B3book Companion

Hongtao Hao

2020-12-01

Contents

Why did I build this website?

D3 is hard, literally. Its learning curve is not fun. The key to mastering it, I believe, is to **practice and keep practicing**. I found Scott Murray's D3 book easy to understand, and very useful. To make the best of the book, however, it's very crucial to practice on your own, rather than just reading the book. Reading others' codes won't make you a good programmer.

Therefore, I documented the codes' objectives in the book, i.e., what the codes in the book aimed to achieve. Then I used these prompts to train myself. I found them useful so I decided to share these prompts through this website.

If you can achieve all of what these prompts asked you to do, then you don't need to read the whole book again.

Chapter 1

Chapter 3 & 4

Some Key Points from Chapter 3 & 4

- What are HTML elements: `html`, `head`, hyperlinks, `title`, `body`, `h1`, `h2`, `h3`, `p`, `div`, `span`, `img`, `ul`, `ol`, `li`, `em`, `strong`, `a`
- `<a href="Link here">Content here</a>`, where `href` stands for **hypertext reference**;
- Difference between `id` and `class`: each element can only have one `id` and each `id` can be used only once. In contrast, elements can have multiple `class`(separated by space), and each `class` can be used for multiple times.
- Comments: "`<!-- Comments here -->`"
- **DOM** stands for *Document Object Model*, which refers to "the hierarchical structure of HTML" (p.50).
- **CSS**: *Cascading Style Sheets*. How to use CSS: Check here.
- In CSS, a child adopts the style of the parent.
- **JavaScript**: Arrays, Objects, JSON (JavaScript Object Notation)
- **SVG**: Scalable Vector Graphics

Chapter 2

Task 5

From Day 6

- 5-1. generate a `dataset` made up of five numbers: 5, 10, 15, 20, 25;
- 5-2. generate five lines of texts, each line of text being “New paragraph!”;
- 5-3. generate five lines of texts: “I can count up to 5”, “10”, …
- 5-4. make all the five lines red;
- 5-5. make only lines in which the number is above 15 red, and all else lines black.

Chapter 3

Task 6: Drawing with Data

3.1 Tasks 6-1

From Day 8

- 6-1-1. Make a bar chart, in which the height of each bar corresponds to its data value.
  - dataset = [5,10,15,20,25];
  - Attributes of the CSS, `div.bar`: `inline-block`, set the width to be `20px`, the height `75px`, and background color to be teal;
- 6-1-2. Make the bar five times taller, and add some space (2px) between bars;
- 6-1-3. Change the dataset to be made up of 25 random numbers that range from 0 to 30 (using `for` loop, `Math.random()`, and `.push()`);
- 6-1-4. Make all the random numbers whole numbers (using `Math.round()`)

3.2 Task 6-2: Drawing SVGs

From Day 8

- 6-2-1, dataset = [5, 10, 15, 20, 25]
- 6-2-2, Create a SVG within `<body></body>`; Set the width of the SVG to be 500, and the height to be 50;

- 6-2-3, `var svg = d3.select("body").append("svg")`, what does this line of code do?
- 6-2-4, Draw five circles, the x-pistion value of each circle's center is (index * 50) + 25, the y-position is half of the pre-defined height, and the value of the radius is the data value itself;
- 6-2-5, Fill the five circles with "yellow", the stroke being "orange", and the stroke width is half of the corresponding data values;
- 6-2-6, Delete all the above codes except for the `div.bar{}` CSS styling, and then set the dataset to be `[ 5, 10, 13, 19, 21, 25, 22, 18, 15, 13,11, 12, 15, 20, 18, 17, 16, 18, 23, 25 ]`, generate `n` number of divs before the enclosing tag of `<body>`(n = the length of the above dataset; and yes, you still need the CSS between `<head> and </head>`), give it a class named as "bar". The height of each bar is five timees the corresponding data values. Attributions of the CSS, `div.bar`: inline-block, set the width to be 20px, the height 75px, background color is teal, and create a 2px margin between bars.

3.3 Task 6-3: Draw a new SVG chart

From Day 8

- 6-3-1. Still use the above dataset with 20 data values;
- 6-3-2. Insert a new `SVG\` between `<body>` and `</body>`. Set the width of the SVG to be 500 and the height 100 pixels;
- 6-3-3. Now, instead of creating `divs`, we'll be creating `rect` s. Create 20 `rect` s, each with x =0, y=0, width = 20, height = 100;
- 6-3-4. Set the `x` to be dynamic, such that the ith rect has a x of `i*21`;
- 6-3-5. Set the `x` to be flexible, such that each x = i * (w / dataset.length); Set the `width` of each rect to be flexible as well, each `width = w / dataset.length - 1`;
- 6-3-6. Try to set the height of each rect to be the corresponding data value;
- 6-3-7. Set the top of each bar to be `h-d` and the height of each bar to be the corresponding data value;
- 6-3-8. Change `d` to be `d * 4`;
- 6-3-9. Chagee the bar color to `teal`.

