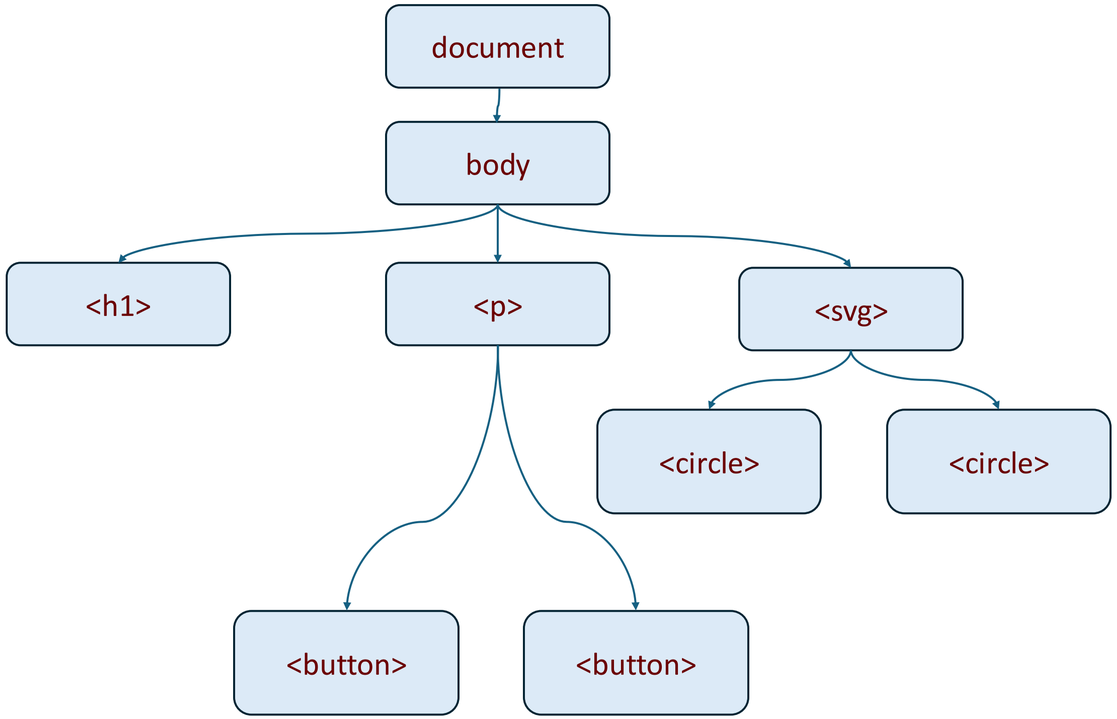
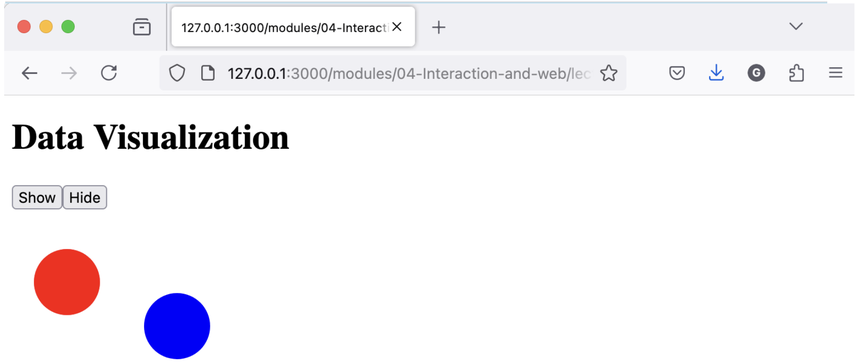
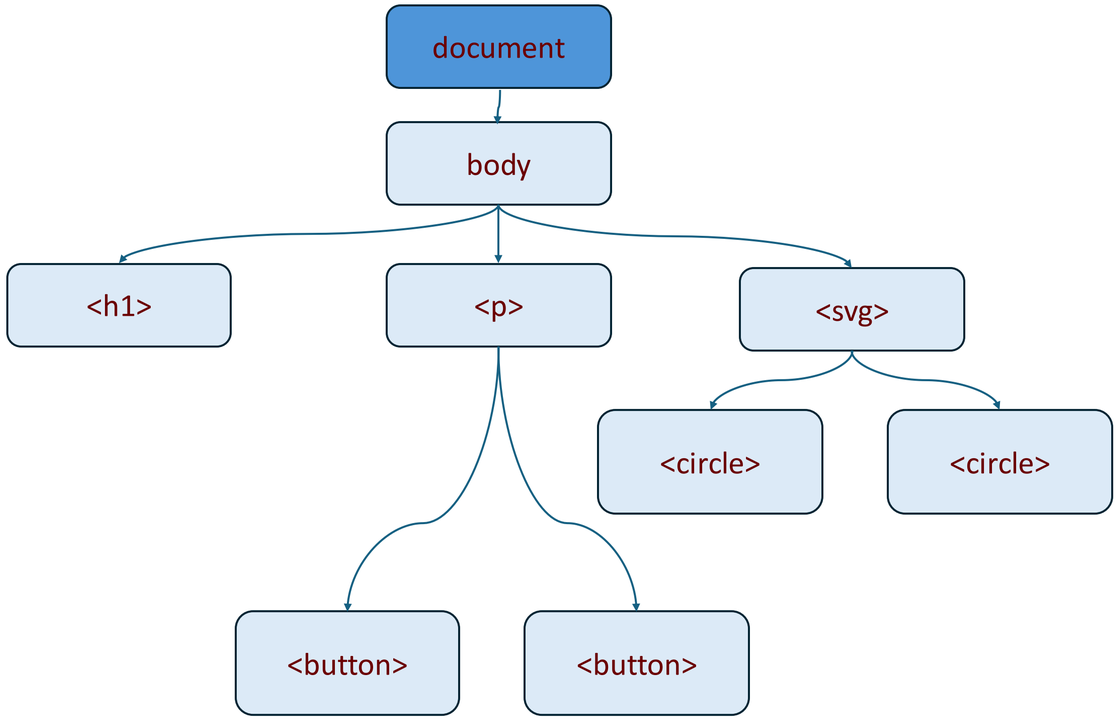
Dynamic visualization with Javascript

