Содержание

Об авторах

Эндрю Ринингсланд – разработчик и журналист, в последние десять лет он большую часть времени занимался созданием интерактивного контента для таких изданий, как Financial Times, Times, Sunday Times, Economist и Guardian. За три года работы в газетах Times и Sunday Times он участвовал в различных проектах – от написания некрологов о смерти таких личностей, как Нельсон Мандела, до резонансных расследований типа допингового скандала – крупнейшей в истории утечки данных об анализах крови спортсменов. В настоящее время является старшим разработчиком группы интерактивной графики в редакции Financial Times.

Выражаю благодарность своей восхитительной подруге Наоми, которая бог знает сколько раз подбадривала и нахваливала меня, пока я работал над книгой. Спасибо также всем членам группы D3.js Slack, оказывавшим мне всемерную поддержку, а особенно завсегдатаям канала #v4-migration, которые помогли освоиться с бесчисленными изменениями в версии 4.

Свизек Теллер, автор книги «Data Visualization with d3.js» – технарь по призванию. В 21 год он основал свой первый стартап, а теперь, в поисках очередной достойной идеи, трудится в роли специалиста по всем уровням веб-разработки, отдавая предпочтение внештатному сотрудничеству со стартапами на ранних этапах становления. Во время, свободное от кодирования, он ведет блог, пишет книги или выступает на различных мероприятиях в Словении и соседних странах, мечтая сделать доклад на какой-нибудь крупной международной конференции. В ноябре 2012 года он начал писать книгу «Почему программисты работают по ночам» и задался целью улучшить жизнь разработчиков, где бы они ни жили.

Я благодарен @gandalfar и @robertbasic, которые подначивали меня по ходу работы и выступали в роли подопытных свинок для проверки моих примеров. Также выражаю искреннюю признательность всем членам группы @psywerx, которые не дали мне спятить и создали один из лучших в мире наборов данных.

О рецензенте

Герардо Фуртадо – преподаватель биологии, популяризатор науки, автор книги по эволюционной биологии, опубликованной Бразильским университетом, а также нескольких художественных произведений.

Его второе академическое пристрастие – визуализация данных, именно оно в конечном итоге и привело его к D3.js, самой мощной (и комплексной) JavaScript-библиотеке для этой цели.

Предисловие

Здравствуйте, читатели! В этой книге вы познакомитесь с основами одной из самых распространенных и мощных библиотек визуализации данных, но не только. К концу нашего совместного путешествия вы приобретете все навыки, необходимые настоящему специалисту по D3, и сможете решить любую задачу: от создания визуализации с нуля до запуска ее на сервере и написания автоматизированных тестов. Тех, чьи знания JavaScript несколько подзамшели, ждет приятный сюрприз – эта книга располагает к использованию самых последних добавленных в язык средств, и мы всегда объясняем, почему это круто и чем отличается от «старой школы».

Краткое содержание книги

В главе 1 «Первое знакомство с D3, ES2017 и Node.js» рассматриваются новейшие средства для создания визуализации данных с помощью D3.

Глава 2 «Начала DOM, SVG и CSS» содержит обзор базовых веб-технологий, используемых в D3.

Глава 3 «Геометрические примитивы в D3» посвящена определению и созданию базовых геометрических элементов, из которых составляются визуализации данных.

В главе 4 «Извлечение пользы из данных» вы научитесь преобразовывать данные, так чтобы D3 могла их визуализировать.

Глава 5 «Все для удобства пользователя» покажет вам, как дополнить визуализацию средствами анимации и интерактивности.

Глава 6 «Иерархические макеты в D3» посвящена иерархической компоновке, которая поднимет ваши навыки владения D3 на новый уровень, где для создания сложных диаграмм применяются повторно используемые шаблоны.

В главе 7 «Прочие макеты» обсуждаются неиерархические макеты, позволяющие ускорить создание диаграмм других типов.

В главе 8 «Использование D3 на сервере с применением Canvas, Koa 2 и Node.js» описаны создание и развертывание веб-службы на базе сервера Node.js, которая отрисовывает визуализации D3 с помощью библиотек Koa.js и Canvas.

В главе 9 «Обретение уверенности в своих визуализациях» показано, как улучшить качество кода за счет проверки соблюдения стандартов кодирования, статической проверки типов и автоматизированного тестирования проектов.

В главе 10 «Проектирование хорошей визуализации данных» сопоставляются различные подходы к визуализации и предлагается набор рекомендаций.