3.4 Task 6-4: Change bar color

From Day 9

- Based on Task 6-3, change the color of the bars using RGB. Let red and green components be fixed at zero. Let the blue component be the corresponding data value times 10, which then is rounded to the nearest whole number (using `Math.round()`)
  - **I still do not understand why we should put the codes in between `"+ +"`. This is so difficult for me**.

3.5 Task 6-5: Adjust labels

From Day 9

- 6-5-1. Add value labels on top of each bar;
- 6-5-2. Move each value label down by 15 pixels and to the right by 5 pixels; Style the font. Use sans-serif, 11px, and in white.

3.6 Task 6-6: Draw a scatterplot

From Day 10

- 6-6-1. First, create a SVG element before the enclosing tag of `<body>`. Set the width of the svg to be 500, and the height 100. Dataset = [[5, 20], [480, 90], [250, 50], [100, 33], [330, 95], [410, 12], [475, 44], [25, 67], [85, 21], [220, 88]]. In each hard bracket, the first number is the x, and the second the y. Draw a scatterplot of all these data points. Basically, you are expected to draw small circles. `cx` and `cy` are the above data points. Set the radius to be 5 pixels.
  - In the above `dataset`, `dataset[2][0] = 480`
- 6-6-2. Label each data point in the format of `d[0], d[1]`. Place each label right beside each data point (just use d[0] as x, and d[1] as y will do the job). Please note here that `text` can both be an SVG element and a D3 method.

Chapter Six, Drawing the data is finally over. We'll begin learning **scales**. p.117

3.7 Task 6 Review

3.7.1 Task 6-1 Review

From Day 9

- 6-1-1:
  - To specify the height of each bar, I should use `.style()`, rather than `.attr()`. In `D3.js`, `.style()` is to assign CSS styles to a HTML element. In CSS, we won't use `style` because everything is `style()`. In HTML, we do use it. For example, `<div style="height: 75px;"></div>`;
  - To make the height of each bar correspondes to its data value, I should use`function(d){return d + "px"}` rather than `function(d){return d }`;
- 6-1-2: To add some space (2px) between bars, I need to use `margin-right: 2px` in the CSS (`div.bar{}`. To learn more about the differences between `padding`, `border`, `margin`, read this amazing tutorial;
- 6-1-3:
  - First, I need to initilize an empty array using `dataset = []`
  - In Javascript, `Math` is a global object, whereas `.push()` is an array method.
  - A `for` loop: `for (initialization; test; updata){ }`

3.7.2 Task 6-2

From Day 9

- 6-2-4
  - When drawing five circles, we used three terms, "svg", "circles", and "circle", which of them should be fixed, and which can be random? "svg" and "circle" should be fixed, but "circles" can be random;
  - One thing I still don't understand is that in Task 6-1-1, I have to use `function(d){return d + "px"}` rather than `function(d){return d }`. However, in Task 6-2-4, using `.attr("r", function(d){return d })` is fine, and in `.attr("cy", h/2)`, I don't need to put `""` to enclose `h/2`. I guess it's because of **the differences between `.style()` and `.attr()`**. But **how different? I don't know.**;

  - When we need the index of the data values, we should use `function(d,i){}` even if we don't need the `d`. However, when we only need the `d`, we can simply use `function(d){}`.

- 6-2-6: Again, we need to add `px` in `style`

3.7.3 Task 6-3

From Day 9

To change the color of bars, we need to use `.attr("fill", "teal")`, rather than `.style()`.

3.7.4 Task 6-1

From Day Ten

- 6-1-1. Again, I forgot the difference between `.attr()` and `.style()`.
  - `.style()` assigns **CSS styles (properties and value)s** to **a HTML element**, just like the way we assign CSS styles when we are writing HTML. For example, `<div style="height: 75px;"></div>`.
  - In comparison, `.attr()` sets **an HTML attribute and its values** on a HTML element. Note that an element's class is stored as an HTML atrribute.

- 6-1-3. `A.push(B)` will put B to the end of A. `.push()` is an array method.

