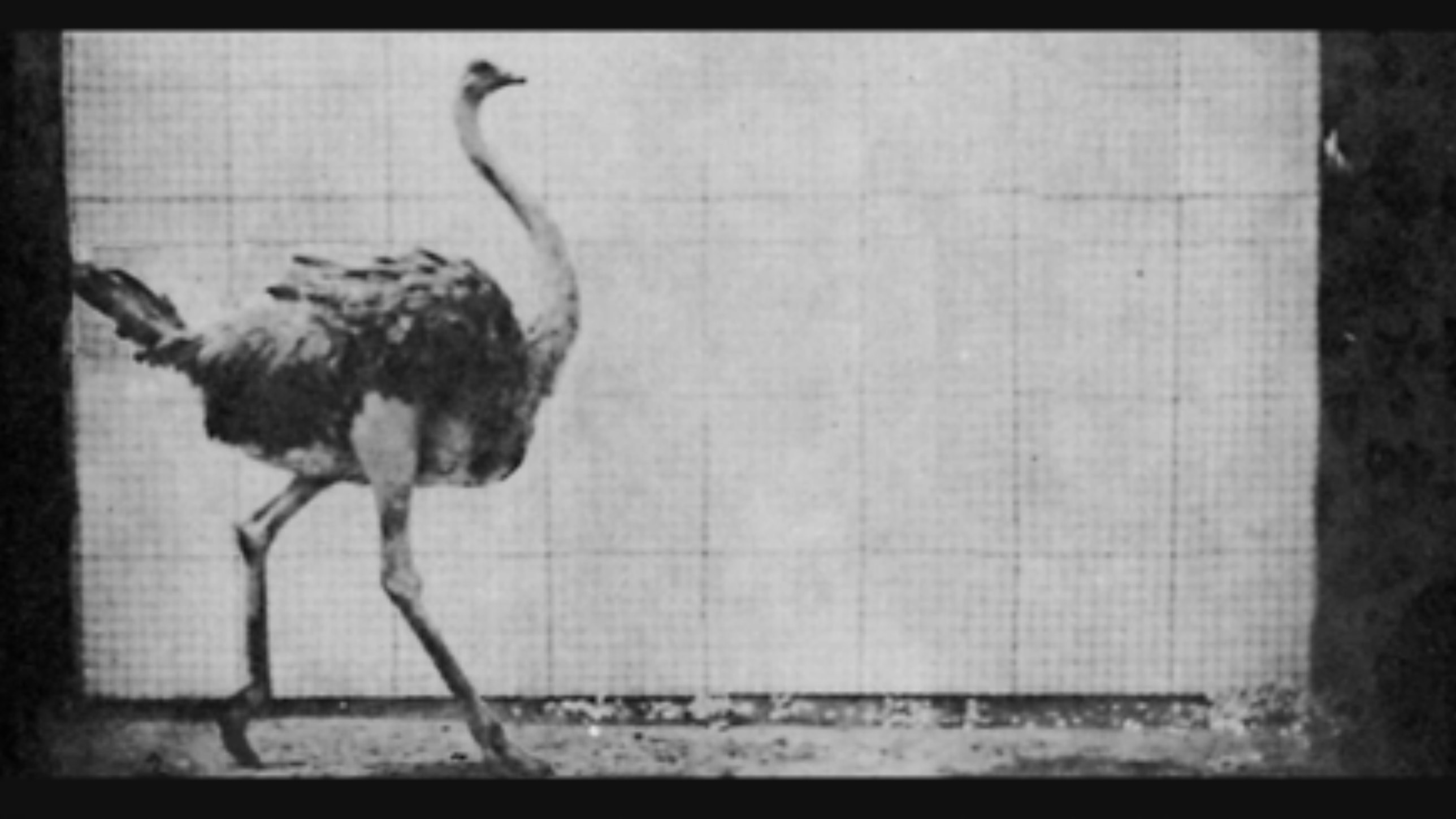
data visualization animations

react week 2019
friday, july 19th
aucher serr

@aucher_serr auchers
aucher.serr@gmail.com

but first...

a little experiment



@2nfo

@aucher_serr

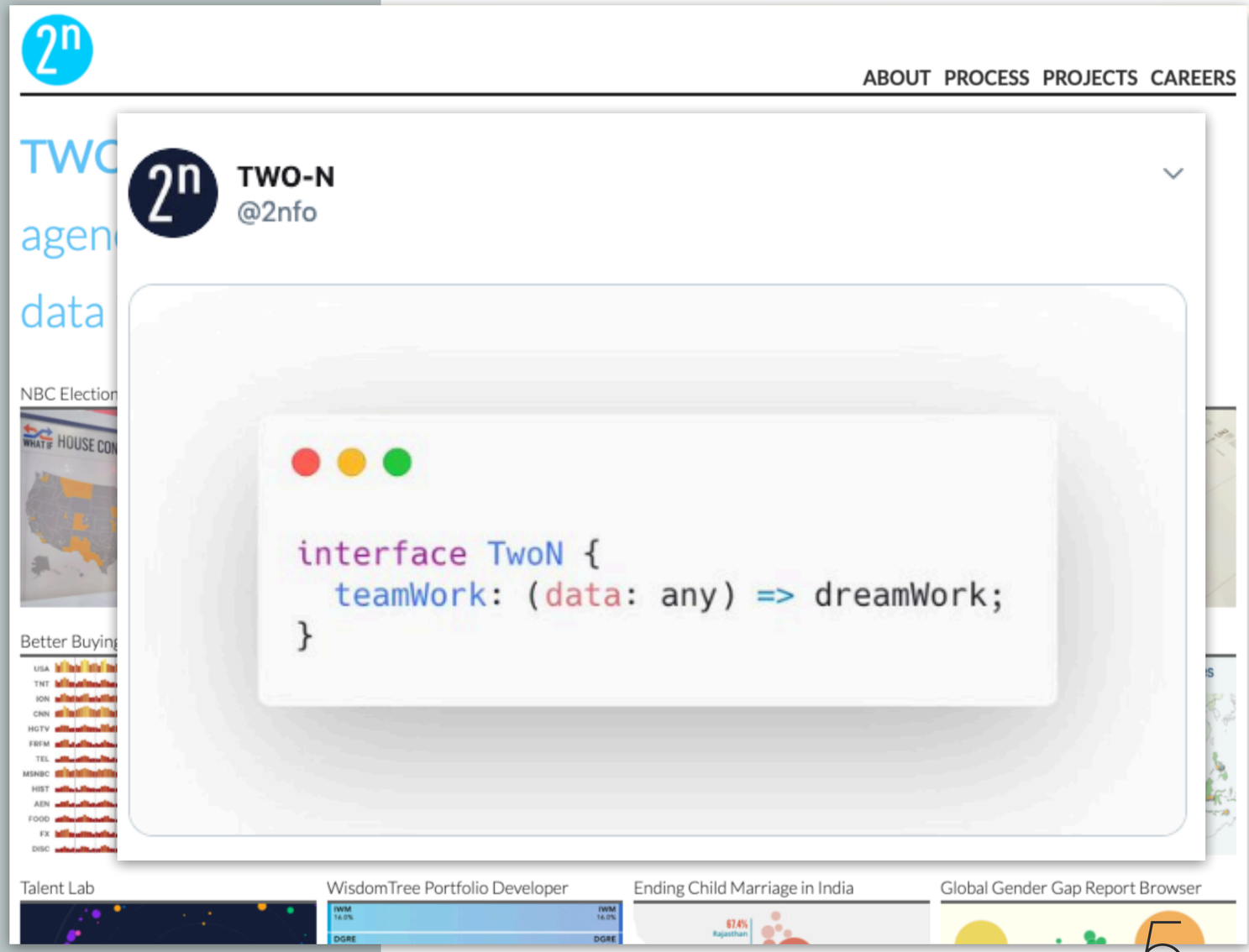
life is a journey...