Что необходимо для чтения книги

Вам понадобится компьютер, на котором можно запустить Node.js. В первой же главе мы обсудим, как его установить, а работать он может практически на любой машине, хотя для разработки несколько лишних гигабайтов памяти не помешает. Для нескольких примеров построения карт нужны солидные вычислительные ресурсы, но большинство компьютеров, выпущенных после 2014 года, с такой нагрузкой справится.

Понадобится также последняя версия браузера; я предпочитаю Chrome и разрабатывал примеры в нем, однако Firefox тоже годится. Можете также попробовать Safari, Internet Explorer/Edge, Opera или любой другой браузер, но, на мой вкус, инструменты разработчика в Chrome самые лучшие.

На кого рассчитана эта книга

На разработчиков веб-приложений, авторов интерактивных новостей, специалистов по анализу и обработке данных и всех, интересующихся интерактивным представлением данных в вебе с помощью библиотеки D3. Предполагается знакомство с основами JavaScript, но предварительный опыт визуализации данных или работы с D3 не потребуется.

Графические выделения

В этой книге тип информации обозначается шрифтом. Ниже приведено несколько примеров с пояснениями.

Фрагменты кода внутри абзаца, имена таблиц базы данных, папок и файлов, URL-адреса, данные, которые вводит пользователь, и адреса в Твиттере выделяются следующим образом: «Если в сообщении говорится что-то типа ‘`Command not found`’, проверьте, все ли правильно установлено, и убедитесь, что Node.js включен в переменную среды `$PATH`».

Кусок кода выглядит так:

```
"babel": {
  "presets": [
    "es2017"
  ]
},
"main": "lib/main.js",
"scripts": {
  "start": "webpack-dev-server --inline",
},
```

Входная и выходная информация командных утилит выглядит так:

```
$ brew install n
$ n lts
```

Новые термины и **важные фрагменты** выделяются полужирным шрифтом. Например, элементы графического интерфейса в меню или диалоговых окнах выглядят в книге так: «Мы в основном будем пользоваться вкладками **Elements** и **Console**: вкладка **Elements** предназначена для просмотра DOM, а **Console** – для ввода JavaScript-кода и анализа ошибок».

Предупреждения и важные примечания выглядят так.

Советы и рекомендации выглядят так.

Отзывы

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или может быть не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Скачивание исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com или www.дмк.рф на странице с описанием соответствующей книги.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг — возможно, ошибку в тексте или в коде — мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в Интернете по-прежнему остается насущной проблемой. Издательство ДМК Пресс и Packt очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в Интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или веб-сайта, чтобы мы могли применить санкции.

Пожалуйста, свяжитесь с нами по адресу электронной почты dmkpress@gmail.com со ссылкой на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, и помогающую нам предоставлять вам качественные материалы.

Глава 1

Первое знакомство с D3, ES2017 и Node.js

Библиотека **Data-Driven Documents (D3)**, разработанная Майком Бостоком (Mike Bostock) и сообществом D3 в 2011 году, пришла на смену прежней библиотеке Бостока Protovis. Она поддерживает отрисовку данных с точностью до пикселя благодаря абстрагированию вычисления масштабов и осей с помощью простого предметно-ориентированного языка (DSL) и благодаря использованию идиом, понятных всякому, кто работал с популярной JavaScript-библиотекой jQuery. В D3, как и в jQuery, нужно сначала выбрать множество элементов, а затем применить к ним операции с помощью цепочки функций-модификаторов. В контексте визуализации данных такой декларативный подход оказывается гораздо проще многих других имеющихся инструментов. На официальном сайте по адресу https://d3js.org/ приведено много примеров, демонстрирующих мощь D3, но поначалу разобраться в них нелегко. Прочитав эту книгу, вы будете знать D3 в достаточной степени, чтобы понять примеры и приспособить их к своим нуждам. Если вы хотите познакомиться с разработкой D3 поближе, скачайте исходный код с сайта GitHub по адресу https://github.com/d3.

В этой главе мы заложим фундамент для выполнения представленных в книге примеров. Я объясню, как писать код на языке ECMAScript 2017 (ES2017) – последней и самой передовой версии JavaScript – и как с помощью программы Babel транслировать его на язык ES5, который понимает любой современный браузер. Затем мы познакомимся с основами D3 v4 на примере создания простой диаграммы.

Что такое D3.js?