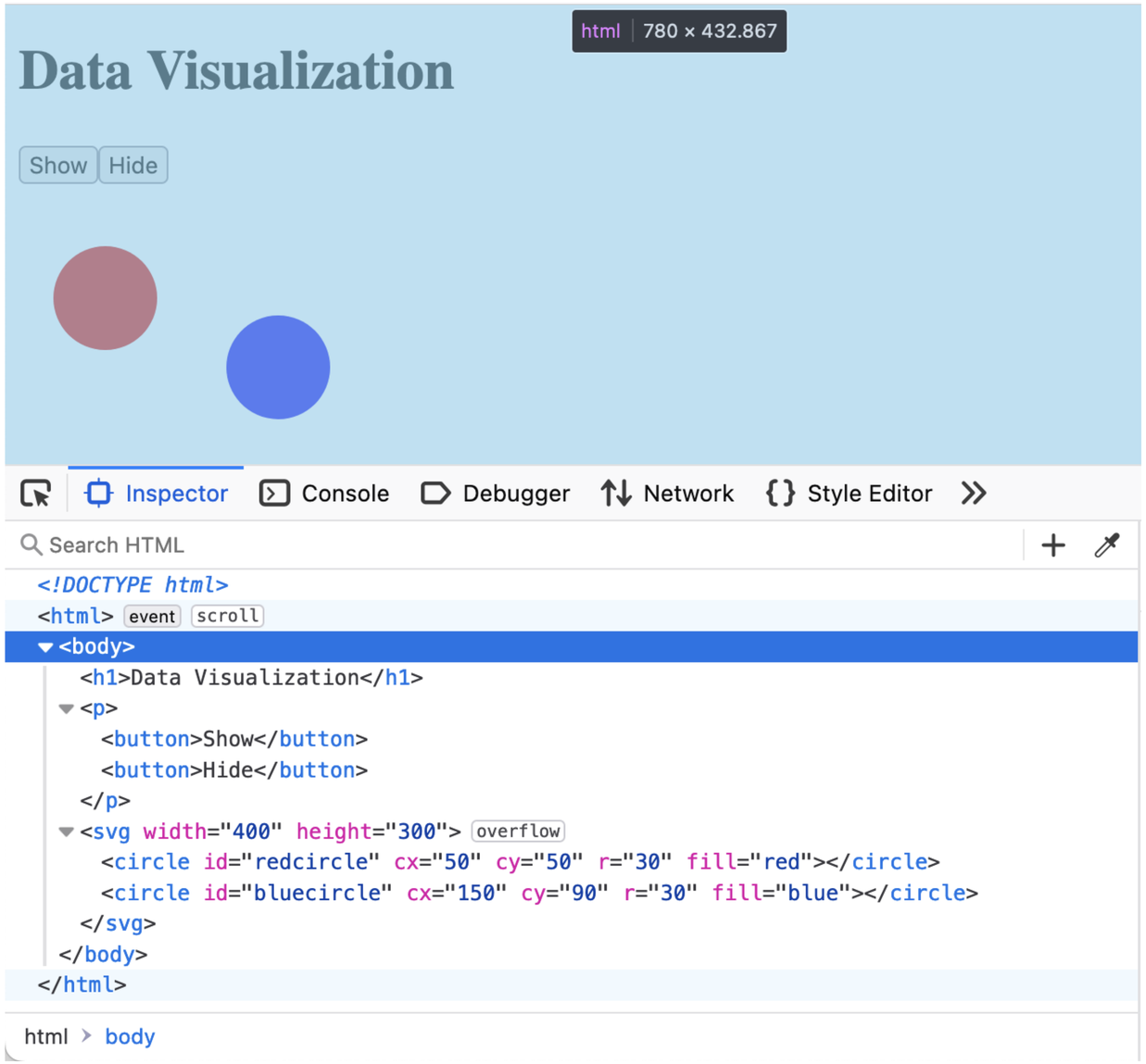
```
<!DOCTYPE html>
<html>
<body>
  <h1>Data Visualization</h1>
  <p>
    <button>Show</button>
    <button>Hide</button>
  </p>
  <svg width="400" height="300">
    <circle id="redcircle" cx="50" cy="50" r="30" fill="red"></circle>
    <circle id="bluecircle" cx="150" cy="90" r="30" fill="blue"></circle>
  </svg>
</body>
</html>
```





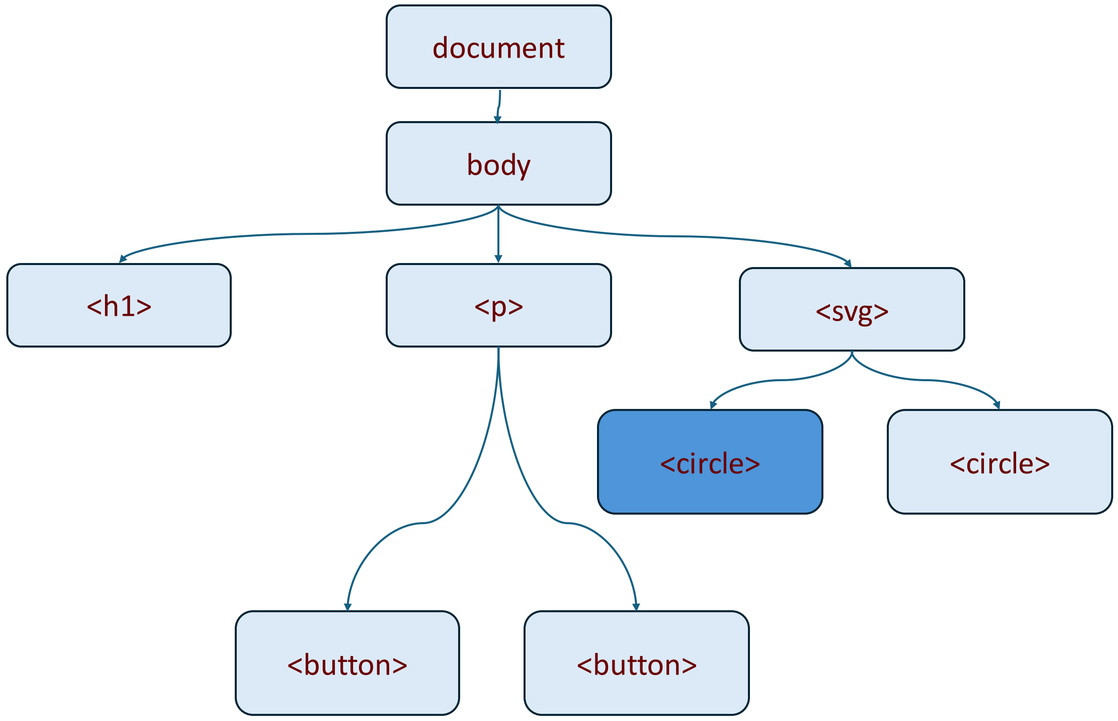

document
body
<h1>

<svg>
<circle>
<circle>
<button>
<button>


html | 780 × 432.867
Data Visualization
Show
Hide
Inspector
Console
Debugger
Network
Style Editor
Search HTML
<!DOCTYPE html>
<html> event scroll
<body>
<h1>Data Visualization</h1>