(and so are many animations)

@2nfo

@aucher_serr

where I spend my days now



...we spend A LOT of time thinking about our animations

why animations?

a

sally she

nice

phone

forgot

had

day

her

sally had a nice day

although unfortunately , but — because

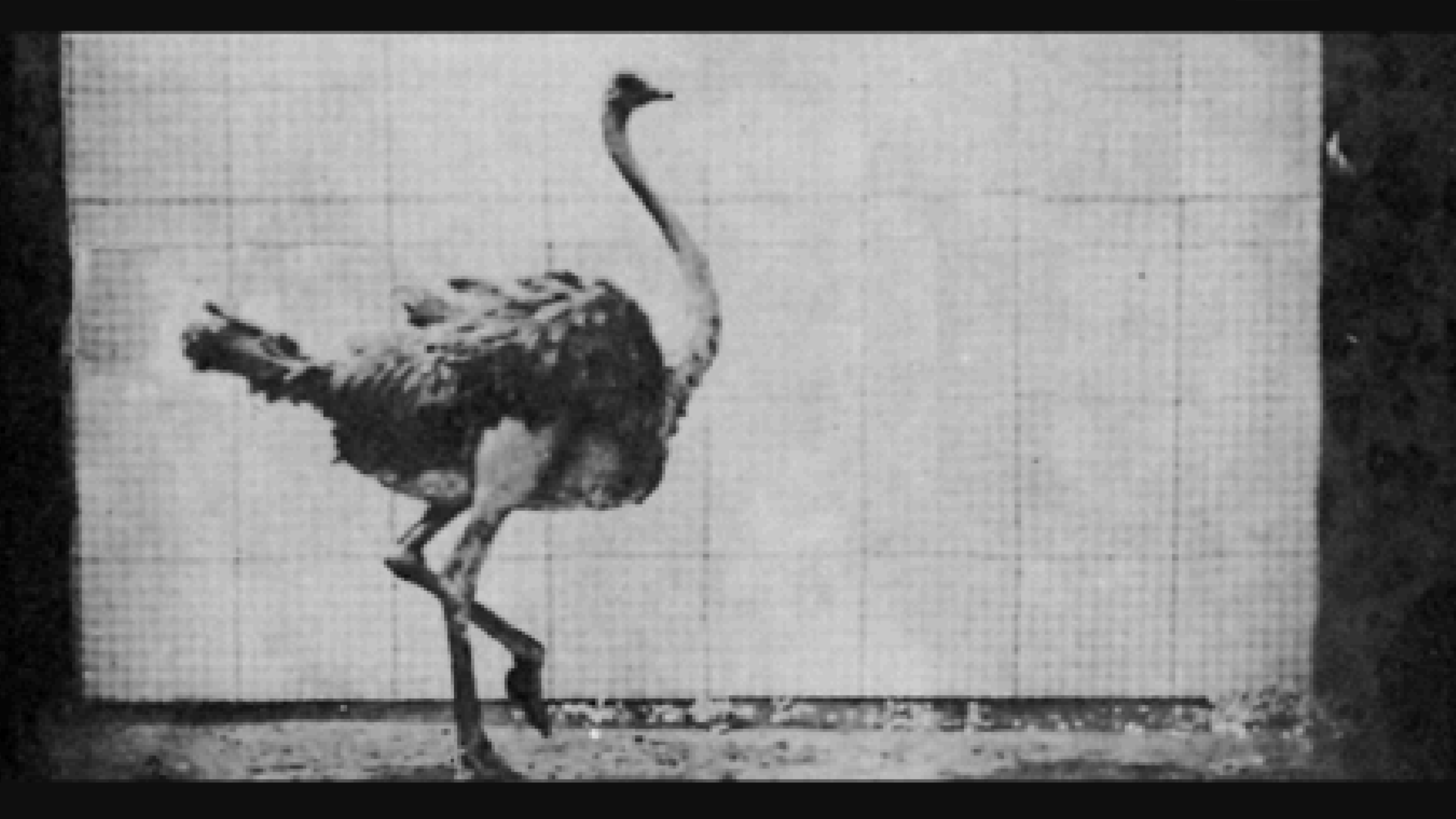
she forgot her phone

Proposition:

animations are to data visualization what punctuation/conjunctions are to language

"Grammar makes language expressive. A language consisting of words and no grammar (statement = word) expresses only as many ideas as there are words. **By specifying how words are combined in statements, a grammar expands a language's scope.**"

"Grammar of Graphics" by Leland Wilkinson



Global Gender Gap Report Browser

Since 2006, the World Economic Forum (WEF) has published an annual Global Gender Gap Report. In these reports, WEF "quantifies the magnitude of gender disparities and tracks their progress over time, with a specific focus on the relative gaps between women and men across four key areas: health, education, economy and politics."

The latest annual report, released in November 2016, covers 144 countries and marks the 11th installment. As the WEF notes, "more than a decade of data has revealed that progress is still too slow for realizing the full potential of one half of humanity within our lifetimes."

We analyzed this data and built an interactive story to explore it.

Scroll ↓

WORLD ECONOMIC FORUM

This interactive report was designed and built by TWO-N, based on data from the World Economic Forum. To learn more about The Global Gender Gap Report, visit the report's website.

Download the full Global Gender Gap Report by WEF (PDF)

Download full raw data (CSV)

2n

TWO-N is an award-winning data visualization agency based in New York City. We design and build interactive software that turns data into graphically delightful informative experiences. To learn more about TWO-N, visit two-n.com.

Extensive Data Shows Punishing Reach of Racism for Black Boys

By **EMILY BADGER, CLAIRE CAIN MILLER, ADAM PEARCE** and **KEVIN QUEALY** MARCH 19, 2018

Black boys raised in America, even in the wealthiest families and living in some of the most well-to-do neighborhoods, still earn less in adulthood than white boys with similar backgrounds, according to a sweeping new study that traced the lives of millions of children.

White boys who grow up rich are likely to remain that way. Black boys raised at the top, however, are more likely to become poor than to stay wealthy in their own adult households.

Follow the lives of 185 boys who grew up in rich families . . .

. . .and see where they end up as adults:

Rich adult

WHITE MEN 0 0%

BLACK MEN 0 0%

Upper-middle-class adult

0 0

benefits

Information Efficiency

we can pack in a lot more information by leveraging the temporal dimension.

