있다.

enter 메서드는 새롭게 생성할 요소를 위해 데이터가 포함된 배열을 가진 셀렉션을 반환한다. 배열의 모든 요소는 데이터를 가지고 있지만 아직까지 페이지에 표현되지 않은 상태다. enter 메서드를 통해 반환되는 셀렉션을 '엔터 셀렉션enter selection'이라고 한다.

> **NOTE_** 얼핏 보면 enter() 메서드는 불필요해 보일 수도 있다. "왜 data() 메서드는 이미 데이터를 가진 배열을 반환하지 않을까"라고 의심할지도 모른다. 이렇게 메서드를 .data()와 .enter()로 구분한 이유는 기능을 명확하게 구분하기 위해서다. 즉, .data()는 단계 설정 같은 셀렉션에 대한 초기화 작업을 수행하고, .enter()는 아직 화면에는 없지만 결합시킬 데이터 포인트를 가진 요소만을 선택한다. 이런 방법은 페이지 요소의 추가와 삭제가 빈번한 동적 페이지에 아주 중요하다. 책에 있는 예제는 앞의 예제와 같이 enter() 메서드를 항상 특정한 형식으로 사용한다. D3에서 데이터를 다루는 방법에 대해 자세히 알고 싶으면 "http://bost.ocks.org/mike/join"을 참조하기 바란다.

첫 번째 예제에서 enter() 메서드는 데이터 요소가 11개 포함된 셀렉션을 반환하

지만 페이지에는 아직 어떠한 요소도 없다. 그러나 엔터 셀렉션은 페이지에 요소들을 추가할 수 있도록 준비된 상태다.

개발자 도구Developer Tool

구글 크롬의 개발자 도구와 파이어폭스의 파이어버그는 웹 개발자들이 사용하는 개발자 도구다. 만약 자바스크립트를 처음 사용한다면, 특히 시각화를 그리는 것이 처음이라면 먼저 이 도구를 사용하는 데 익숙해지도록 노력해야 한다.

개발자 도구를 이용하려면 크롬에서는 "설정 및 관리→도구→개발자 도구"(크롬 버전 23 기준)를 선택하고, 파이어폭스는 "http://getfirebug.com/"에서 파이어버그를 다운받아 설치하여 보기 메뉴에서 선택한다.

d3.select()에서 시작되는 흐름을 분석하기 위한 유용한 방법은, 개발자 도구의 콘솔에서 명령어를 실행시키고 실제로 무엇이 실행되는가를 알 수 있는 첫 번째 명령어를 살펴보는 것이다. 웹 브라우저에 시각화가 그려져 있어도, 콘솔을 열어 일련의 명령어를 각기 하나씩 실행시켜볼 수 있다. 콘솔에서 데이터에 접근하기 원한다면, d3.json("data/some_data.json", function(data){d=data})를 이용해 '글로벌 변수 d'에 데이터를 할당하여, 다음과 같이 실행한 후 출력된 내용을 살펴보면 된다.

```
d3.select("body")
   .selectAll("p")
   .data(d)
```

그리고 Command Line API를 이용하여 엔터 셀렉션 명령어를 만들어 선택하고 싶은 객체를 살펴볼 수 있다.

마지막으로 엔터 셀렉션의 각 요소를 정확히 어떻게 생성해야 하는지 D3에 알려줘야 한다. 이번 시각화 예제에서는 엔터 셀렉션에 노선 이름과 상태가 포함된 텍스트를 가진 〈li〉 요소를 추가하였다.

데이터 개개의 요소에 접근하려면 text 함수의 매개변수로 콜백 함수를 등록해야 한다. 콜백 함수는 현재 표현해야 할 데이터와 요소의 인덱스를 넘겨받는다. 예제에서 콜백 함수는 데이터의 요소 두 개(name과 status)를 단순히 연결한 텍스트를 반환하였다. 그 결과 다음과 같이 출력될 것이다.