<button>Show</button>
<button>Hide</button>

<svg width="400" height="300"> overflow
<circle id="redcircle" cx="50" cy="50" r="30" fill="red"></circle>
<circle id="bluecircle" cx="150" cy="90" r="30" fill="blue"></circle>
</svg>
</body>
</html>
html > body



Data Visualization

Show Hide

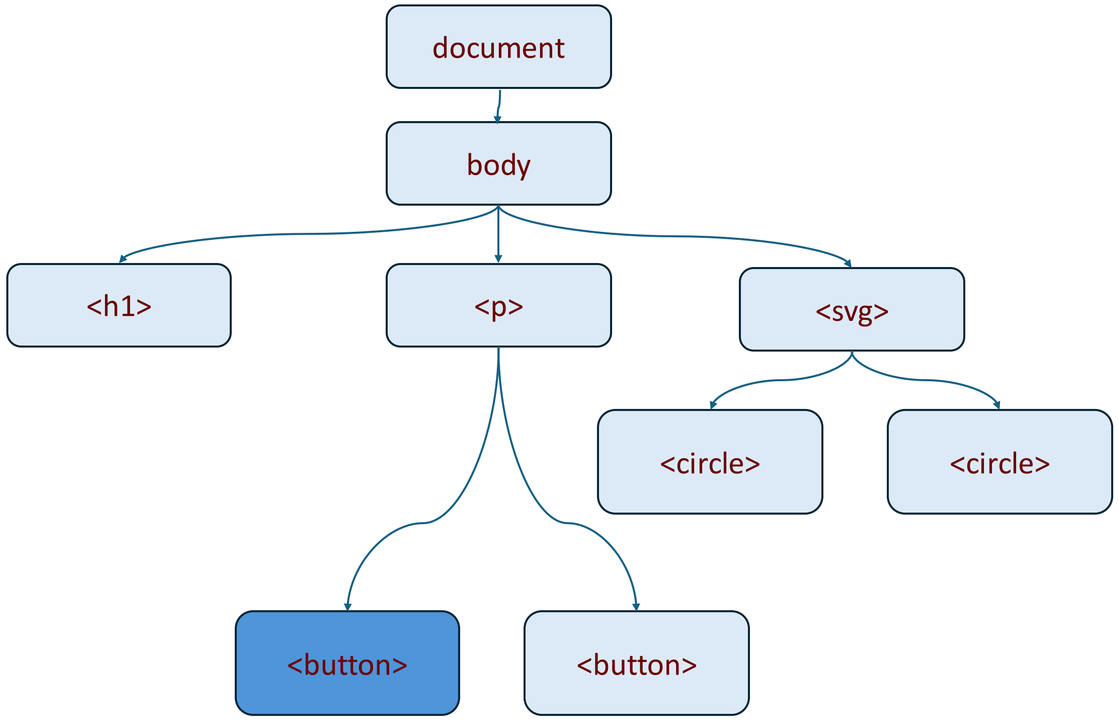
circle#redcircle 60 × 60

Inspector Console Debugger Network Style Editor

Search HTML

```
<!DOCTYPE html>
<html> event scroll
 <body>
   <h1>Data Visualization</h1>
  <p>
    <button>Show</button>
    <button>Hide</button>
   </p>
  <svg width="400" height="300"> overflow
    <circle id="redcircle" cx="50" cy="50" r="30" fill="red"></circle>
    <circle id="bluecircle" cx="150" cy="90" r="30" fill="blue"></circle>
   </svg>
 </body>
</html>
```

html › body › svg › circle#redcircle



Data Visualization

button 45.9 × 22

Show Hide

Inspector Console Debugger Network Style Editor

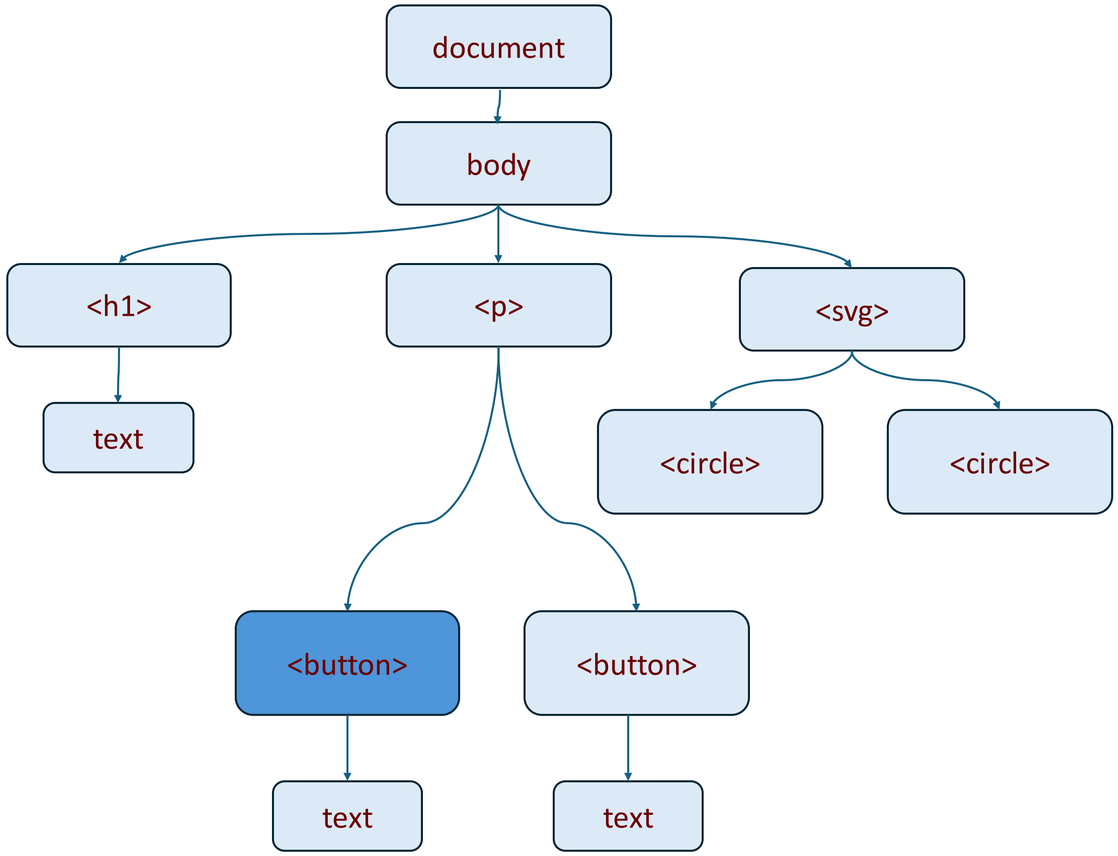
Search HTML

```
<!DOCTYPE html>
<html> event scroll
  <body>
    <h1>Data Visualization</h1>
    <p>
      <button>Show</button>
      <button>Hide</button>
    </p>
    <svg width="400" height="300"> overflow
      <circle id="redcircle" cx="50" cy="50" r="30" fill="red"></circle>
      <circle id="bluecircle" cx="150" cy="90" r="30" fill="blue"></circle>
    </svg>
  </body>
</html>
```