Object Constancy

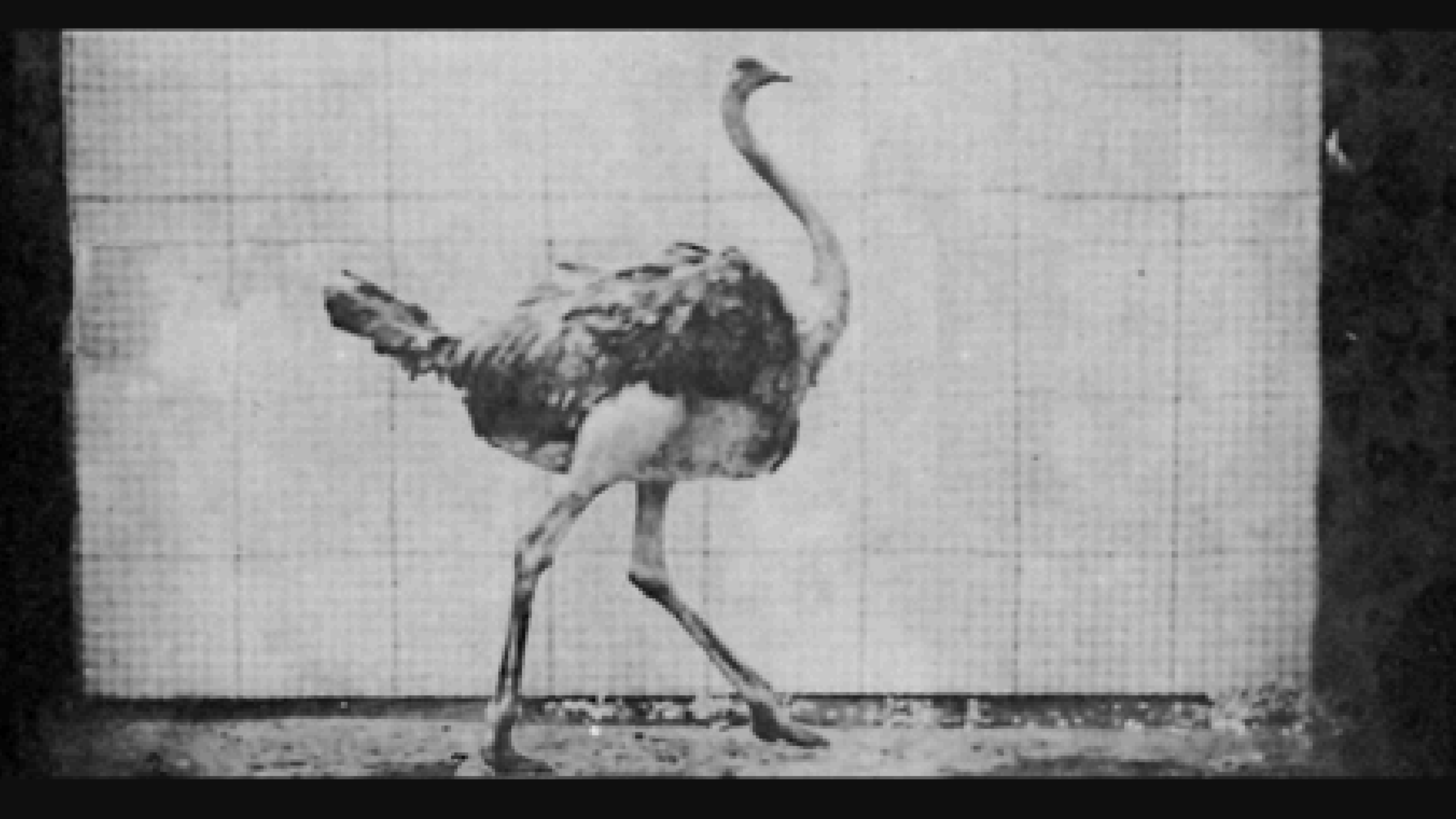
enables the user to track entities over time as they change form/shape/color.

Context and Way-finding

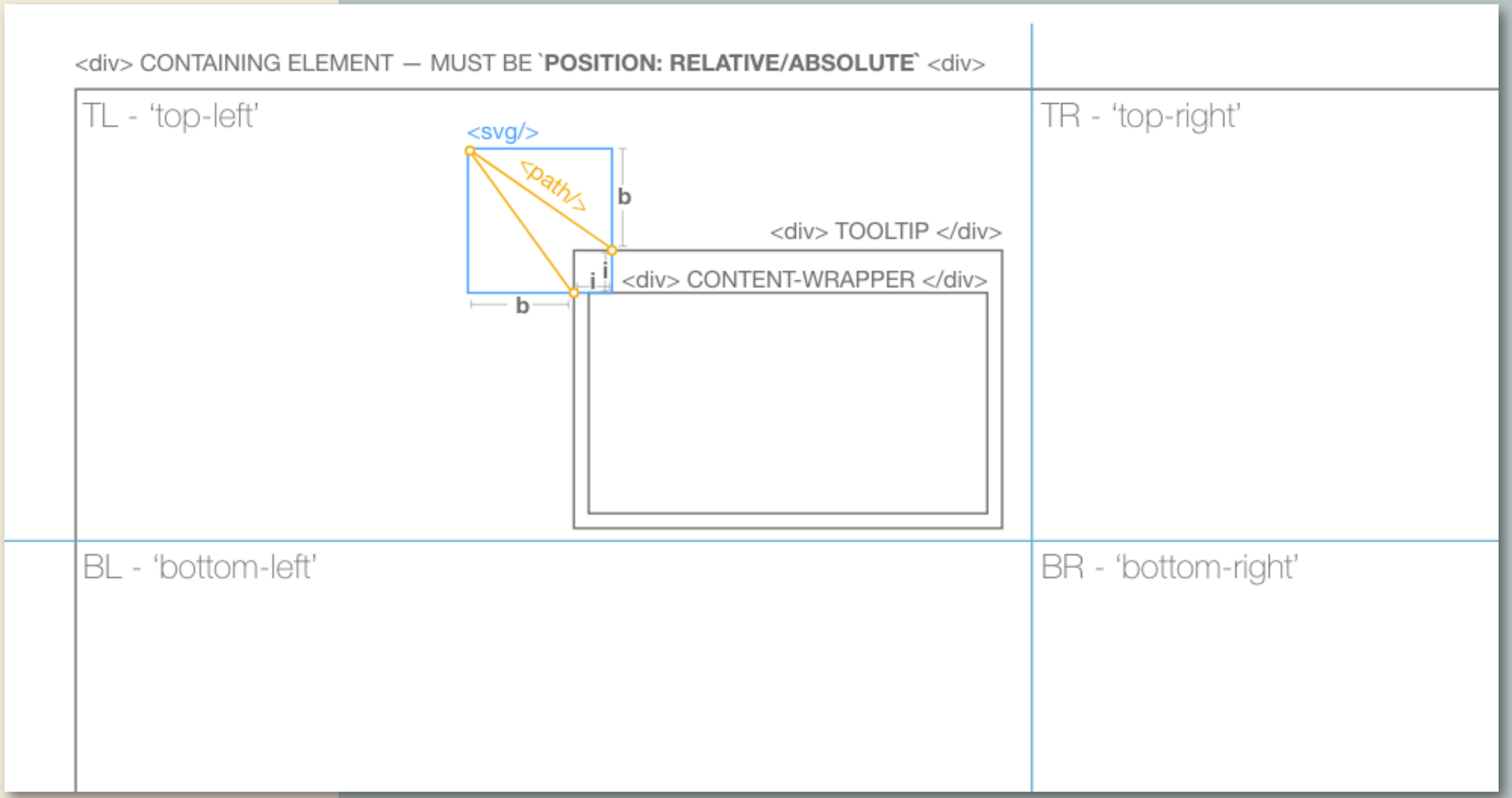
allows us to show how data is structured and how things relate to each other.