그림 2-1 상태 리스트

- 123: PLANNED WORK
- 456: PLANNED WORK
- 7: PLANNED WORK
- ACE: PLANNED WORK
- BDFM: PLANNED WORK
- G: GOOD SERVICE
- JZ: PLANNED WORK
- L: GOOD SERVICE
- NQR: PLANNED WORK
- S: GOOD SERVICE
- SIR: GOOD SERVICE

2.1.2 데이터에 따른 스타일 추가

그림 2-1의 리스트는 기능적으로는 문제가 없지만 약간 지루해 보인다. 정보를 쉽게 공감할 수 있도록 상태가 'GOOD SERVICE'일 때는 일반 폰트로, 그 외의 상태에는 볼드 폰트로 나타나도록 꾸며보자.

```
d3.selectAll("li")
   .style("font-weight", function (d) {
   if (d.status == "GOOD SERVICE"){
      return "normal";
   } else {
      return "bold";
```

```
    }
})
```

이 소스 코드는 draw 함수의 리스트를 그려주는 소스 코드 다음에 있어야 하며, 이미 생성한 〈li〉 요소를 선택하여 위에서 정한 규칙대로 스타일을 적용한다.

데이터와 밀접하게 연결되어 있음을 주의해야 한다. 엔터 셀렉션과 엮인 개개의 데이터 포인트는 페이지상의 해당 요소와 연결되어 있다. 그래서 리스트의 모든 요소를 선택하여 데이터에 따라 스타일을 적용할 수 있는 것이다. 위의 소스 코드를 실행하면 데이터에 따라 다른 스타일이 적용되어 다음과 같이, 비교적 매력적인 리스트를 볼 수 있다.

그림 2-2 데이터에 따라 font-weight가 적용된 상태 리스트

- **123: PLANNED WORK**
- **456: PLANNED WORK**
- **7: PLANNED WORK**
- **ACE: PLANNED WORK**
- **BDFM: PLANNED WORK**
- G: GOOD SERVICE
- **JZ: PLANNED WORK**
- L: GOOD SERVICE
- **NQR: PLANNED WORK**
- S: GOOD SERVICE
- SIR: GOOD SERVICE

2.2 프라자의 일일 평균 교통량 그래프 만들기

매일 아침 수만 명의 사람들이 자동차를 타고 다리와 터널을 통해 맨해튼으로 통근하며, 그 중 상당수가 요금이 징수되는 11개의 프라자[01] 중 하나를 이용한다. MTA는 비용을 지불한 자동차들의 수치 데이터를 전자 결제와 현금으로 구분하여 매일 공개하고 있다. 이번 섹션에서는 프라자별 하루 교통량을 보여주는 막대 차트를 만들 것이다.

01 통행료를 받는 부스가 죽 늘어선 도로 요금소다.

데이터는 MTA의 웹 사이트에서 TBTA_DAILY_PLAZA_TRAFFIC.csv 파일로 제공된다. 예제에서는 일일 평균을 산출하기 위해 카운트를 정수로 변경하고, 일일 평균을 구하여 프라자별로 표시할 것이다(각 프라자는 프라자 이름으로 구분한다). 프라자 이름은 즉시 사용할 수 있는 형태가 아니지만, 다행히 'MTA 개발자 리소스' 사용자 그룹이 이러한 요구가 있을 때 사용할 수 있는 기능(plaza_traffic.json)을 만들어 두었다. plaza_traffic.json을 호출하면 JSON 결과가 반환된다. 반환된 JSON 배열 중 한 요소를 살펴보면 다음과 같다.

```
{
   "count": 26774.09756097561,
   "id": 1,
   "name": "Robert F. Kennedy Bridge Bronx Plaza"
},
```

2.2.1 div 태그를 이용하여 수평 막대 차트 만들기

앞서 만든 리스트처럼 이번에도 같은 패턴을 이용하여 차트 요소를 배치한다. 우선 시각화로 구현되는 리스트 항목 대신, div 태그들을 이용하여 일정한 너비의 막대 차트를 만들 것이다. 소스 코드의 구조는 다음과 같고 앞으로 만들 다른 소스 코드들도 이와 유사하다.