3.7.5 Task 6-2

From Day Ten

- 6-2-2.
  - `<svg>` is just like an HTML element. `width`, `height` are its attributs. `rect`, `circle`, `text` are all elements. `x`, `y`, `cx`, `cy` and `r` are all attributes. Read this amazing tutorial to really understand what is an element and what are attributs.
  - **I still did not understand it. Why is that I need to use `style()` to specify the `height` of `div` in** Task 6-1-1** but I need to use `.attr()` to specify the `height` of a `svg` here??** I guess it's because that for `div`, `height` is a CSS property, but for `svg`, `height` is an attribute? Maybe.

- 6-2-4. I need to use `svg.selectAll('circle')` rather than `svg.select('circle')`. Think about why for a moment;
- 6-2-5. As the amazing tutorial mentioned, here, using either `.attr()` or `.style()` is fine. The key difference is that I need to use `return d / 2 + "px"` for `stroke-wideth` when using `.style()`.

3.7.6 Concluding words on .style() and .attr():

- For `<div>`, `height` is a CSS property, so I should use `style`;
- For `<svg>`, `height`, `width`, `x`, `y`, `cx`, `cy`, `r` are attributes, so I should use `attr()`;
- For `<svg>`, `fill`, `stroke`, and `stroke-width` are also attributes. I can use either `style()` or `attr()`, but `style()` is better because it is not about **position** and **size**.
- Refer to this tutorial and this tutorial.

3.7.7 Task 6-5

From Day Eleven

- 6-5-1. When adding value labels on top of each bar, I should not use `rect.selectAll("text").data(dataset).enter().append("text")....` Doing so will add twenty `text` element within each `rect` element! This is not I am looking for at all. Rather, I should use `svg.selectAll("text").data(dataset).enter().append("text")`.
- 6-5-2. `.style("font-size", "11px")`.

3.7.8 Task 6-6

From Day Eleven

- 6-6-2. Whenever you use `function(){}`, remember to add `return`.

3.7.9 Task 6-6

From Day Eighteen

Chapter 4

Task 7: Scales

Basic Knowledge about Scales:

1. Why using scales? From Day 11

Scales are **functions** that map from an **input domain** to an **output range**.

Why do we need scale. Imagine we have this dataset: [1, 2, 4, 10, 100, 1000, 50000, 10000000000000]. Now, I ask you to draw a bar chart of this dataset, with the height of each bar corresponding to each data value. The first method we can think of is that the first bar will be 1 pixel tall, the second 2 px tall, the third 4 px tall. But the last data value will be 10000000000000 tall! Can you imagine how big a display we need to accommodate this value?

Instead, we might better set the height of the last bar to be 1000, which is reasonable for most displays people are using today. Then, what will be the height of the third bar that corresponds to the value of 4? That's what **scales** do!

2. Understanding Array From Day 13

What is an array? According to Scott's book, an array is "**a sequence of values**, conveniently stored in a single **variable**."

3. How to find the largest number in a dataset

If the dataset is a one-dimensional array, then it's very simple. Just use the `d3.max()`function is okay.

For example:

```
var dataset_01 = [4, 5, 9, 10, 24, 56]

d3.max(dataset_01)  //56
```

However, things get a little bit more complicated when the dataset is two dimensional: an array of arrays. Simply put the dataset into `d3.max()` won't give us the largest number.

```
var dataset_02 = [
  [0, 4], [5, 9], [8, 90], [47, 33], [55,99]
  ]

d3.max(dataset_02) //[8, 90]
```

Okay, what should we do then? If we want to identify the largest number among all the first number in each array within `dataset_02`, we can do it this way:

```
d3.max(dataset_02, function(d){
  return d[0];})
```

Here, `function(d){return d[0]}` is called an *accessor function*, as it helps us **access** the first number in each array.

How to understand the above code? I would understand it this way. `d3.max()` need to first have all the numbers we ask it to evaluate, in the example above, all the first numbers in each array. To get the first numbers, we need to use the accessor function. Then, there is a problem, what does the `d` mean? `d3.max()` doesn't know, so we have to specify that. We tell `d3.max()` that each `d` comes from `dataset_02`, then the stupid `d3.max()` will know where to find each `d`.

4.1 Task 7-1 to 7-9

From Day 13

- 7-1. Based on Task 6-6. For the X coordinate, set the input domain to be [0, the largest number of d[0]], and the output range to be [0, w]. For the Y coordinate, set the input domain to be [0, the largest number of d[1]], and the output range to be [0, h].