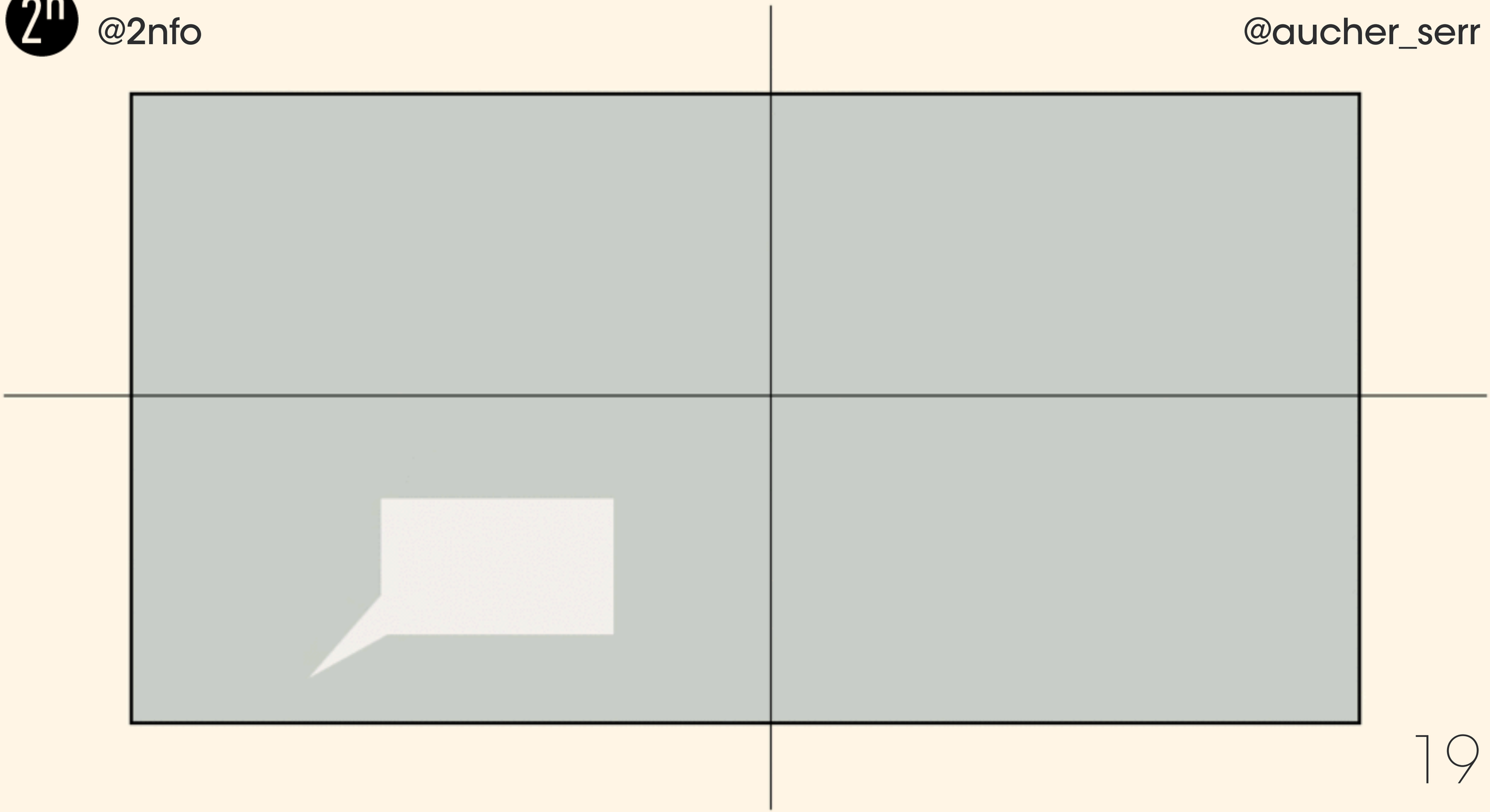
Engaging and Playful

drawing users in by giving them opportunities to explore data in new ways.



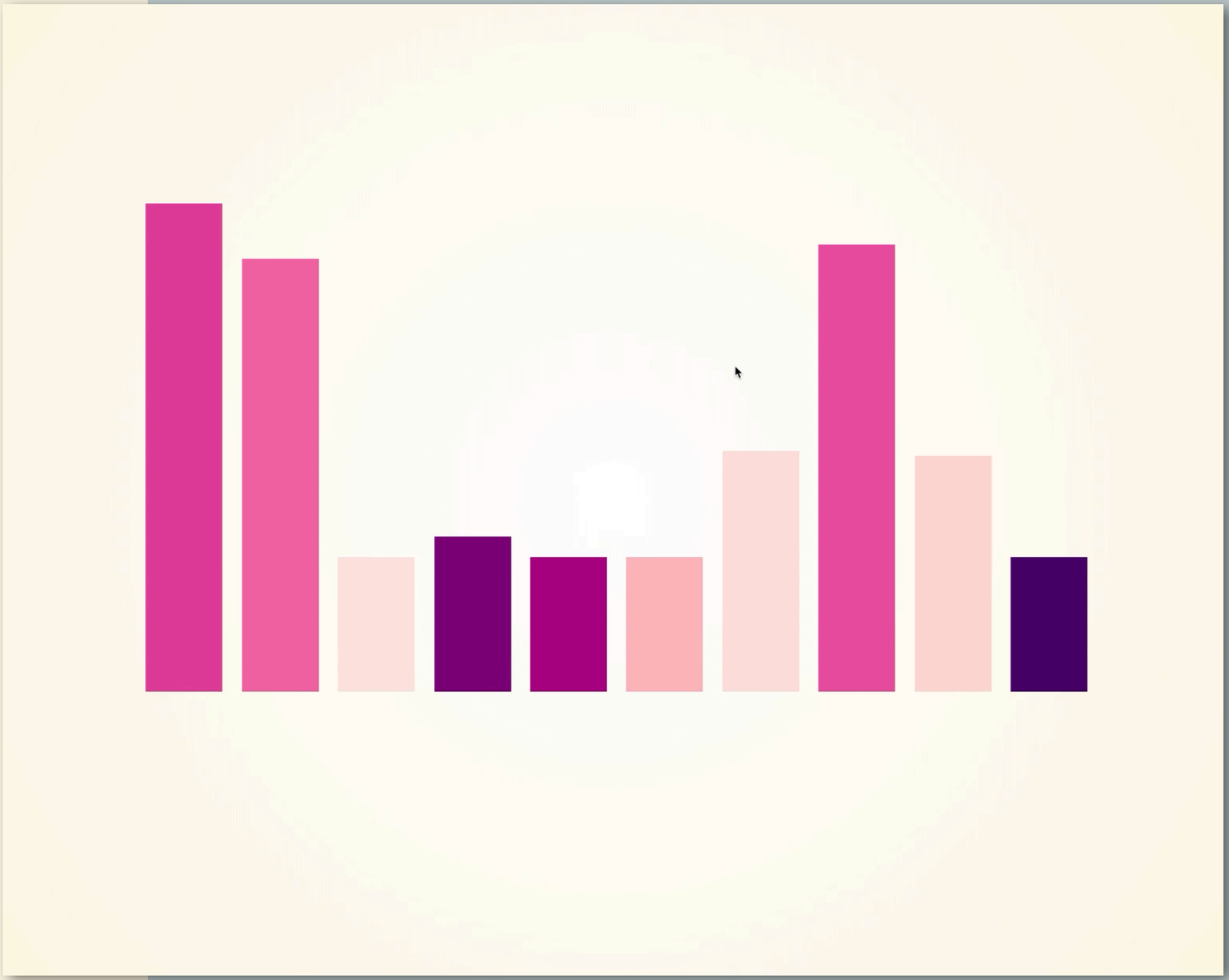
case study: our humble tooltip


 CONTAINING ELEMENT — MUST BE `POSITION: RELATIVE/ABSOLUTE`
TL - 'top-left'
TR - 'top-right'
<svg/>
<path/>
b
 TOOLTIP
i
i
 CONTENT-WRAPPER
b
BL - 'bottom-left'
BR - 'bottom-right'