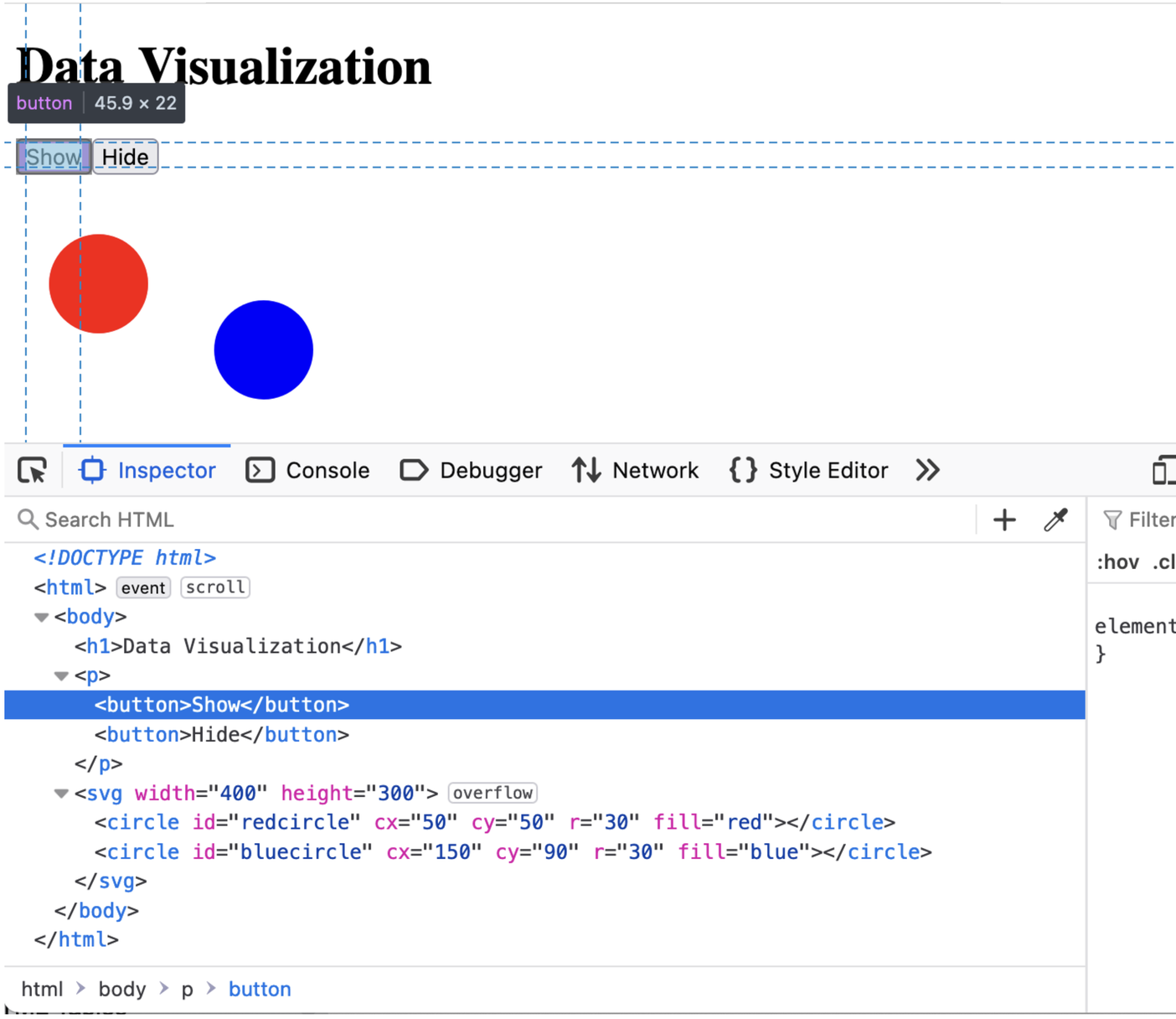
html › body › p › button

:hov .cl

element
}

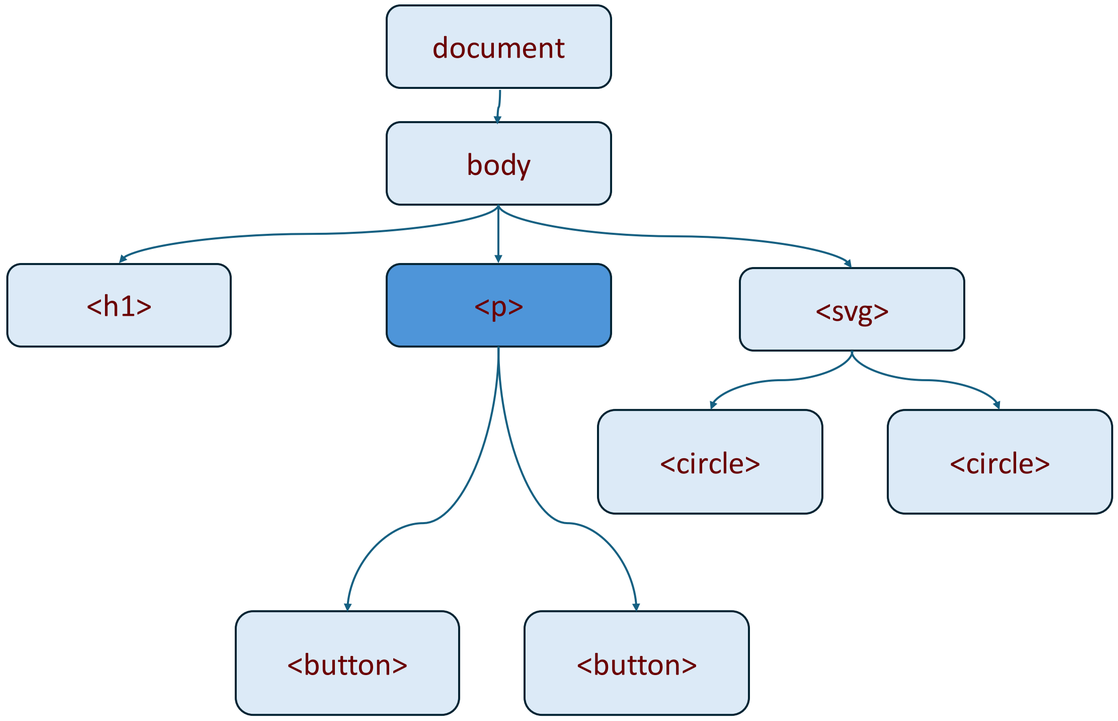

document
body
<h1>

<svg>
text
<circle>
<circle>
<button>
<button>
text
text


Data Visualization
button 45.9 × 22
Show
Hide
Inspector
Console
Debugger
Network
Style Editor
Search HTML
<!DOCTYPE html>
<html> event scroll
<body>
<h1>Data Visualization</h1>