Change the location of each circle and annodated text accordingly. For the text, use "sans-serif" in 11px and in red.

- 7-2. Let the circles with larger y values be at the top of the plot and those with smaller values at the bottom. *Hint*: change the output range of `yScale`.

- 7-3. To keep the circle elements 20 pixels away from the boarder of the SVG. *Hint*: create a padding of 20 and then change the output range of both `xScale` and `yScale`.

- 7-4. Texts on the far right are still being cut off, solve this problem. *Hint*: adjust the `range` of `xScale`.

Note that to create paddings on each side of our charts, a better way is Mike Bostock's **Margin Convention**.

- 7-5. Use the method of *Margin Convention* mentioned above. top: 20, right: 30, bottom: 30, left:40

- 7-6. (**Please note that this is not the ideal way to scale cirlces! We'd better scale them by area rather than by radius. This is just for illustration and for the sake of exercise!**) Create a custom scale for radius such that the greater the y value, the bigger the radius. Let the domain be [0, the largest number of y value], and the range be [2, 5].

- 7-7. Add an array to the original dataset, [600, 150].

- 7-8. Change the width of the SVG to be 600, and the height 300.

- 7-9. Tell me what the functions of `nice()`, `rangeRound()`, and `clamp()` do and how to use them.

4.2 Task 7-10 & 7-11

From Day 18

- 7-10. Check out the element inspector after Task 7-1-9 is done. You'll find that the numbers for `cx` and `cy` are pretty long. Try to round up those numbers using the `rangeRound()` method;

- 7-11. Scale those circles by area rather than by radius, such that the bigger the `y` value, the bigger the area of a circle. Let the area be within [0, 10]. *hint* : use `d3.scaleSqrt()`, and remember to replace `rScale` with `aScale`.

  - Please note that for `aScale`, the starting point for both the input domain and the output range **must** be zero. If you don't understand why, read my blog post on d3.js scales.

  - Why should we use `d3.scaleSqrt()` here: If the `y` value of data point A is four times as big as that of data point B, then we want to make sure that Circle A's area is also four times as large as that of Circle B. To maintain this ratio, we need to make sure that Circle A's radius is **twice** as large as that of Circle B. `d3.scaleSqrt` maps from the input domain to the output range using this fomula: `y = m*x^(1/2) + b`, where `x` represents the input domain (i.e., the y value of data points, or the area of Circel A, B, C…) and `y` represents the output range (i.e., the radius of Circle A, B, C…). In our case, sine the starting point for both the input domain and the output range is zero, `b = 0`. Therefore, `y_1 / y_2 = (x_1 / x_2)^(1/2)`.

4.3 Task 7 Review

4.3.1 From Day 18

- 7-1. You cannot use the function of `max` by calling itself. You have to call `d3` first, just like when you are using `scaleLinear()`. This is because both `scaleLinear()` and `max()` are `d3` functions or methods.
- 7-5.
  - My original understanding of **margin convention** is wrong. There isn't magic involved here. What `var margin = ({ })` does here is simply assign numbers to names, so that those numbers can be referred to by calling `margin.top`, `margin.left`….
  - If the range of `yScale` was initially `[h, 0]`, do pay attention when you use margin convention. If you can do it right, you have clearly understood margin convention. *Hint* : **It should be `.range([h - margin.bottom, margin.top])`**.
  - Why is that we should use `.range([h - margin.bottom, margin.top])` here? The domain here is ([0, the maximum of d[1]), and we want circles with smaller d[1] values appear at bottom, so the range should start from `h` something. Now, suppose `margin.top = 20`, and `margin.bottom = 40`. If we use `.range([h - margin.top, margin.bottom])`, then `0` in the input domain will become `h - margin.top`, and that will be the lowest point in our chart. Is this what we want? Not really. I want the lowest point to be at `h - margin.bottom`.

4.3.2 From Day 19

- 7-1. The order of codes does matter in D3.js. If I put `xScale` and `yScale` before `var dataset = ...`, then D3 would not be able to generate any

results. This because I need to call `dataset` in `xScale` and `yScale`.

4.3.3 From Day 22

7-1-5. In `var margin = ({top: 20, right: 30, bottom: 30, left: 40});`, you **should not**, and in fact **cannot**, put `;` right after `left:40`.