starting point

tooltip shows up whenever user hovers over a data element

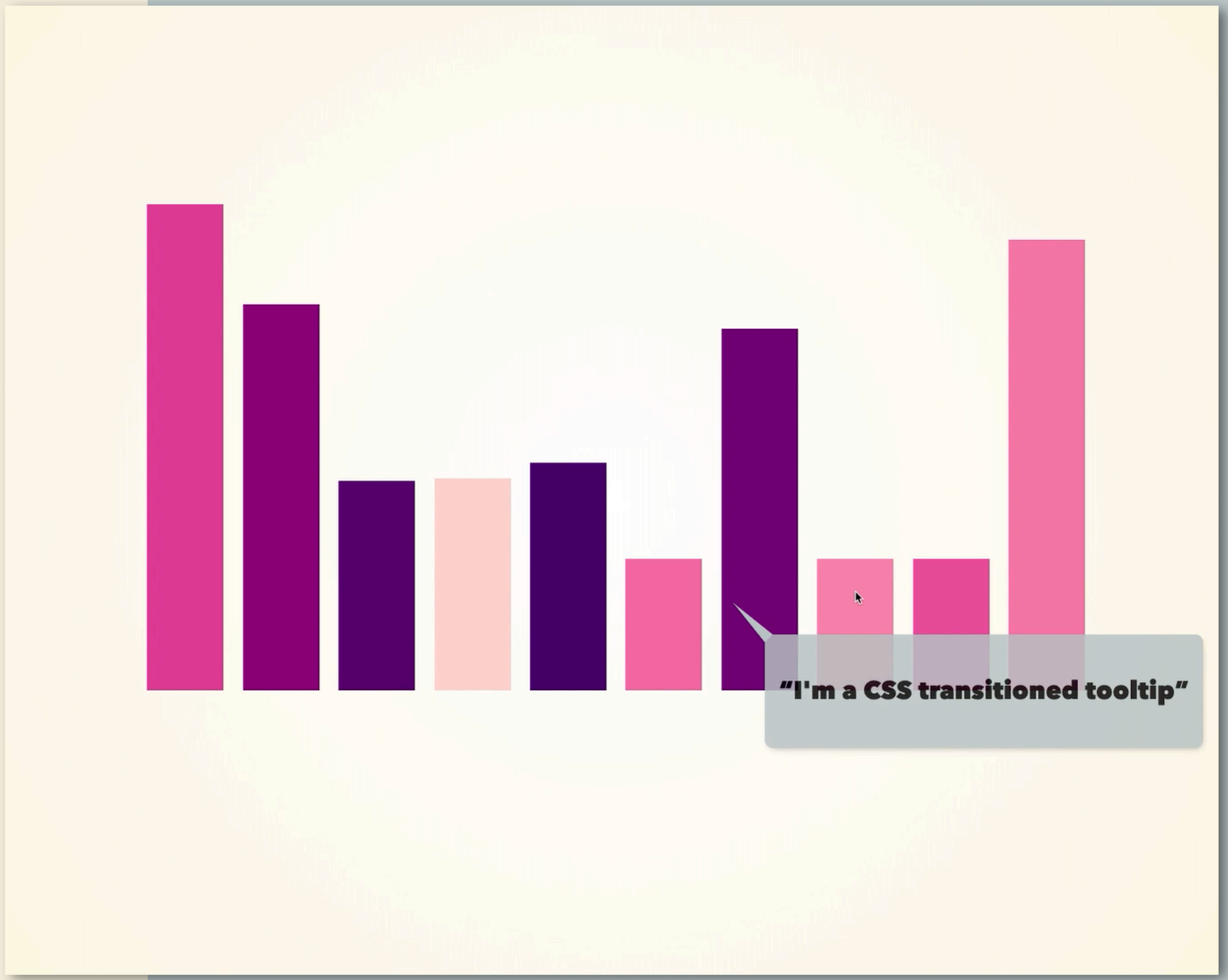


```
const App = () => {
    ...
  return (
    <div className="App">
      <div className="chart-wrapper">
        {data.map((d, i) => (
          <div
            key={d.id}
            className="bar"
            onMouseOver={onMouseMove}
            style={{
              left: `${xScale(i)}px`,
              height: `${yScale(d.height)}px`,
              width: `${xScale.bandwidth()}px`,
              backgroundColor: `${colorScale(d.value)}`
            }}
          />
        ))}
      </div>
      <Tooltip
        origin={origin}
        generateContent={generateContent}
        isVisible={isVisible}
        options={options}
      />
    </div>
)}
```

```
const Tooltip = () => {
...
  return (
      <div
        className={"Tooltip"}
        style={{
          left: `${tooltipDims[X]}px`,
          top: `${tooltipDims[Y]}px`,
        }}
      >
        <div className="content-wrapper">
          {generateContent()}
        </div>
        <svg className="tooltip-tail">
          <path/>
        </svg>
      </div>
    );
}
```

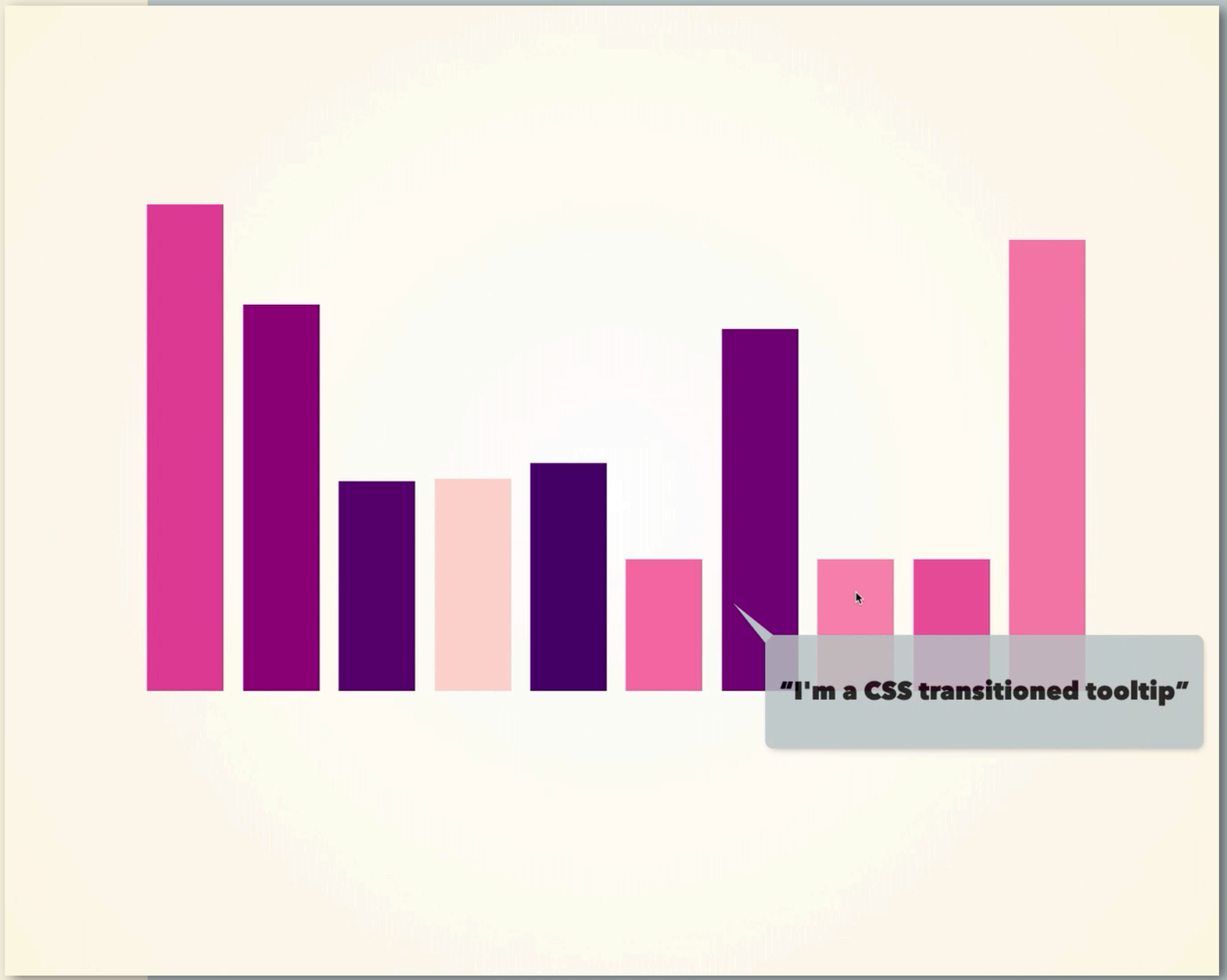
adding delay

tooltip moves to new location but
transitions there gradually


"I'm a CSS transitioned tooltip"

```
const Tooltip = () => {
...
  return (
      <div
        className={"Tooltip"}
        style={{
          left: `${tooltipDims[X]}px`,
          top: `${tooltipDims[Y]}px`,
          transition: `top ${duration}ms , left ${duration}ms`
        }}
      >
        <div className="content-wrapper">
          {generateContent()}
        </div>
        <svg className="tooltip-tail">
          <path/>
        </svg>
      </div>
    );
}
```

