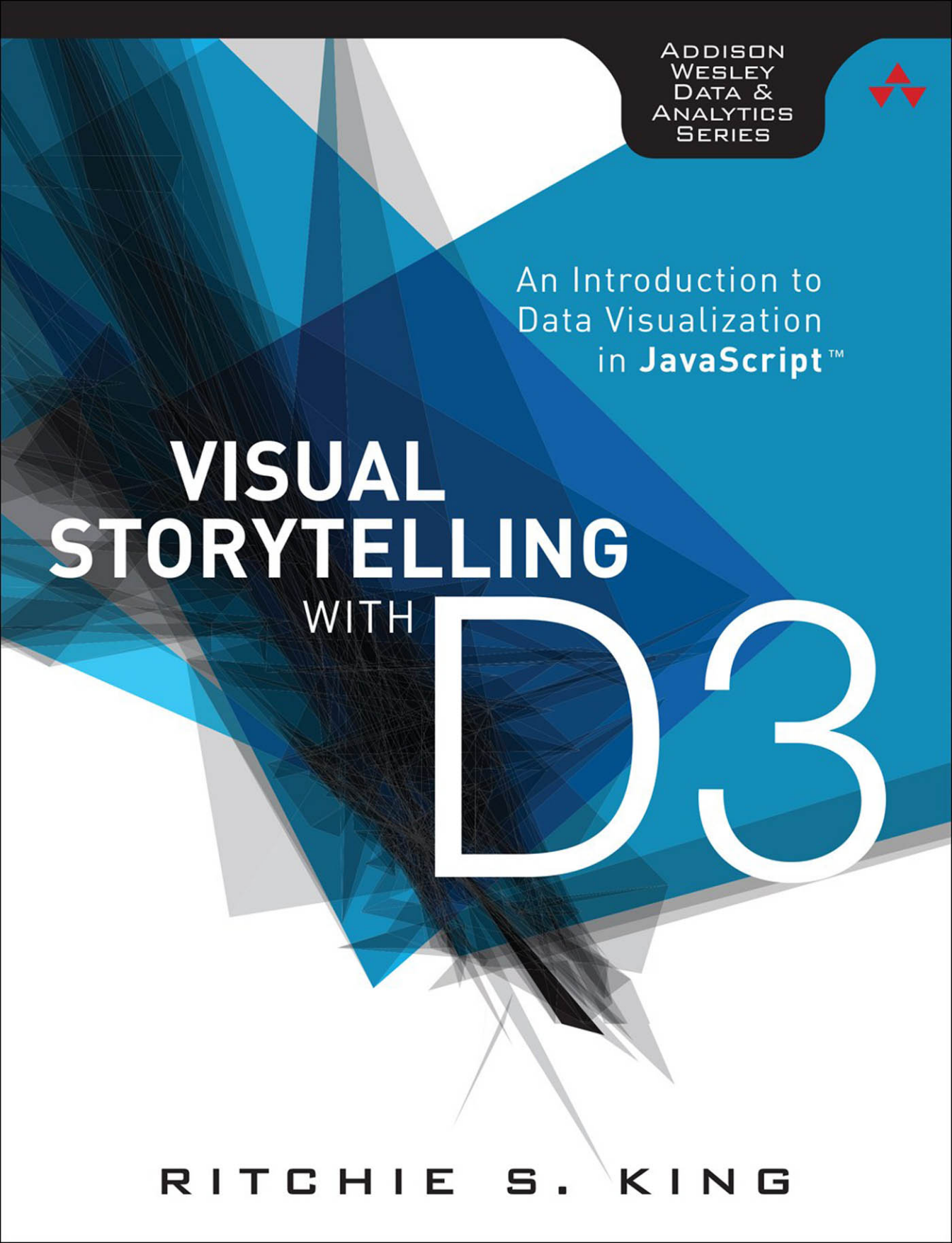
Addison Wesley Data & Analytics Series
An Introduction to Data Visualization in JavaScript™
VISUAL STORYTELLING WITH D3
RITCHIE S. KING

Visual Storytelling with D3

Listing 4.1 shows a snippet of code you can use to get started building a web page using D3.