Chapter 5

Task 8: Axes

5.1 Task 8-1 to 8-6

From Day 19

- 8-1. Create an axis on top of the chart for `xScale`. Assign a class named "axis" to the group element you just created at the end of the `svg` element. *hint* : think about whether you need to use `d3.axisTop()` or `d3.axisBottom()`.
- 8-2. Push the axis to the base of the chart using SVG *transformations*.
- 8-3. Style `path` and `line` in `.axis` such that they are in teal (*Hint*: using `stroke: teal;`). Style `text` such that it is Optima, sans-serif, bold, 14px, and in teal.
- 8-4. Set the rough number of ticks to be 5. // D3 might override it, though.
- 8-5. Only label 0, 100, 250, and 600, using `tickValues([])`.
- 8-6. Generate a Y-axis on the left. Set the rough number of ticks to be 5. Push this axis to the right by `margin.left` px. Also assign the class of "axis" to it.

5.2 Some key points about `Math`

From Day 20

- `Math.random()` will generate random numbers less than 1.0.

- If `random = 8.9999`, then `Math.round(random) = 9` but `Math.floor(random) = 8`.
- `Math` is **parallel** with `d3`, so `d3.Math....` is unnecessary and wrong.

5.3 Task 8-7 to 8-9

From Day 20

- 8-7. Let the dataset be made up of 50 pairs of random numbers between 0 and 100 that are integers.
- 8-8. Remove labels.
- 8-9. Treat values on x-axis as percentages with one decimal point precision. *Hint* : using `d3.format("")`, and `.tickFormat()`.

5.4 Task 8 Review

From Day 22

- 8-1. `g` element is a `group` element. It is an SVG element, just like `line`, `circle`, and `rect`. One thing different is that `g` element is invisible. It serves two functions:
  1. Helps us group other elements;
  2. Accepts transformations onto it, which is very handy.
- 8-7. `for (var i = 0; i < 50; i++){}`. Two points to remember:
  1. It should be `var i = 0`, rather than simply `i = 0`;
  2. Seperate the codes inside the bracket by `;`, not `,`.

Chapter 6

Task 9: Updates & Transiations

6.1 Task 9-1

6.1.1 From Day 20

- 9-1-1. Make a normal-looking bar chart for this dataset: `[ 5, 10, 13, 19, 21, 25, 22, 18, 15, 13, 11, 12, 15, 20, 18, 17, 16, 18, 23, 25 ]`. Width and height of the SVG: 600, 250.

6.1.2 From Day 21

- 9-1-2. Create an `xScale` using `d3.scaleBand()`. Let the output range be `[0, w]`, and assign 5 percent of the bandwidth to spacing in between bands. Use this `xScale` when setting `x`, and `width` of each `rect` accordingly. *Hint* :

    1. For `.domain()`, use `d3.range()`. For example, `range(10)` will produce `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`. **Note** that stop is exclusive.
    2. `bandwidth()`.

- 9-1-3. Put the text of corresponding data values right in the middle of the top of each bar. Think hard about how to set the `x` (*Hint* : You'll need to use `xScale(i)`, and `xScale.bandwidht()`.

- 9-1-4. Let the text "Click here" appear above the bar chart. When visitors click "Click here", the text "Hey, don't click that" will pop up.

  - *Hint* : Add a paragraph element before our javascript. Down at the end of our d3 code, select this new p, and add an *event listener* to the element. You'll use the `alert()` function within a `on` function.
  - What is an *event listener*: an anonymous function that listens for an event on an element or elements.
  - The `.on` method takes on two arguments: the event type, and the listener itself which is an anonymous function.

- 9-1-5. Rather than generating the annoying pop-up text, updating the dataset. That is to say, after people click on the text, the bar chart will represent this updated dataset: `[11, 12, 15, 20, 18, 17, 16, 18, 23, 25, 5,10,13,19,21,25,22,18,15,13]`;

- 9-1-6. Add transition to the bars heights.

- 9-1-7. Let the transition last for 1 second. Now, we can see that labels did not have smooth transitions along with the bar heights. Add transition and duration to labels as well.

- 9-1-8. Make the rate of motion be linear using the `ease()` function. *Hint* : `easeLinear`.

- 9-1-9. Create a one second delay before each transition.

- 9-1-10. Make the delay dynamic (so called "staggered delays") such that each element is delayed 100 ms more than the preceding one.

6.1.3 From Day 22

- 9-1-11. Continue with 9-1-10. To scale the dynamic delay to the length of the dataset such that visitors need to wait for at most 1 second to see the last rect element move.

  - Why? If we only have 25 data values, 9-1-10 might be okay. However, if we have 1000 data points, then visitors have to wait for a very long time to see the transition finish. Scaling the delay to the length of the data makes more sense.