Благодаря точному контролю и элегантности D3 заслуженно считается одной из самых мощных библиотек визуализации с открытым исходным кодом. Но это также означает, что для простых задач – изображения одного-двух линейных графиков – она не слишком подходит; в этом случае лучше взять какую-нибудь библиотеку для рисования графиков. Кстати, многие из них сами основаны на D3. Обширный список опубликован по адресу https://github.com/sorrycc/awesome-javascript#data-visualization.

D3 построена на принципах функционального программирования, которое ныне переживает второе рождение в сообществе JavaScript. Эта книга не о функциональном программировании, но многое покажется знакомым тем, кто уже сталкивался с этими принципами. Если вы не из их числа или привыкли к объектно-ориентированному программированию, не расстраивайтесь, я буду объяснять все необходимое по ходу дела и надеюсь, что раздел о функциональном программировании в начале главы 4 поможет вам понять, почему эта парадигма так полезна, особенно в задачах визуализации данных и конструирования приложений.

Что случилось с классами?

Во втором издании этой книги было много примеров использования механизма классов, появившихся в языке ES2015. Но теперь мы используем *фабричные функции*, а ключевое слово `class` ни разу не встречается. С чего бы это?

В ES2015 классы на самом деле были *синтаксическим сахаром* поверх фабричных функций, т. е. в конечном счете компилировались в фабричные функции. Хотя с помощью классов можно повысить уровень организации сложного кода, в итоге они просто скрывают происходящее внутри. К тому же использование объектно-ориентированных парадигм, в т. ч. классов, идет вразрез с одним из самых действенных и элегантных аспектов языка JavaScript – полноправными функциями и объектами. Ваш код станет проще и изящнее, если вы будете придерживаться функциональных парадигм, да и примеры, созданные сообществом D3, читать будет проще, поскольку в них классы почти никогда не используются.

Против использования классов можно привести еще много аргументов, для изложения которых здесь просто нет места. Рекомендую обратиться к великолепной серии статей Эрика Эллиота «The Two

Pillars of JavaScript» по адресу www.medium.com/javascript-scene/the-two-pillars-of-javascript-ee6f3281e7f3.

Что нового в версии D3 v4?

Одно из главных событий, случившихся после выхода предыдущего издания этой книги, – выпуск версии 4.

Из многочисленных изменений самым важным является полный пересмотр пространства имен D3. Это означает, что ни один пример, приведенный в этой книге, не будет работать в версии D3 3.x, а примеры из второго издания книги «Learning D3.js Data Visualization» не будут работать в D3 4.x. Это, пожалуй, самое страшное, что жестокий м-р Босток мог учинить над такими авторами учебников, как я (шучу, шучу). Но шутки в сторону – из-за этого многие примеры кода, разработанные сообществом D3, потеряли актуальность и могут даже показаться странными тому, для кого эта книга стала первым знакомством с библиотекой. Поэтому так важно проверять, для какой версии D3 написан пример, – если для 3.x, то стоит поискать аналогичный пример для 4.x, чтобы не пасть жертвой когнитивного диссонанса.

Обычно в примерах версия D3 указывается в тегах `script` в начале кода. Если вы видите такой код:

```
<script src="https://d3js.org/d3.v3.min.js"></script>
```

то пример написан для версии 3.x. А если такой:

```
<script src="https://d3js.org/d3.v4.min.js"></script>
```

то это более современный пример, рассчитанный на версию 4, т. е. вы движетесь в правильном направлении.

С этим связано также разбиение D3 на много библиотек меньшего размера (микробиблиотек). Вы можете пойти по одному из двух путей:

- работать с D3 как с единой библиотекой (монобиблиотекой) так же, как в версии 3;
- включать в проект лишь отдельные компоненты D3 (микробиблиотеки).

В этой книге принят первый подход. Использование микробиблиотек заметно усложнило бы изучение D3, хотя при этом уменьшается размер конечного комплекта файлов, который должны будут загрузить пользователи для просмотра вашей графики. Тем не менее я буду отмечать, в каком пакете находится та или иная функциональность,

а вы, когда получше освоитесь с D3, сможете перейти на микробиблиотеки, вместо того чтобы включать все подряд, нужное и ненужное.

Что такое ES2017?