<button>Show</button>
<button>Hide</button>

<svg width="400" height="300"> overflow
<circle id="redcircle" cx="50" cy="50" r="30" fill="red"></circle>
<circle id="bluecircle" cx="150" cy="90" r="30" fill="blue"></circle>
</svg>
</body>
</html>
html › body › p › button



Data Visualization

p | 764 × 22

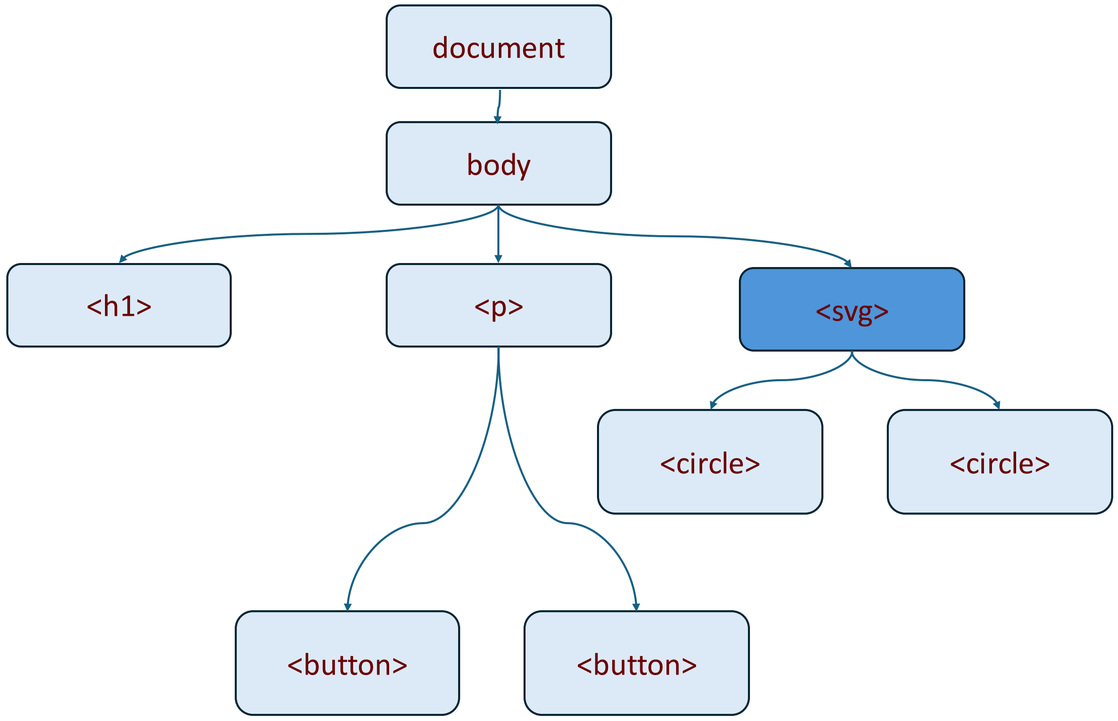
Show Hide

Inspector Console Debugger Network Style Editor

Search HTML

```
<!DOCTYPE html>
<html> event scroll
<body>
  <h1>Data Visualization</h1>
  <p>
    <button>Show</button>
    <button>Hide</button>
  </p>
  <svg width="400" height="300"> overflow
    <circle id="redcircle" cx="50" cy="50" r="30" fill="red"></circle>
    <circle id="bluecircle" cx="150" cy="90" r="30" fill="blue"></circle>
  </svg>
</body>
</html>
```

html › body › p



Data Visualization

Show Hide

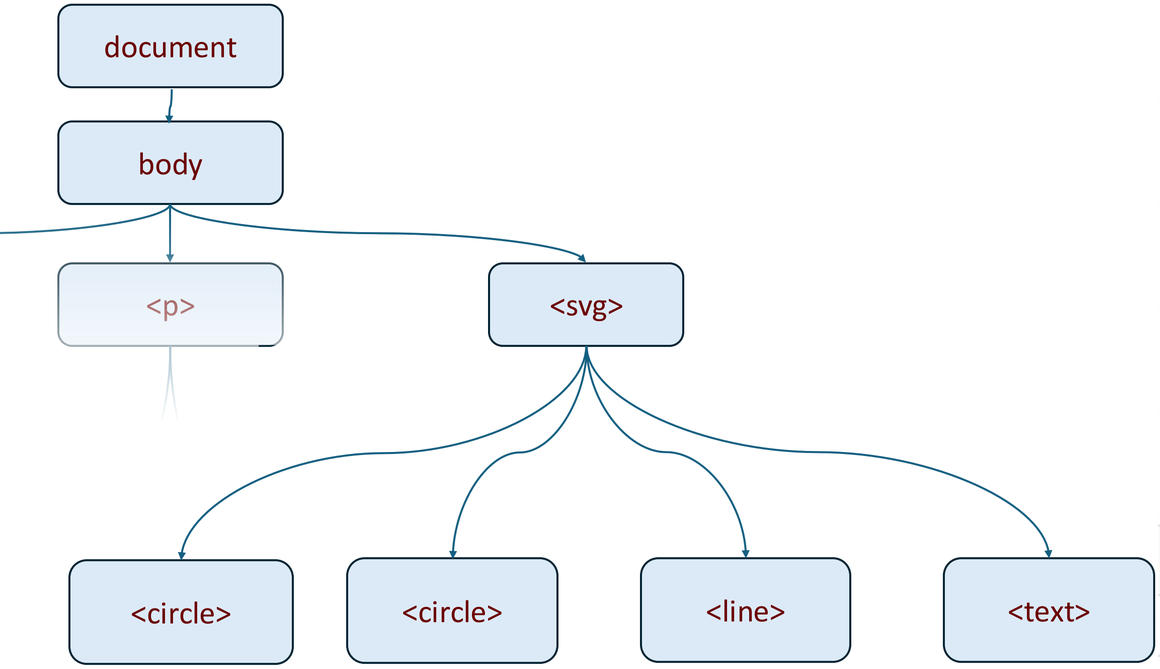
svg | 400 × 300

Inspector Console Debugger Network Style Editor

Search HTML

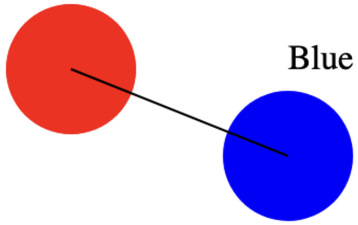
```
<!DOCTYPE html>
<html> event scroll
 <body>
   <h1>Data Visualization</h1>
  <p>
    <button>Show</button>
    <button>Hide</button>
   </p>
  <svg width="400" height="300"> overflow
    <circle id="redcircle" cx="50" cy="50" r="30" fill="red"></circle>
    <circle id="bluecircle" cx="150" cy="90" r="30" fill="blue"></circle>
   </svg>
 </body>
</html>
```