6.2 Task 9-2

Continue with Task 9-1-11.

6.2.1 From Day 23

- 9-2-1. Make changes so that every time you click, new data will show up with transitions. The new data will be an array of random **integers** with values between 0 and 24. Let the new array have the same length as the original one (the one before any updates). *Hint* :
  - Use a `for` loop;
  - `Math.random()` generates random numbers from 0 up to, but not including 1.
- 9-2-2. Add extra logical to set the vertical placement and style of labels. "When the data value is less than or equal to 1, place the label up above the bar and set its fill to black. Otherwise, place and style the label normally" (p. 167). *Hint* :

```
if () {}
  else {}
// Since we use function (d) { }, we need to add "return" in our if statement.
// Also note that we still need to fill out the else {} part. We cannot leave it as blank.
```

- 9-2-3. We earlier used `Math.random() * 25`. Replace `25` with a variable `maxValue`. Set the `maxValue` to be `100`.

Note: We'll stop editing bar charts temporarily starting from 9-2-4.

- 9-2-4. Based on the results we got from Task 8, do the following: 1. Delete the CSS for .axis path, line, and text; 2. Let the radius of each circle be a constant: 2 px. Then, do the following:
  1. Rename the class of xAxis as `x axis` and that of yAxis as `y axis`.
  2. When people click on the text "Click here for new data", a new pair of 50 integers between 0 and 100 will show up. Make transitions to these changes and set the duration to be 1 second. Make sure that scales, and both axes are also updated. *Hint* : to update the axes, first select the axis, and make a transition, set the transition's duration, and call the axis generator.
- 9-2-5. Make changes to the code so that when you click the trigger, the fill of all the circles changes to magenta, and the radius becomes 3 px. When the transition is over, the fill changes to black and the radii become 2px.
  - *Hint* : Within circle-updating code, use two `.on()` statements, whithin which you need to first specify whether it's `start` or `end`, and then use an anonymous function to first select the current element and set the attributions.

- 9-2-6. Add a transition (which lasts for 1 second) to the start of the transition and see what will happen.

6.2.2 From Day 25

- 9-2-7. Modify the code so that when you click the trigger, each circle will turn pink and its radius immediately increases from 2 to 7. Then circles move (1 second) to their new positions. In the end, circles **transition** (1 second) to original color and size.
- 9-2-8. "Instead of reelecting elements and calling a new transition on each one with `.on("end", ...)`, just tack a second transition onto the end of the chain." (p. 174). This chaining approach is recommended when "sequencing multiple transitions".

6.2.3 From Day 28

- 9-2-9. **Go back to our bar charts.** Based on codes from Task 9-2-3. Make changes so that every time you click the trigger, a new integer between 0 and 24 will be added. Update both the bars and the text labels. The newly generated bars and labels will move from the right border of the SVG to its proper place. Make the transition last for half a second. *Hint* : Use `.merge()`.

6.2.4 From Day 31

- 9-2-10. Create a `p` element with the text "Click here to remove data". Every time you click it, the first bar and its associated label will be removed. Add a transition (half a second) to the removal so that the **exiting** bar will move to the bottom-right corner of the SVG and then disappear. Make sure to update the bars and labels.

Hint : - To remove the first data value from dataset, use `.shift()`;

- To remove svg elements (rect and text), use `.exit().remove()`. The `.exit()` function returns a reference to the exiting element; Transition syntaxes should be added between `.exit()` and `.remove()`;
- To make sure that the exiting bar moves to the bottom-right corner before disappearing, set its position after the transition and before the removal;
- After the removal, we need to update the bars and labels. That's fairly easy. For `rect`, just use `svg.selectAll("rect")`, and then set its position, width, height and color using `attr("x", ...)`.... For `text`,

first use `svg.selectAll("text").text(d => d)`, and then set texts' positions.

6.3 Unsolved puzzles in Task 9

6.3.1 Puzzle one from Day 26

Maybe the answer: See Day 30.

6.3.2 Puzzle two

Something I don't understand but don't have time to figure out:

- I want to let the exiting label to turn red and move to the upper-right corner before disappearing, so I specified `.attr("x", w).attr("y", 0).` after the transition syntaxes and before `.remove()`. However, I noticed that it just cannot move horizontally. It keeps within its original horizontal position.
- I just don't know why and how Scott Murray's codes worked. Why should we use `bars.enter()...` here?
- How to get the transition in Scott Murray's codes when updating the bars?
- I used `dataset.pop()`, which is supposed to delete the last data value. Yes, when I used

```
svg.selectAll("rect")
                .data(dataset)
                .exit()
                .transition()
                .attr("x", w)
                .attr("y", h)
                .remove()
```

The exiting bar traveled from the left! Why???