Одно из главных изменений в этой книге с момента ее первого издания – упор на современный JavaScript, в данном случае ES2017. Эта версия, ранее называвшаяся ES6 (Harmony), – крупный шаг в развитии языковых средств JavaScript, позволивший реализовать новые паттерны, которые упрощают восприятие кода и повышают его выразительность. Если вы писали на JavaScript раньше и примеры в этой главе кажутся вам странными, значит, вы имели дело с прежним, более распространенным синтаксисом ES5.

Но не переживайте! На привыкание к новому синтаксису не уйдет много времени, а я уж постараюсь объяснять новые языковые возможности по мере надобности. Хотя поначалу кривая обучения может показаться довольно крутой, в конце вы будете писать код куда лучше и окажетесь на переднем крае современной разработки на JavaScript.

Отличное введение во все новшества, которые принес нам ES2015-17, можно найти в руководстве, составленном авторами программы Babel.js (https://babeljs.io/docs/learn-es2015/), которой мы будем постоянно пользоваться на страницах этой книги.

Прежде чем двигаться дальше, я хотел бы развеять некоторые мифы о том, чем на самом деле является ES2017. Первоначально версии стандартов ECMAScript (сокращенно ES) нумеровались последовательно: ES4, ES5, ES6, ES7. Но начиная с ES6 этот порядок был изменен, поскольку важно было отразить тот факт, что новые стандарты выходят ежегодно, чтобы не отставать от современных тенденций разработки. Так что текущий стандарт относится к 2017 году. Важной вехой стала версия ES2015, которая примерно соответствует ES6. Стандарт ES2016, ратифицированный в июне 2016 года, основывается на предыдущем стандарте с добавлением нескольких исправлений и двух новых возможностей. ES2017 пока находится на стадии проекта, т. е. новые предложения рассматриваются и разрабатываются – до ратификации, которая произойдет в течение 2017 года. Эта книга писалась, когда новые средства еще не были утверждены, и, возможно, они даже не войдут в стандарт ES2017, так что с их официальным включением, быть может, придется подождать до выхода следующей версии стандарта.

Но это не причина для беспокойства, поскольку мы все равно используем Babel.js для компиляции всего кода на ES5, чтобы он одинаково работал в Node.js и в браузере. Для полноты картины я буду указывать, в какой версии стандарта появилось то или иное средство (например, модули были включены в ES2015), но, говоря о JavaScript, я всегда имею в виду современный JavaScript безотносительно к номеру спецификации ECMAScript.

Запускаем Node и Git из командной строки

В этой книге я постараюсь не выказывать предпочтений какому-либо конкретному редактору или операционной системе (хотя сам использую Atom в macOS X), но все-таки какие-то предварительные условия должны быть соблюдены.

Первое из них – Node.js. В настоящее время Node широко применяется для веб-разработки и, по сути дела, представляет собой средство для выполнения JavaScript-кода из командной строки. Ниже я покажу, как написать серверное приложение для Node, а пока займемся просто установкой Node и `npm` (замечательного менеджера пакетов для Node).

Если вы работаете в Windows или macOS X без Homebrew, то воспользуйтесь установщиком по адресу https://nodejs.org/en/. Если в вашей системе на базе macOS X имеется Homebrew, то я рекомендую вместо этого установить пакет `n`, который позволяет без труда переключаться с одной версии Node на другую:

```
$ brew install n
$ n lts
```

Пользователям Windows знак `$` может показаться непонятным. В операционных системах на базе UNIX обычный пользователь видит в командной строке приглашение `$`, а суперпользователь *root* – приглашение `#`. Включая знак `$`, я показываю, что вы должны выполнять команды от имени обычного пользователя, а не суперпользователя.

В любом случае по завершении проверьте результат, выполнив такие команды:

```
$ node --version
$ npm --version
```

Если будут напечатаны версии `node` и `npm`, значит, все хорошо.

Я работаю соответственно с версиями 6.5.0 и 3.10.3. Ваша версия может отличаться, но важно, чтобы версия Node была не ниже 6.0.0.

Если в сообщении говорится что-то типа `Command not found`, проверьте, все ли правильно установлено, и убедитесь, что Node.js включен в переменную среды `$PATH`.

В этой книге мы с помощью Babel и Webpack будем преобразовывать наш изысканный модульный современный код на JavaScript в нечто, что могут выполнить даже самые захудалые, побитые молью браузеры (ау, Internet Explorer 9!).