html › body › svg



Data Visualization

Show Hide

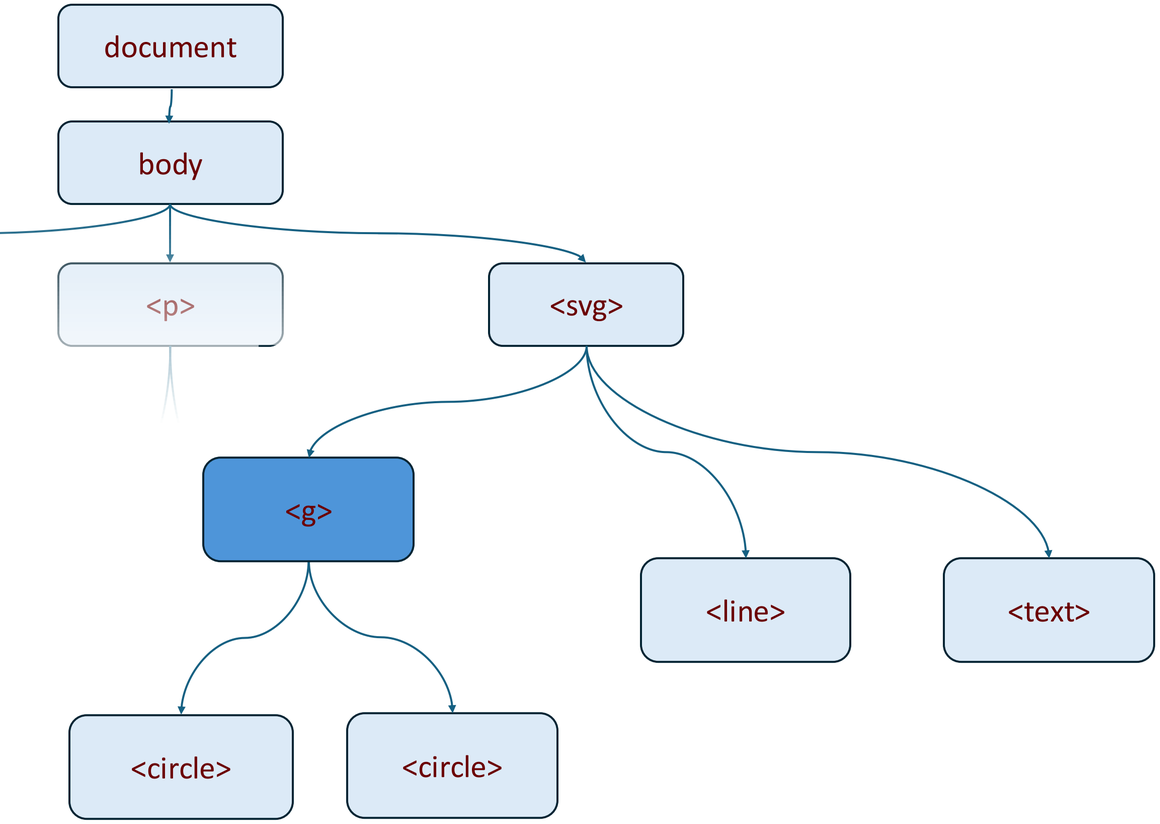


Inspector Console Debugger Network Style Editor

Search HTML

```
<!DOCTYPE html>
<html> event scroll
  <head></head>
  <body>
    <script type="text/javascript" src="/___vscode_livepreview_injected_script"></
    script>
    <h1>Data Visualization</h1>
    <p> ··· </p>
    <svg width="400" height="300"> overflow
      <circle id="redcircle" cx="50" cy="50" r="30" fill="red"></circle>
      <circle id="bluecircle" cx="150" cy="90" r="30" fill="blue"></circle>
      <text id="label" x="150" y="50">Blue</text>
      <line x1="50" y1="50" x2="150" y2="90" stroke="black"></line>
    </svg>
```