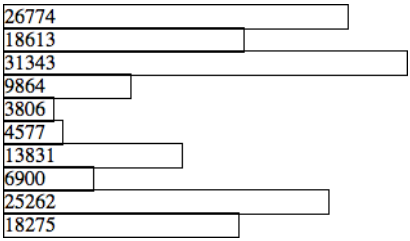
```
function draw(data) {
   d3.select("body")
      .append("div")
      .attr("class","chart")
      .selectAll(".bar")
      .data(data.cash)
      .enter()
```

```
        .append("div")
        .attr("class","bar")
        .style("width", function(d){return d.count/100 + "px"})
        .style("outline", "1px solid black")
        .text(function(d){return Math.round(d.count)});
}
```

div 태그를 삽입하고 class는 'chart'로 설정하였다. 이렇게 설정하면 추후에 선택이 용이하고 차트 전체에 스타일을 적용하기 편리하다. 클래스가 bar인 모든 div 태그를 선택하기 위해 selectAll을 호출한다(이전처럼 이것은 빈 셀렉션이다). 그 다음 빈 셀렉션을 데이터에 연결하고 엔터 셀렉션(데이터 포인트마다 하나의 요소를 가지는 배열)을 생성한다.

엔터 셀렉션에 class는 bar로 하여 div 태그를 추가하는데, div 태그의 width와 text는 count 속성의 값에 따라 결정된다. 이 count의 값은 대략 수만 정도라서, 이를 100으로 나눠 자동차 수를 픽셀로 표현할 수 있도록 변경한다(다음 장에서 범위를 표현하는 제대로 된 방법을 살펴볼 것이다).

그림 2-3 프라자별 일일 평균 교통량 차트



그 결과가 그림 2-3의 막대 차트다. 이처럼 간단하게 div 태그들을 배치하는 것만으로도 실제로 서비스할 수 있을 정도의 막대 차트를 만들 수 있다. 그렇지만 차트의 모양이 너무 엉성해서 서비스 이용자들을 기분 좋게 할 수는 없을 것 같다.

2.2.2 CSS를 사용하여 시각화 스타일 적용하기

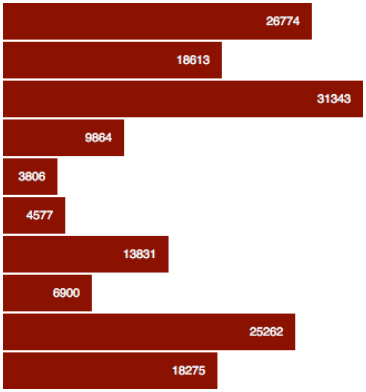
자바스크립트에서 외곽선 스타일을 제거하고 HTML의 style 태그 안에 다음 CSS를 삽입해보자.

```
div.chart{
   font-family:sans-serif;
   font-size:0.7em;
}

div.bar {
   background-color:DarkRed;
   color:white;
   height:3em;
   line-height:3em;
   padding-right:1em;
   margin-bottom:2px;
   text-align:right;
}
```

그림 2-4처럼 차트에 스타일이 아주 보기 좋게 적용되었다. 이로써 웹 페이지의 기본 요소를 표현하는 시각화 디자인에 스타일을 적용하는 것이 얼마나 단순한지 알았을 것이다. D3는 이미 잘 개발된 언어인 CSS를 이용하여 페이지 스타일을 정밀하게 제어할 수 있도록 해준다. 그래서 CSS 정의만 변경하면 플랫폼별로, 혹은 사용자별로 다른 시각화 스타일을 적용할 수 있다.

그림 2-4 약간의 CSS를 적용한 프라자별 일일 평균 교통량 차트



2.2.3 라벨 붙이기

그림 2-4의 그래프는 많은 정보를 얻기에는 미흡하다. 이 그래프로는 단지 특정 프라자가 다른 곳보다 교통량이 꽤 많다는 정도만 알 수 있을 뿐이다. 여기에 라벨을 붙여, 비교 분석을 할 수 있도록 해보자. JSON에 저장되어 있는 데이터를 이용하여 라벨을 표현하기 위해 소스 코드를 약간 수정할 것이다.