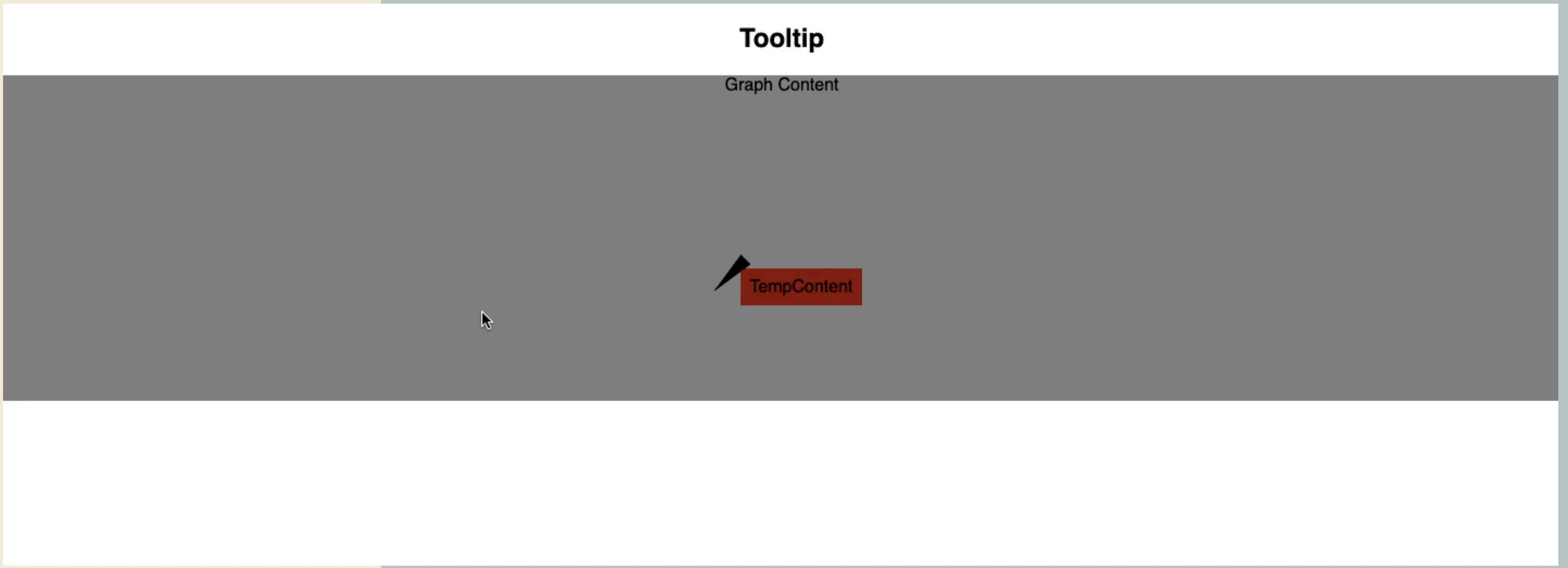
```
const Tooltip = () => {
...
  return (
      <div
        className={"Tooltip"}
        style={{
          left: `${tooltipDims[X]}px`,
          top: `${tooltipDims[Y]}px`,
          transition: `top ${duration}ms , left ${duration}ms`
        }}
      >
        <div className="content-wrapper">
          {generateContent()}
        </div>
        <svg className="tooltip-tail">
          <path/>
        </svg>
      </div>
    );
}
```

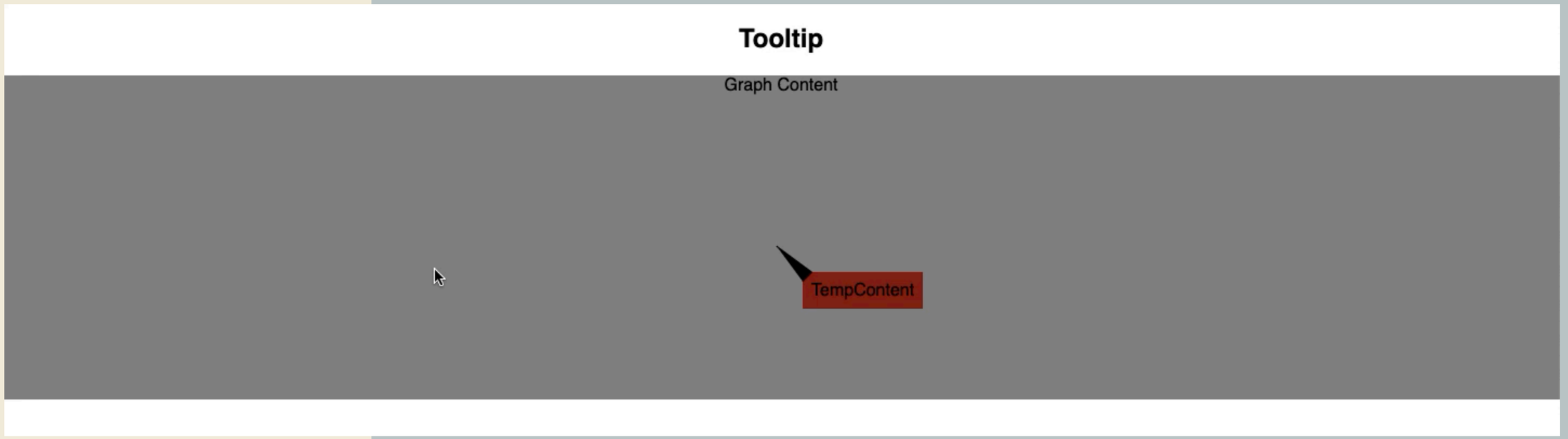

"I'm a CSS transitioned tooltip"

blooper interlude

read: path transitions can be hard!

```
const Tooltip = ({origin}) => {

 useEffect(() => {
   // Maths
   const prevPath = calculatePath(prevOrigin)
   const path = calculatePath(origin)

    d3.select(pathRef)
        .attr("d", prevPath)
        .transition()
        .attr("d", path)
  }, [origin])

 return (
    <div className={"Tooltip"}>
      <div className="content-wrapper">
        {generateContent()}
      </div>
      <svg className="tooltip-tail">
        <path ref={pathRef}/>
      </svg>
    </div>
  );}
```