html › body › svg



Data Visualization

Show Hide

g#circles | 160 × 100

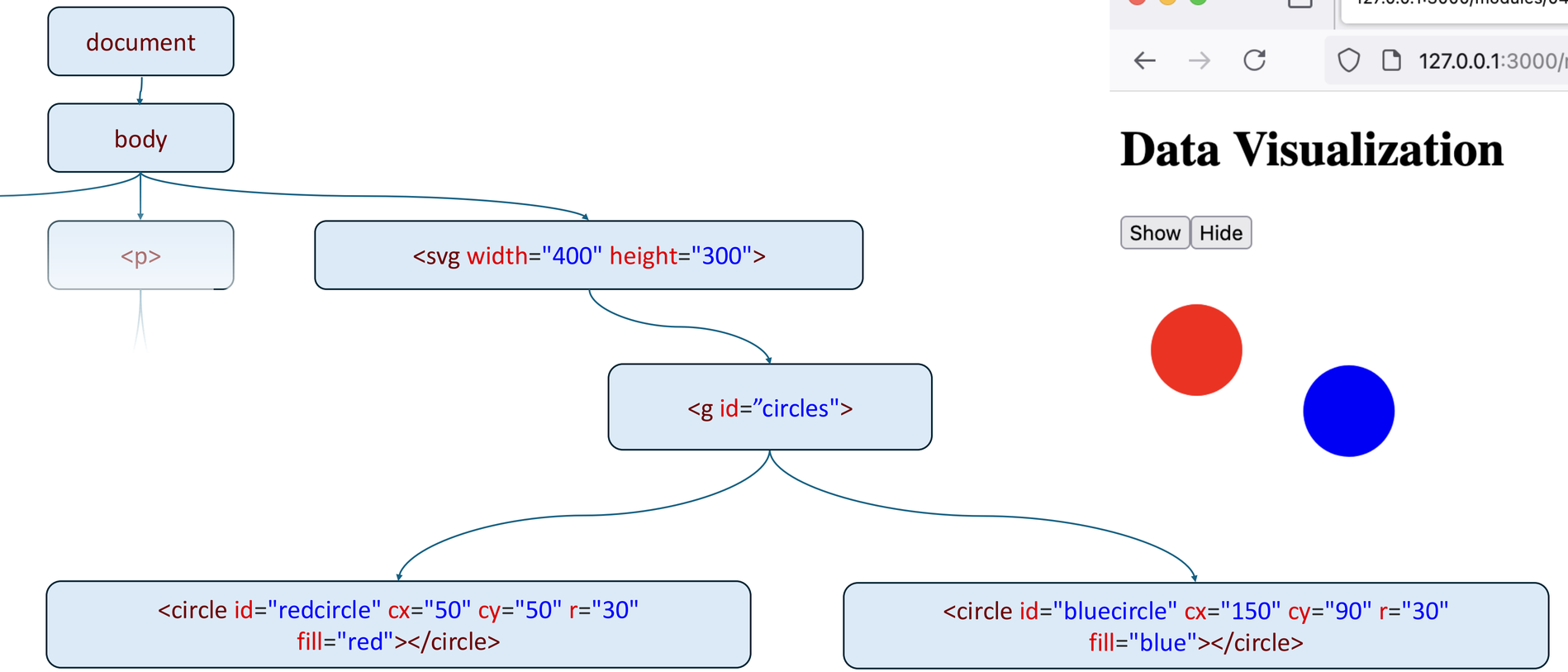
Blue

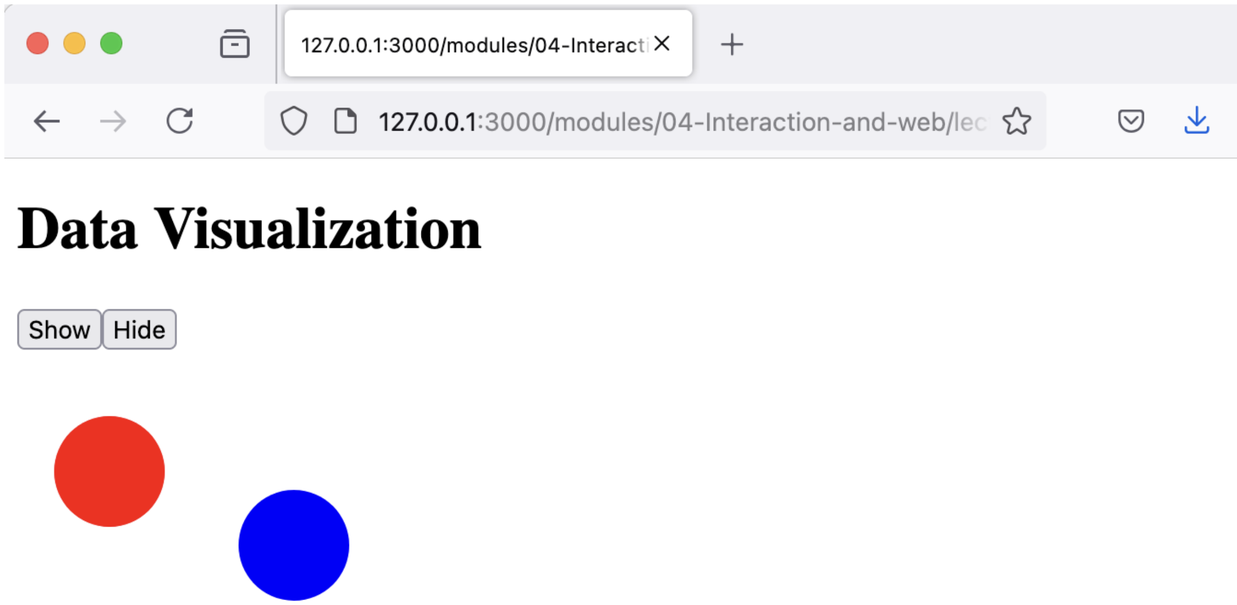
Inspector Console Debugger Network Style Editor

