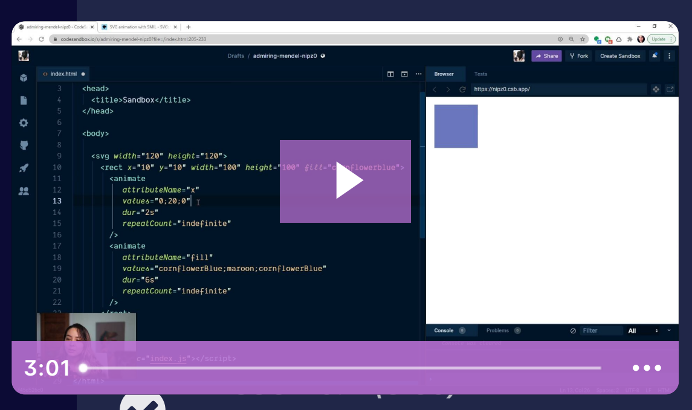


🙋 Ask a question

Fullstack D3 Masterclass › Animations and Transitions

Animations and Transitions

We talk about the different ways we can animate changes in our charts, and rule out SVG <animate>.

LESSON DISCUSSION 0

When we update our charts, we can animate elements from their old to their new positions. These animations can be visually exciting, but more importantly, they have functional benefits. When a bar animates from one height to another, the viewer has a better idea of whether it's getting larger or smaller. If we're animating several bars at once, we're drawing the viewer's eye to the bar making the biggest change because of its fast speed.

By analogy, imagine if track runners teleported from the start to the finish line. For one, it would be terribly boring to watch, but it would also be hard to tell who was fastest.

There are multiple ways we can animate changes in our graphs:

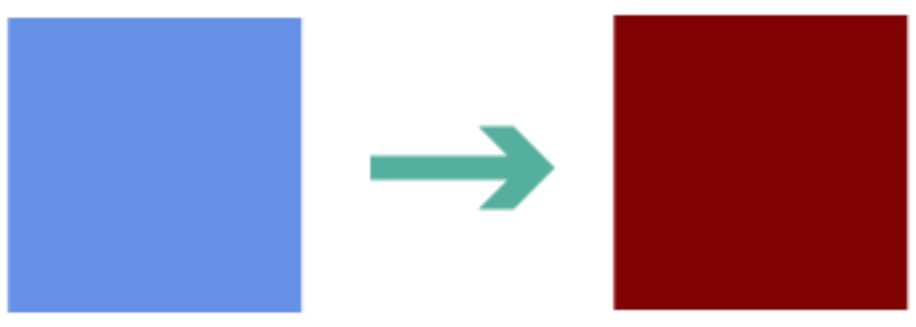
- using **SVG** `<animate>`
- using **CSS** `transition`
- using `d3.transition()`

Let's introduce each of these options.

SVG `<animate>`

`<animate>` is a native SVG element that can be defined inside of the animated element.

```
<svg width="120" height="120">
  <rect x="10" y="10" width="100" height="100" fill="cornflowerblue">
    <animate
      attributeName="x"
      values="0;20;0"
      dur="2s"
      repeatCount="indefinite"
    />
    <animate
      attributeName="fill"
      values="cornflowerBlue;maroon;cornflowerBlue"
      dur="6s"
      repeatCount="indefinite"
    />
  </rect>
</svg>
```



Unfortunately this won't work for our charts. For one, `<animate>` is **unsupported in Internet Explorer**, and its future is uncertain. But the bigger issue is that we would have to set a static start and end value. We don't want to define static animations, instead we want our elements to animate changes between two dynamic values. Luckily, we have other options.

Previous Lesson: Week 3: Exercise

Next Lesson: CSS transitions