Listing 4.1 *A starter snippet*

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
       /* Your CSS goes here */
</style>
</head>
<body>
<script src="http://d3js.org/d3.v3.min.js"></script>
       <!-- Your HTML goes here -->
<script>
       // Your JavaScript goes here
</script>
</body>
</html>
```

All of the "Your X goes here" statements above are comments, so your machine won't try to interpret them as code. (The reason the symbols surrounding each comment are different is that each one is written in a different language.)

Note that while you reference the D3 library before your HTML, the JavaScript you write goes below your HTML. Why? Because you use D3 to manipulate elements on the web page, and you need to create those elements first!

Making Selections

So what is a selection, exactly? Well, it's a selection of elements from your web page's DOM, or Document Object Model. The DOM is the hierarchical, HTML-based structure that underpins every web page. When your browser is rendering a web page, it is interpreting the underlying DOM.

When you take a peek at the DOM (which you can do with the Web Inspector), it looks a lot like an HTML file, and it sometimes mirrors the HTML file being loaded exactly. But there's a big difference between a static HTML file and the DOM. The DOM always represents the current state of a web page. It's initial state could be defined by a static HTML file, but then change based on, say, some JavaScript.

Back to selections. What kinds of elements can a selection be? It could be a single `<div>`, it could be all of the page's paragraph elements, it could be everything with the class `myClass`, it could be an SVG circle, or it could even be the entire body of the page.

The best way to understand what selections are and how they work is to use them. So, put the following code into a file called **selections.html** and open it up in your browser (see Figure 4.1). Then we can create some selections in the console!

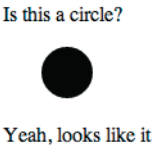

Figure 4.1 Looks like a circle to me, too

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
      /* Your CSS goes here */
</style>
</head>
<body>
<script src="http://d3js.org/d3.v3.min.js"></script>
      <div id="header">
            <p>Is this a circle?</p>
      </div>
      <svg width="500px" height="40px">
            <circle r="20" cx="50" cy="20"/>
      </svg>
      <div class="footer">
            <p>Yeah, looks like it</p>
      </div>
<script>
      // Your JavaScript goes here
</script>
</body>
</html>
```

D3 has two methods for creating selections: `d3.select()` and `d3.selectAll()`. Let's start with `d3.select()`, which only selects one element at a time. Open up the console, which you can do in Chrome by going to View > Developer > JavaScript Console. Type `d3.select("p")` into the console. Figure 4.2 shows what you should see.

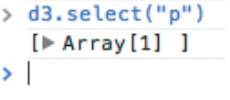

Figure 4.2 A D3 selection

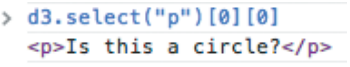

Figure 4.3 The element inside

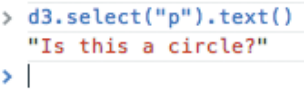

Figure 4.4 Paragraph text on display

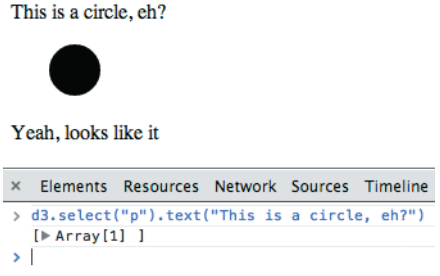

Figure 4.5 Translated to Canadian English

Doesn't look like much—but that's it! That's our selection, an array with another array inside of it. Selections always look like this, but don't worry about why for now. Let's just take a peek inside. We can peel back the layers by indexing into the array: Enter `d3.select("p")[0][0]` into the console. The result is shown in Figure 4.3.

Yep—that looks familiar. It's the markup for the first paragraph at the top of **selections.html**. You may have figured out what `d3.select()` did in this case: Since the method was passed the argument `"p"`, it went hunting for paragraph elements. And the way `d3.select()` works is that it always creates a selection out of the first match it finds.

So now that we have a selection, what can we do with it? So many things! Creating and then changing selections is what D3 is all about. D3 has scores and scores of methods that are built to operate on selections and modify the element or elements inside. One such method is `text()`. Type `d3.select("p").text()` into the console, and you should get what is shown in Figure 4.4.

As you can see, `text()` returns whatever text is stored inside of your selection. (Or, to be more precise, it returns the text inside of the opening and closing tags of the element stored inside of your selection.) Pretty cool. Now for some magic: You can change that text by passing an argument to the `text()` method. Try

```
d3.select("p").text("This is a circle, eh?").
```

Figure 4.5 shows what you'll get.

Congratulations! You have officially used D3 to drive a web document. Two Ds down, one to go!

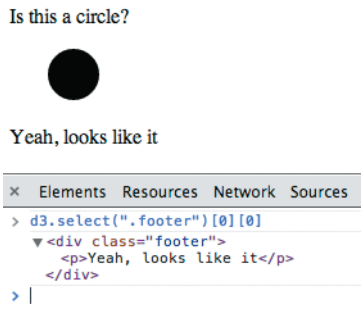

Figure 4.6 Selecting by class

Using CSS Selectors to Create Selections

Let's try selecting some of the other elements on the page. Go ahead and refresh **selections.html** so that it is no longer in "Canadian English." Say we wanted to select the second `<div>` element. Since `d3.select("div")` will always pick the first `<div>` it comes across, we have to use something else to hone in on the second. In this case, the second `<div>` is the only element on the page with the class `"footer"`. We can take advantage of this by typing `d3.select(".footer")`. The result is shown in Figure 4.6.