Tooltip
Graph Content
TempContent


Tooltip
Graph Content
TempContent

final tooltip

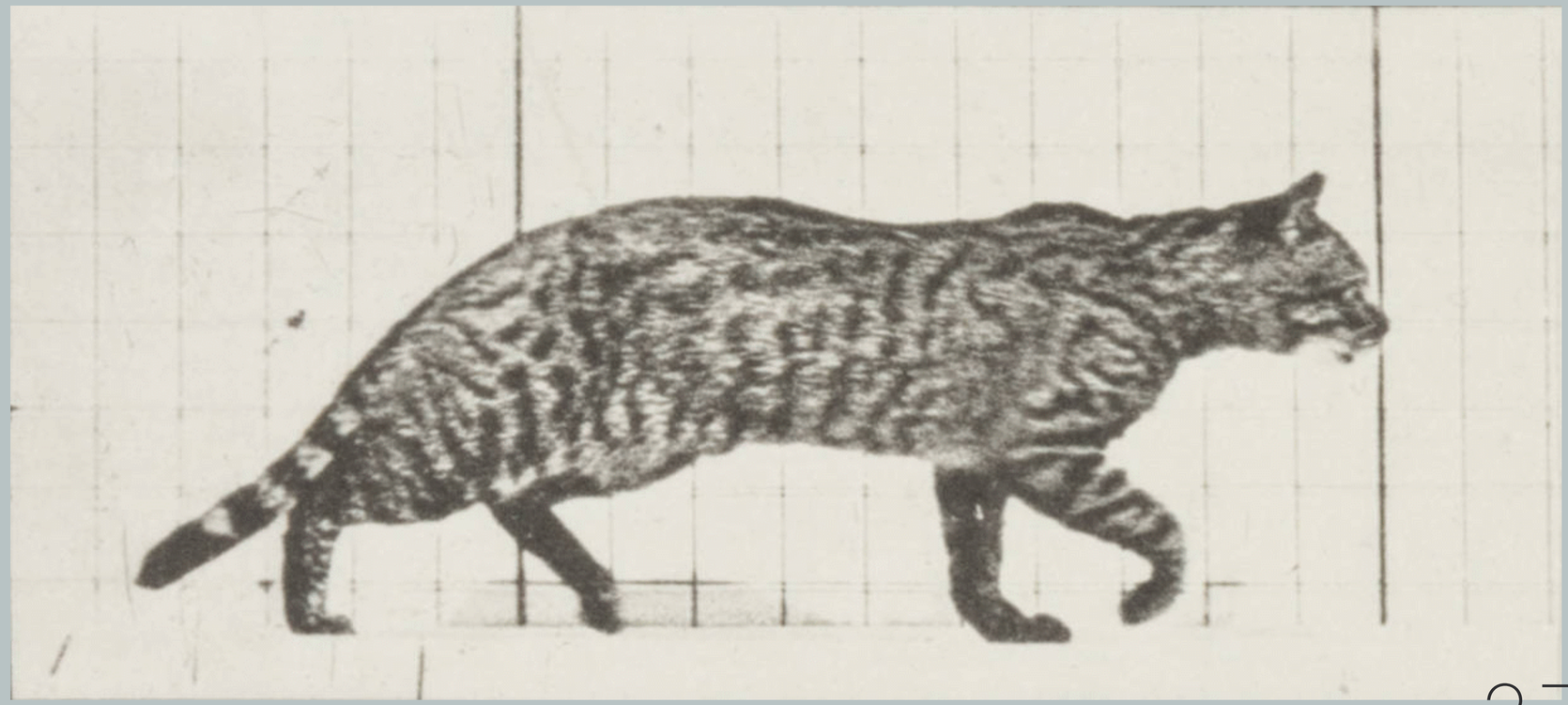
css moves tooltip container, path is recalculated on each frame with RequestAnimationFrame

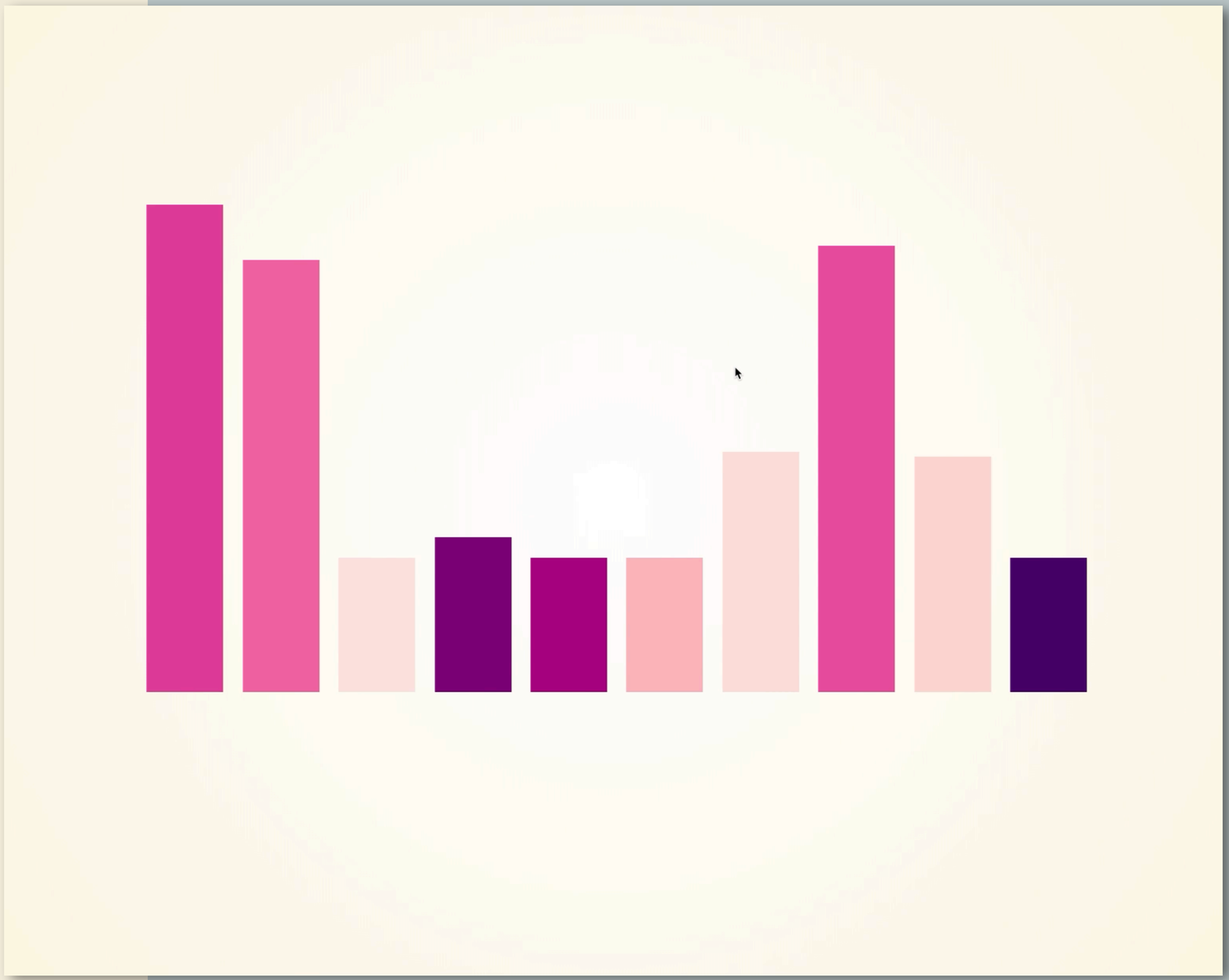

"I'm the final tooltip"