6.3.3 Puzzle three from Day 40

I was redoing Task 9-2-1. The answer is Chapter 09/17_randomized_data.html.

The only difference is that I have already updated the `xScale` and `yScale` within the bar and text updating codes. Also, I made a "small" mistake: I put `dataset = []` **inside** the `for` loop.

To make the changes more noticeable, change the color of the bar to red within within the bar and text updating codes. Here are the things I didn't understand:

- Why isn't the red obscuring the colors of previous rects?
- Why is there a larger gap between the start of the first bar?

See more information about d3 band scales here.

6.3.4 Puzzle four and five from Day 45

6.4 Translating from D3 to Observable

See Day 27

Some points worth mentioning:

- You can write codes involving multiple variables by including them in curly brackets. If you don't use curly brackets, you have to dedicate one cell to each variable. See this example if you don't know what "dedicating one cell to each variable" means.
- In the book, you normally write `var w = 600`. However, now `let` and `const` are preferred. Read here and here for explanations.
- To show the results, you need to use `return svg.node()`. Again, don't ask me why.
- By the way, `d => yScale(d)` is simply short for `function(d) { return yScale(d) }`. Similarly, `(d, i) => xScale(i)` means `function(d, i) { return xScale(i)}`

6.5 Understanding the `selectAll().data().enter().append()` pattern

6.5.1 From Day 29

- `svg.selectAll("placeholder-for-rects")` // This will select all placeholder elements and hand them off to the next step
- `.data(dataset)` // This will bind data to the selected placeholder elements, and then return references to all elements to which data was just bound. References to these elements are called "update selection".

- `.enter()` // Select elements that do not yet exist and hand them off to the next step.
- `.append("rect")` // Create a rect element within each element returned by `enter()`, and then return reference to this newly created element.

6.6 Task 9-1 Review

6.6.1 From Day 21

- 9-1-1. In `yScale`, if `.range(0, h)`, then the `y` axis should be `h - yScale(d)`. If `.range(h,0)`, then the `height` of bars should be `h - yScale(d)`.

6.6.2 From Day 22

- 9-1-2.
  - It should be `paddingInner()`, rather than `PaddingInner()`.
  - `{return "rgb( )"}`
- 9-1-3. `.attr("text-anchor", "middle")`
- 9-1-4. I should use `d3.select("p")`, rather than `svg.select("p")`. Why? Because the new `p` element is parallel with the `svg` element, I cannot select the `p` within `svg`.
- 9-1-6. The `.transition()` should be added within the anonymous function. It should be put after you have updated the data and before any changes.
- 9-1-8. It is `ease(d3.easeLinear()`, not `ease(d3.scaleLinear())`.
- 9-1-9. Scott Murray was wrong. The position of `.delay()` is not flexible at all. It mush be put AFTER `.transition()`.
- 9-1-10. Please keep in mind that if you have `.delay()` after the `.transition()`, then `duration()` is applied to each individual transition, rather than all transitions in aggregate.

6.6.3 Recap: functions you absolutely HAVE TO call `d3` first:

- d3.scaleLinear() …

- d3.max()
- d3.range()
- d3.easeLinear

6.6.4 From Day 23

- 9-1. `return "rgb(0, 0, " + Math.round(d*10) + ")"`. You are not allowed to have space between `rgb` and `(`.

6.6.5 From Day 24

- `xScale.bandwidth()`.
- `d3.range()`, rather than `range()`.
- Within the anonymous function of `.on()`, there is no need to write `return ...`

6.6.6 From Day 29

- 9-1-4.
  - `<p>Click Here</p>`. No need to use ””
  - `alert("phrase here")`
- 9-1-5. Even if you used `svg.selectAll("placeholder-for-rects")` to create and select placeholder elements to be used later, you still need to use `svg.selectAll("rect")` in `.on()` when you updating the data.

6.7 Task 9-2 Review

6.7.1 From Day 24

- 9-2-2. `if(){return } else{ return }`
- 9-2-4.
  - `d3.select("p)` rather than `d3.select(p)`
  - `.on("click", ...)` rather than `.on("Click", ...)`
  - `xScale.domain([0, d3.max(dataset, function(d){ return ...}])`. Be sure to add `dataset`.