> **NOTE_** 이 예제에서는 시각화를 위해 div 태그들을 사용하였다. HTML 이외에 다른 것을 이용하지 않았으니 오래된 웹 브라우저에서도 차트를 볼 수 있을 것이다. 다만, 웹 브라우저의 레이아웃 규칙에 주의해야 하고 자바스크립트 코드가 필요 이상으로 복잡해진다. 막대들의 크기를 제어하기 위해 CSS를 이용하면 사용자 스타일 시트에 따라 시각화 형태가 변경되는 등의 문제가 발생할 수도 있다. 3장에서 알아볼 'SVG'를 이용한 시각화는 이러한 문제에서 자유롭다.

각 막대의 왼쪽에 프라자 이름을 표시할 것이다. 프라자 이름은 JSON 안에 name으로 저장되어 있으니, 이 값을 div의 text 속성에 설정하면 간단히 프라자의 이름을 표시할 수 있다. 단, 프라자 이름은 막대의 왼쪽에 있어야 하므로 막대를 만드는 div 태그보다 이전에 그려야 한다. 그리고 라벨과 막대는 공통된 데이터 요소를

공유해야 하므로 데이터 요소마다 div 태그를 하나씩 생성하여 라벨과 막대를 추가하는 데 사용한다.

먼저 데이터 포인트마다 div 태그를 만드는 것부터 시작하도록 소스 코드를 수정한다.

```
d3.select("body")
   .append("div")
   .attr("class", "chart")
   .selectAll("div.line")
   .data(data.cash)
   .enter()
   .append("div")
   .attr("class","line");
```

이제 소스 코드가 조금 익숙해졌을 것이다. 먼저 페이지의 body를 선택하고 차트 div를 추가한다. 그리고 아직은 추가하지 않았지만, class가 line인 div 태그( )를 선택하여 데이터와 함께 빈 셀렉션을 연결한다. 그 다음, 셀렉션에 진입하고 데이터의 각 요소에 대해 class가 line으로 지정된 div 태그를 추가한다. 수행된 결과를 HTML로 표현하면 다음과 같다.

```
<div class="chart">
   <div class="line"/>
   <div class="line"/>
   <div class="line"/>
   ...
   <div class="line"/>
</div>
```

이렇게 단순해진 구조에서 각 라인에 라벨을 추가한다.

```
d3.selectAll("div.line")
   .append("div")
   .attr("class","label")
   .text(function(d){return d.name});
```

다음으로 각 라인에 막대를 추가한다.

```
d3.selectAll("div.line")
   .append("div")
   .attr("class","bar")
   .style("width", function(d){return d.count/100 + "px"})
   .text(function(d){return Math.round(d.count)});
```

이전 예제에서 언급했듯 d3.selectAll("div.line")에서 반환된 셀렉션은 데이터와 아주 밀접하게 연관되어 있다. 그래서 매번 데이터를 참조할 수 있는 셀렉션을 만들 수 있다. 데이터는 셀렉션의 모든 자식에 순차적으로 전달되므로 하나의 div.line에 있는 div.label과 div.bar는 같은 데이터에서 각 항목을 가져와 표시할 수 있다.

마지막으로 스타일을 적용하여 어지럽게 배치되어 있는 라벨과 막대를 정리하자. 라벨이 구분되도록 다음과 같이 div.bar의 왼쪽으로 여백을 보기 좋게 넣어준다.

```
div.label {
   height:3em;
   line-height:3em;
   padding-right:1em;
```

```
    margin-bottom:2px;
    float:left;
    width:20em;
    text-align:right;
}
```

float:left는 막대가 라벨의 오른쪽에 위치하도록 한다. 마지막으로 라벨의 너비를 정해 놓으면 모든 막대가 보기 좋게 일렬로 나열된다. 최종적인 시각화 결과는 그림 2-5와 같다.

> **NOTE_** 정확한 데이터를 표시하기 위해서 MTA에서 직접 데이터를 다운받았다면, 그림 2-5와는 조금 다르게 나타날 것이다. 데이터가 교통량에 따라 동적으로 생성되어 변경되었기 때문에 그런 것이니, 염려하지 않아도 된다.

그림 2-5 라벨이 포함된 프라자별 일일 평균 교통량 차트