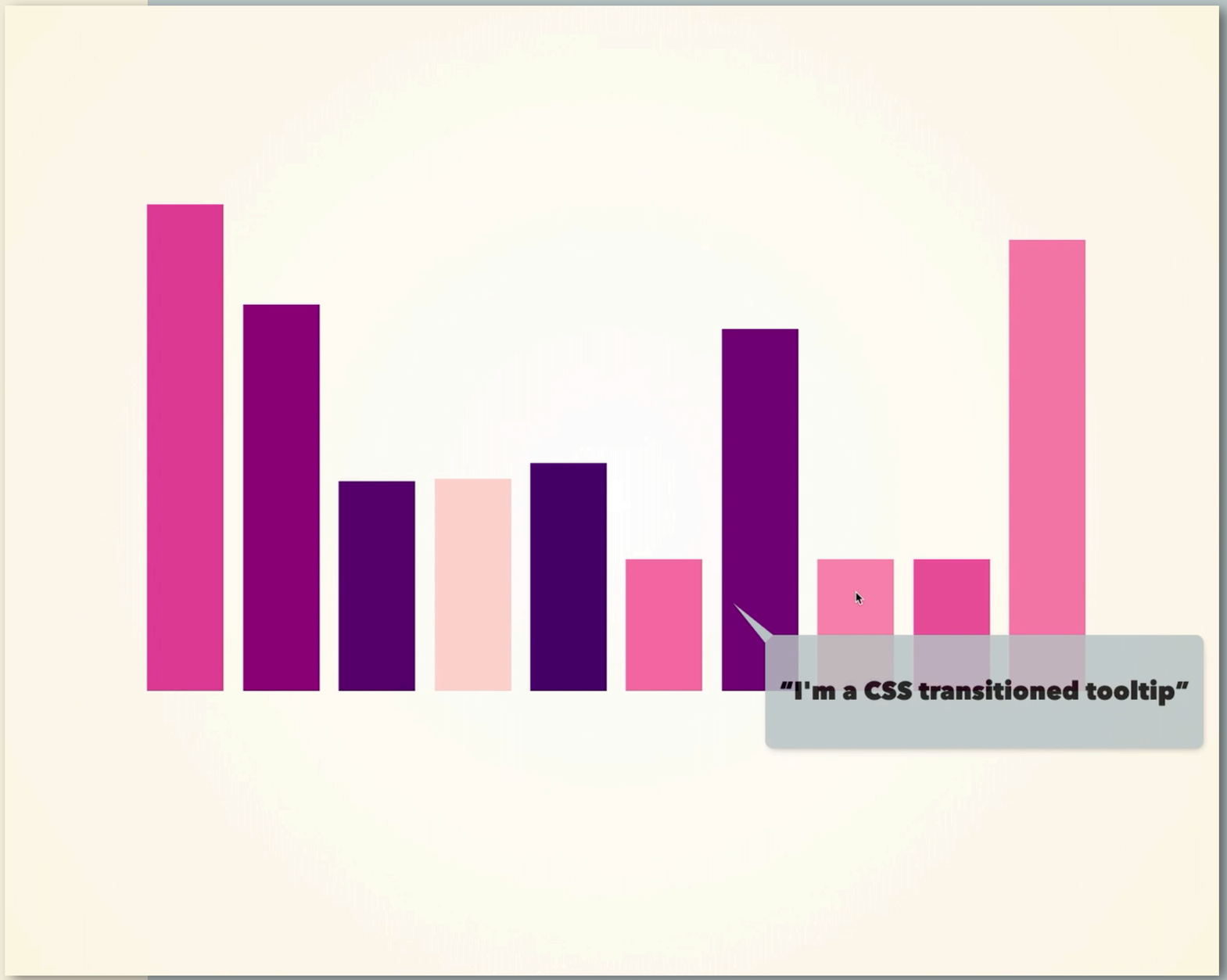
```
const Tooltip = ({ origin }) => {
  ...
  useEffect(() => {

    const animate = () => {
      // Maths, updates path
      const isDone = updatePath()

      if (!isDone) {
        return window.requestAnimationFrame(animate)
      }
    }

    animate()
  }, [origin])

  return (
    <div className={'Tooltip'}>
      <div className="content-wrapper">{generateContent()}</div>
      <svg className="tooltip-tail">
        <path ref={pathRef} />
      </svg>
    </div>
  )
}

```

```jsx
const Tooltip = ({ origin }) => {
  ...
  useEffect(() => {

    const animate = () => {
      // Maths, updates path
      const isDone = updatePath()

      if (!isDone) {
        return window.requestAnimationFrame(animate)
      }
    }

    animate()
  }, [origin])

  return (
    <div className={'Tooltip'}>
      <div className="content-wrapper">{generateContent()}</div>
      <svg className="tooltip-tail">
        <path ref={pathRef} />
      </svg>
    </div>
  )
}

```

how this works






"I'm a CSS transitioned tooltip"


"I'm the final tooltip"

building blocks

Interpolation

specifying what is changing and how we get from point A to point B.

Duration and Delay

deciding how long an animation should take and when should it start.

Timing and Orchestration

exploring easing functions, as well as how the animation should unfold (staggering).

Object Lifecycle

elements should behave differently depending on state relative to the app.

our toolkit

CSS Transitions

by far our most frequently used!

React (CSS) Transition Group

helpful for orchestrating entering and exiting elements.

D3.js

great utility function, easing functions, element lifecycle control (enter, update, exit).

RequestAnimationFrame

when we need frame x frame control over our animation.

special mention: Tween.js, React-Spring

questions to think about

when designing data visualizations transitions

What is Point A and B?

What are the points you are transitioning between and what is their semantic connection?

What Happens if Interrupted?

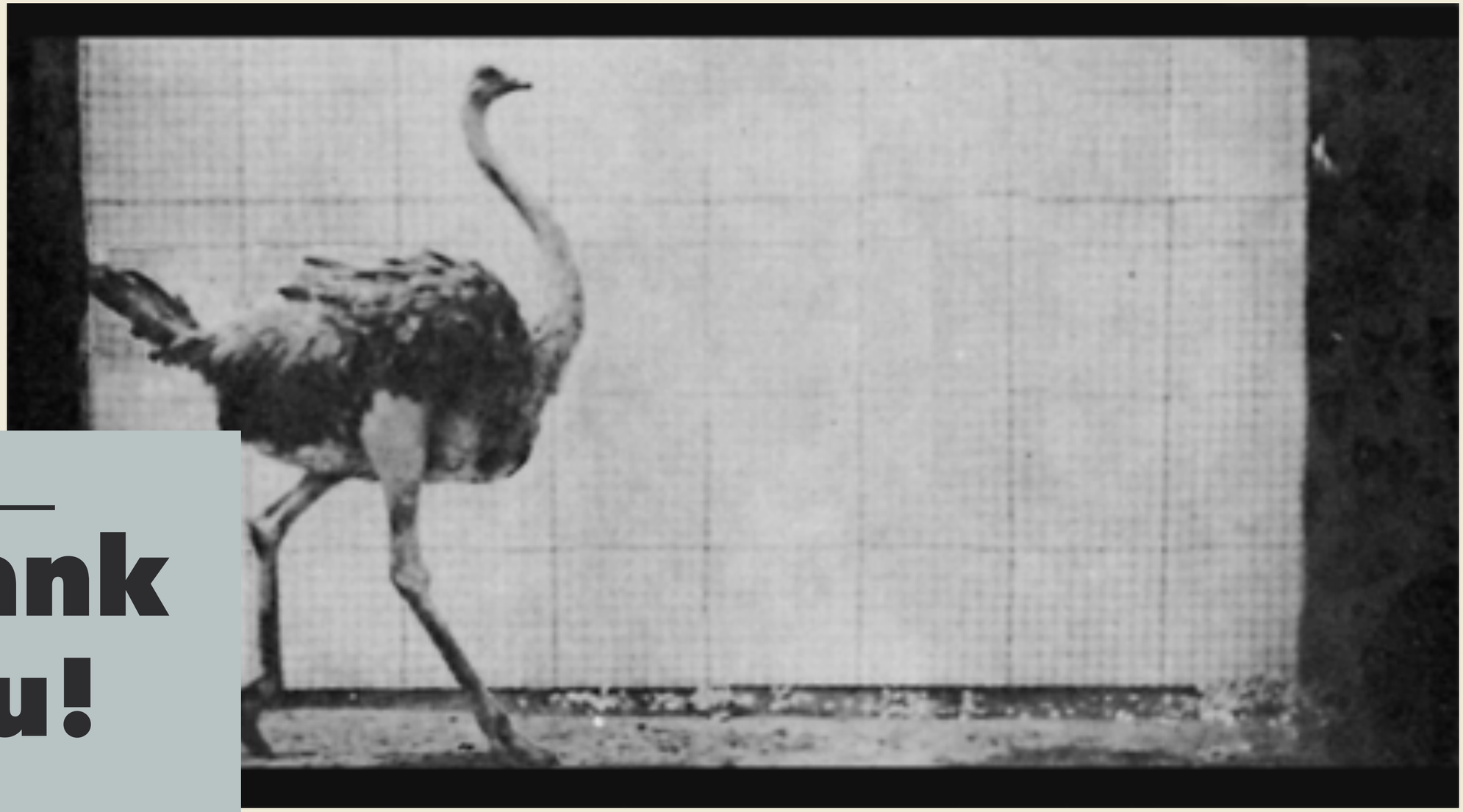
What is the user expectation, should the transition jump to the end? Return to the beginning? Tween to the new position?

What is the Element Lifecycle?

How do these elements behave and change as they enter, exit, and update?

Can This Delight and Engage?

How can you enrich a viewer's world? Convey more meaning? Draw people in, in a surprising and satisfying way?



thank you!

auchers
aucher.serr@gmail.com