- 9-2-5.
  - It should be `d3.select(this)`, rather than `svg.select(this)`.
  - "start" can be placed either before or after `duration()` and "end" should be placed at the end of the circle-updating code.
- 9-2-6. Only one transition can be active for a given element at any given time.

6.7.2 From Day 29

- 9-2-5. Within `.on("start", ...)`, you need to use `d3.select(this)`, rather than `svg.select(this)`.
- 9-2-7. You can add many transitions you want. However, you cannot add a transition within in `.on("start", ...)`, which is designed for immediate transformations, not interpolated ones.

For example, the following is okay:

```
svg.selectAll("circle")
     .data(dataset)
     .transtion()
     .duration(1000)
     .transition()
     .duration(1000)
     .
     .
     .
```

- 9-2-9.
  - `bars.enter().append("rect")...` will return references to newly created element. Therefore, all the codes that follow will be used for this newly created bar, until you use `merge()`.

Chapter 7

Task 10: Interactivity

Background knowledge:

`d3.select("p).on("click", function() {} );` binds an *event listener* to the p element.

7.1 Task 10

7.1.1 From Day 32

- 10-1. Make a bar chart for this dataset: `[ 5, 10, 13, 19, 21, 25, 22, 18, 15, 13, 11, 12, 15, 20, 18, 17, 16, 18, 23, 25 ]`. Set width of the SVG to be 600 and the height 250. Set the inner padding to be 5% of the bandwidth.
- 10-2. Tack on an `.on()` function when you create `rect` so that when you click each bar, that bar's data value will be shown in the console.
- 10-3. Add `CSS` so that when the mouse is hovering the element, the bar turn orange.
  - To enable this highlighting effect, make sure that you are using `.attr("fill", ...)`, rater than `.style("fill", ...)`.
- 10-4. Use `.on("mouseover", ...)` to achieve the same effect.
- 10-5. You will see that when the mouse leaves the bar, the color does not restore. Use `mouseout` to solve this problem. In addition, add a transition (1/4 second) to color restoration.

  - **Note** : In Scott Murray's book, 05_mouseout_html and 06_smoother.html of Chapter 10 did not work the way he mentioned in the book. When the mouse left a bar, it turns black, not its original color.
  - To solve this problem, reference nothing when defining the anonymous function right after `mouseout`. Then, use an anonymous function referencing `d` within `.attr("fill", ...)`.

- 10-6. Continue with 10-5. Mouse over a bar and then move the pointer right above the label of this bar. See what happens. Modify your codes so that when the pointer is on the label, the bar does not change its color. This can be done with either CSS or D3. Try both.

- 10-7. How to achieve the result of 10-6 by grouping SVG elements (p. 202, page number in the book, not that on pdf).

  - If you find it too difficult to implement, see Day 33 for an answer.

- 10-8. Continue with `10-6`. Add an event listener for a click event. Whichever bar you click, the bars will be sorted based on their associated data values. Add a transition (one second) to this process.

  - *Hint* : Add a click event right at the end of rects-creating codes. Name the event listener as `sortBars` or anything else you prefer. Later define this `sortBars`. Select all the `rects`, then use a `sort()` method whitin which we need to use a `comparator` function, to sort the data
  - d3.ascending()

7.1.2 From Day 33

- 10-9. Continue with 10-8. During the transition of sorting, if you mouse over some bars, they won't fall into place. Solve this problem by assigning names to transitions.

- 10-10. So far, you can only click once because a second click won't trigger any changes. Modify the codes so that a second click will trigger a re-sort, "placing the bars in descending order" (p. 206). Then a third click again will place the bars in ascending order, and a fourth click will…

- 10-11. Add a staggered delay to the transition so that each bar will move 50ms after the preceding bar moves.

- 10-12. First remove all the labels. Then, add browser default tooltips. When mousing over a bar, a tooltip of "this value is…" will show up. *Hint* : `.append("title").text(...)`

7.2 Task 10 Review

7.2.1 From Day 34

- 10-2. How to interpret the results of 10-2 in console? I didn't know.
- 10-3. There shouldn't be any space between `rect:` and `hover`.
- 10-7.
  - `.attr("transform", (d, i) => "translate(...)")`
  - Why didn't the following work? Why did I have to use `transform`? **I didn't understand**.

```
svg.selectAll("g")
   .data(dataset)
   .enter()
   .append("g")
   .attr("x", (d, i) => xScale(i))
```

- I tried the Scott Murray's codes implementing SVG element tooltips, but they didn't work. Here is the solution to this problem.

7.3 Unsolved Puzzles in Task 10

7.3.1 Puzzle one from Day 33