Для начала создадим файл `package.json`, в котором будут храниться версии всех нужных нам зависимостей. Для этого создадим новую папку и выполним команду `npm init`:

```
$ mkdir d3-projects
$ cd d3-projects
$ npm init -y
```

Флаг `-y` означает, что `npm init` должна использовать параметры по умолчанию, не задавая никаких вопросов.

Затем с помощью `npm` установим все необходимое:

```
$ npm install "babel-core@^6" "babel-loader@^6" "babel-preset-es2017@^6"
"babel-preset-stage-0@^6" "webpack@^2" "webpack-dev-server@^2" css-loader
style-loader json-loader --save-dev
```

Эта команда установит версию 2 Webpack, версию 6 Babel и комплект начальных установок и дополнительных модулей для того и другого. Имена и номера версий программ будут записаны в файл `package.json`, так что для повторной установки понадобится всего лишь ввести команду

```
$ npm install
```

Еще установим D3:

```
$ npm install d3 --save
```

Далее нужно создать конфигурационный файл для Webpack. Не буду объяснять, что означает каждая директива, все комментарии вы сможете найти в репозитории кода к этой книге. Просто сохраните показанный ниже код в файле `webpack.config.js`:

```
const path = require('path');
module.exports = [{
```

```
  entry: {
    app: ['./lib/main.js'],
  },
  output: {
    path: path.resolve(__dirname, 'build'),
    publicPath: '/assets/',
    filename: 'bundle.js',
  },
  devtool: 'inline-source-map',
  module: {
    rules: [{
      test: /.js?$/,
      exclude: /(node_modules|bower_components)/,
      loader: 'babel-loader',
    }, {
      test: /.json$/,
      loader: 'json-loader',
    }, {
      test: /.css$/,
      loader: 'style-loader!css-loader',
    }],
  },
}];
```

Напоследок отредактируем файл `package.json`, добавив несколько аббревиатур, которые упростят нам жизнь. После строки, начинающейся словом `name`, вставьте такие строки:

```
"babel": {
  "presets": [
    "es2017"
  ]
},
"main": "lib/main.js",
"scripts": {
  "start": "webpack-dev-server --inline",
},
```

Все это необходимо, если вы начинаете новый проект с нуля.

Можно вместо этого клонировать репозиторий книги из GitHub. GitHub – это место, где размещается большая часть всего открытого и иного кода в мире. Я не только положил туда примеры и тестовые данные, но и поработал над конфигурацией. Далее я буду исходить из предположения, что вы клонировали репозиторий, создав копию на своей машине. Для этого выполните такие команды:

```
$ git clone https://github.com/aendrew/learning-d3-v4
$ cd learning-d3-v4
```

В результате будут клонированы среда разработки и все примеры в каталоге `learning-d3-v4/`, после чего мы сделаем этот каталог текущим и установим все зависимости с помощью `npm`.

Есть еще один вариант: разветвить репозиторий на GitHub и клонировать свое ответвление вместо моего. Это позволит публиковать результаты своего труда в облаке, упростит получение помощи от коллег, даст возможность отображать завершенные проекты на страницах GitHub и даже отправлять исправления и дополнения в родительский проект. Что, в свою очередь, будет способствовать улучшению будущих изданий этой книги. Чтобы разветвить проект `aendrew/learning-d3-v4`, нажмите кнопку «fork» на сайте GitHub и замените строку `aendrew` в приведенном выше фрагменте кода своим именем пользователя GitHub.

Каждая глава книги хранится в отдельной ветке. Для переключения между главами выполните команду вида

```
$ git checkout chapter1
```

Вместо `1` укажите номер интересующей вас главы. Но пока останемся в главной ветке. Чтобы вернуться к ней, введите такую команду:

```
$ git stash save && git checkout master
```

В главной ветке вы будете писать свой код по мере работы над книгой. Установить зависимости все равно нужно, сделаем это прямо сейчас:

```
$ npm install
```

Весь исходный код, с которым вы будете работать, находится в папке `lib/`. Обратите внимание, что там имеется только файл `main.js`; почти всегда мы будем иметь дело именно с ним, поскольку `index.html` – всего лишь минимальный контейнер, в котором отображается результат нашей работы. Ниже он приведен целиком, и это первый и последний раз, когда в книге встретится HTML-код:

```
<!DOCTYPE html>
<html>
  <head>
  <meta charset="utf-8">
  <title>Learning Data Visualization with D3.js</title>
  </head>
  <body>
  <script src="assets/bundle.js"></script>
  </body>
</html>
```