You should recognize the convention D3 uses to select by class: `.className`. Prefacing a class name with a period—that's straight from CSS. In fact, D3 always employs CSS selectors to make selections. That term—**selector**—may be new to you, but what it refers to should be familiar. The selector for paragraph elements, for example, is `p`. The selector for elements with the id "myID" is `#myID`. Collectively, selectors are CSS's nomenclature for referring to elements. I'm sure you've used them countless times.

SVG elements also have CSS selectors, which is why you can use CSS to style them. Those selectors, just like the selectors for HTML elements, are the names of SVG tags. The selector for a rectangle, which has the tag `<rect>`, is `rect`. The selector for a group, which has the tag `<g>`, is `g`. If you're selecting an SVG element by class, then of course, the convention is the same: `.className`.

Try typing `d3.select("circle")` into the console. Of course, `d3.select("circle").text()` doesn't do much, but we'll learn some methods in the next section you can use to manipulate SVG elements.

Creating Selections from Other Selections

It's also possible to create a selection from another selection. Say we wanted to select the second paragraph element in **selections.html**. If we use `d3.select("p")` it will always return the first paragraph. And the second paragraph has no class or id, so we can't use one of those to hone in on it. What we can do instead is select the second `<div>` using its class name and then select the first paragraph within that `<div>`:

```
d3.select(".footer").select("p")
```

In the code above, D3 is only hunting for paragraph elements within the `<div>` that has the class `footer`.

Assigning Selections to Variables

One final note about selections: Since they are arrays, you can assign them to variables. Try typing the following into the console:

```
var myGraf = d3.select("p")
myGraf.text()
```

Changing a Selection's Attributes

Being able to change the text inside of an HTML element is pretty cool. But what if you wanted to change the font size or reformat the element in some other way? Or, what if you wanted to adjust an SVG shape, which has no text?

D3 has a really useful method that enables you to change the attributes of a selection—it's called `attr()`. (I find that saying function and method names out loud or in my head actually makes me feel more comfortable using them. But abbreviating the word "attribute" after the "r" produces a fragment that's difficult to say (unless you're French). So, many people pronounce `attr` such that it rhymes with "hatter.")

The SVG circle in **selections.html** hasn't gotten a lot of play, so let's use `attr()` on it. First, since we'll be referencing the circle quite a bit, let's create a variable for the selection just to make things easy. Type the following into the console:

```
var myCircle = d3.select("circle");
```

Now type this:

```
myCircle.attr("fill", "red");
```

You will get what is shown in Figure 4.7.

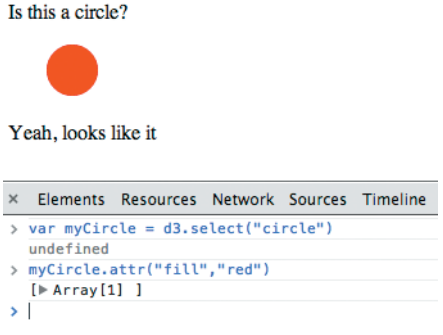

Figure 4.7 It is a circle, and now it's red

It should be pretty obvious what happened here: `attr()` changed the fill attribute of the circle to red. Terrific! Now, say you wanted to change the radius, too. To do that, type:

```
myCircle.attr("r",5)
```

Figure 4.8 shows the change in the radius.

You can also use `attr()` to add a stroke, even though no stroke is defined in the markup.

```
myCircle.attr("stroke", "black")
```

Figure 4.9 shows the circle with a stroke around it.

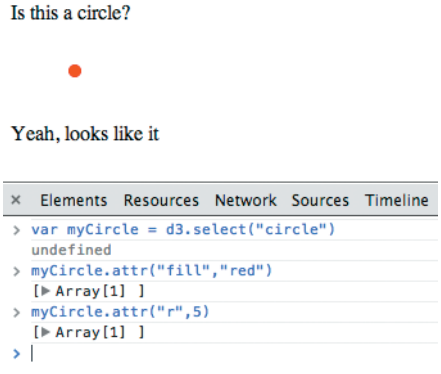

Figure 4.8 And now it looks like a dot

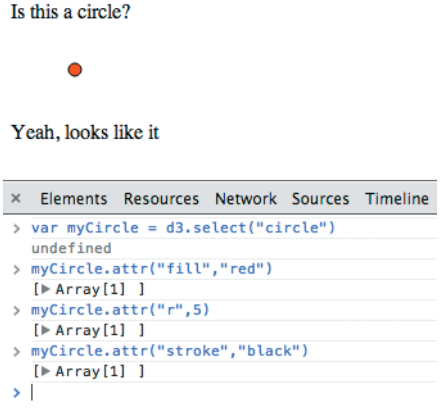

Figure 4.9 Stroke of genius