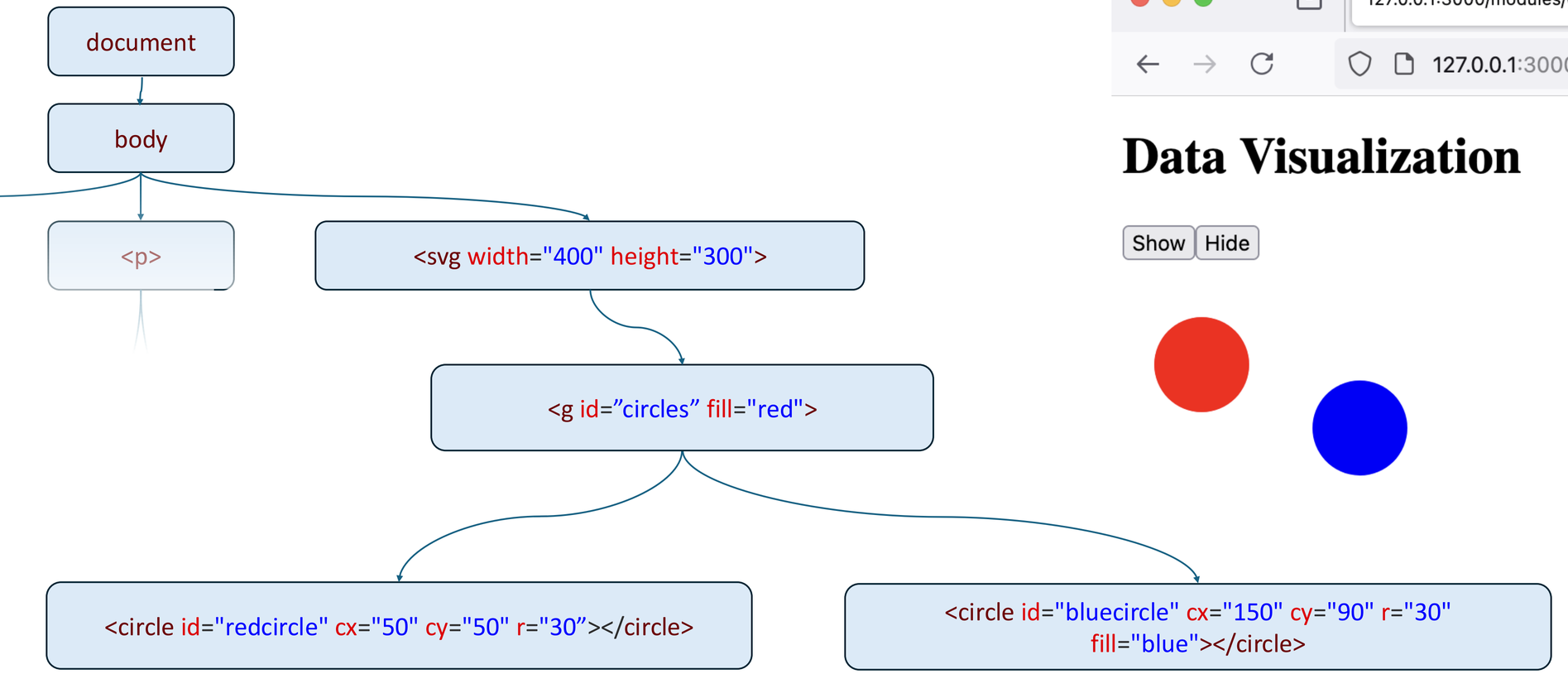
Search HTML

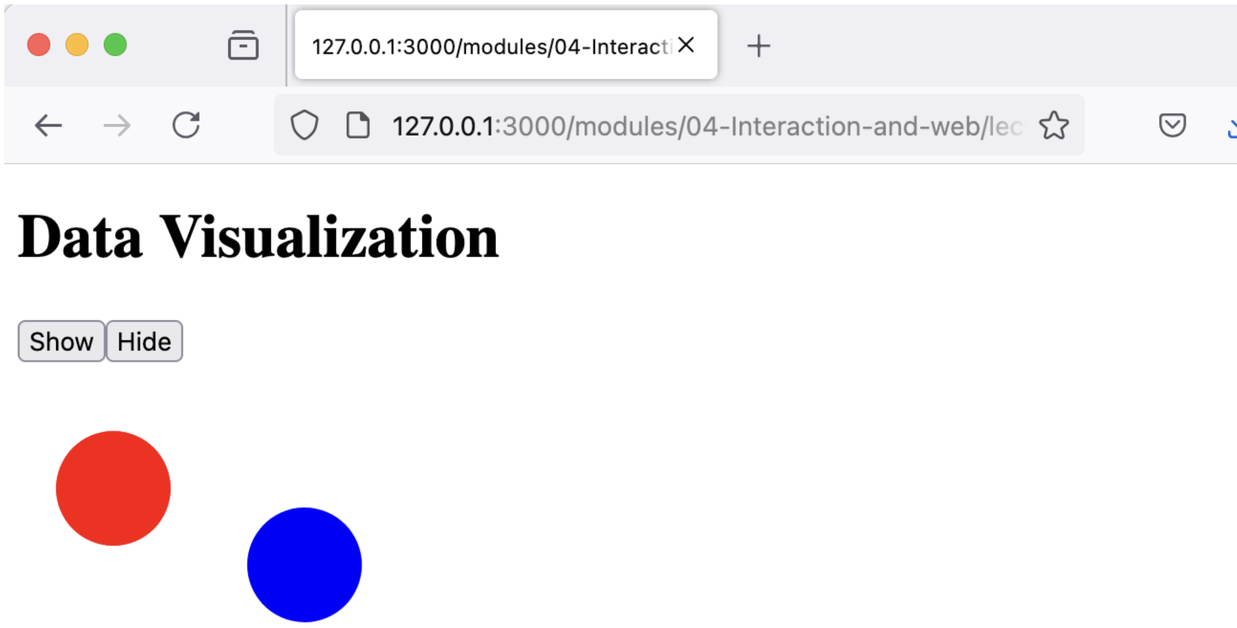
```
<script type="text/javascript" src="/___vscode_livepreview_injected_script"></
script>
<h1>Data Visualization</h1>
<p> ··· </p>
<svg width="400" height="300"> overflow
<g id="circles">
<circle id="redcircle" cx="50" cy="50" r="30" fill="red"></circle>
<circle id="bluecircle" cx="150" cy="90" r="30" fill="blue"></circle>
</g>
<text id="label" x="150" y="50">Blue</text>
<line x1="50" y1="50" x2="150" y2="90" stroke="black"></line>
</svg>
</body>
</html>
```

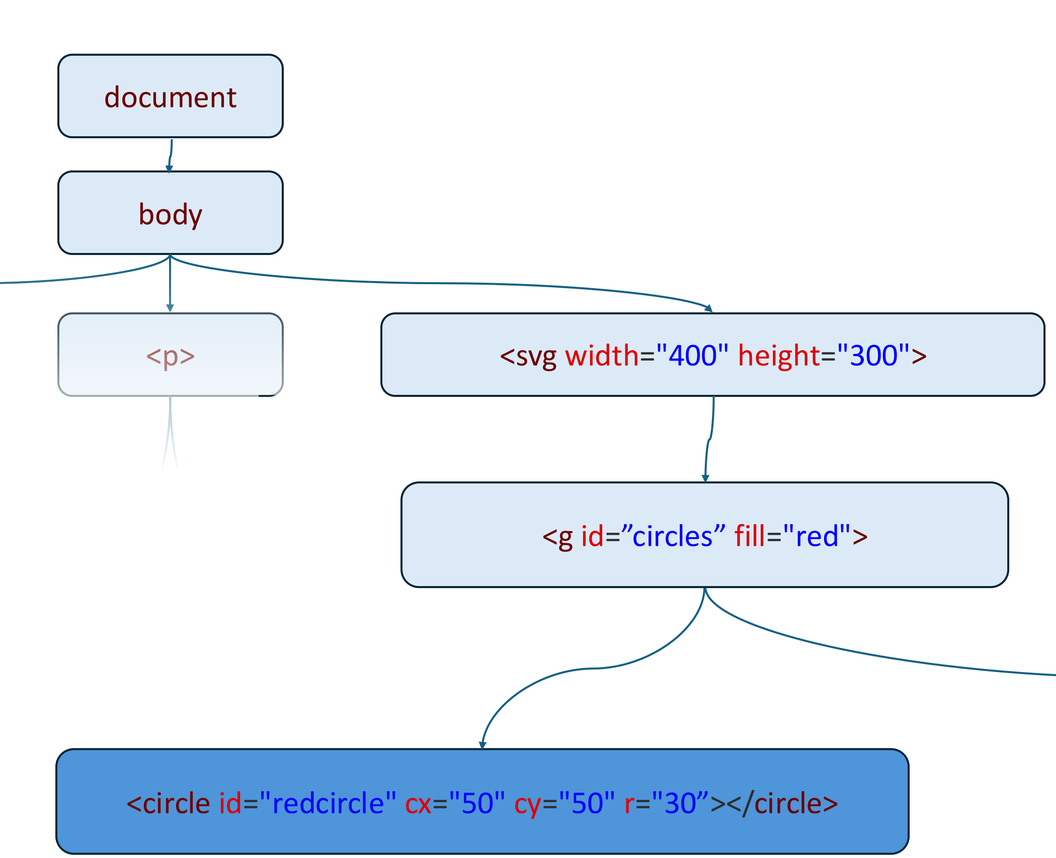
html › body › svg › g#circles

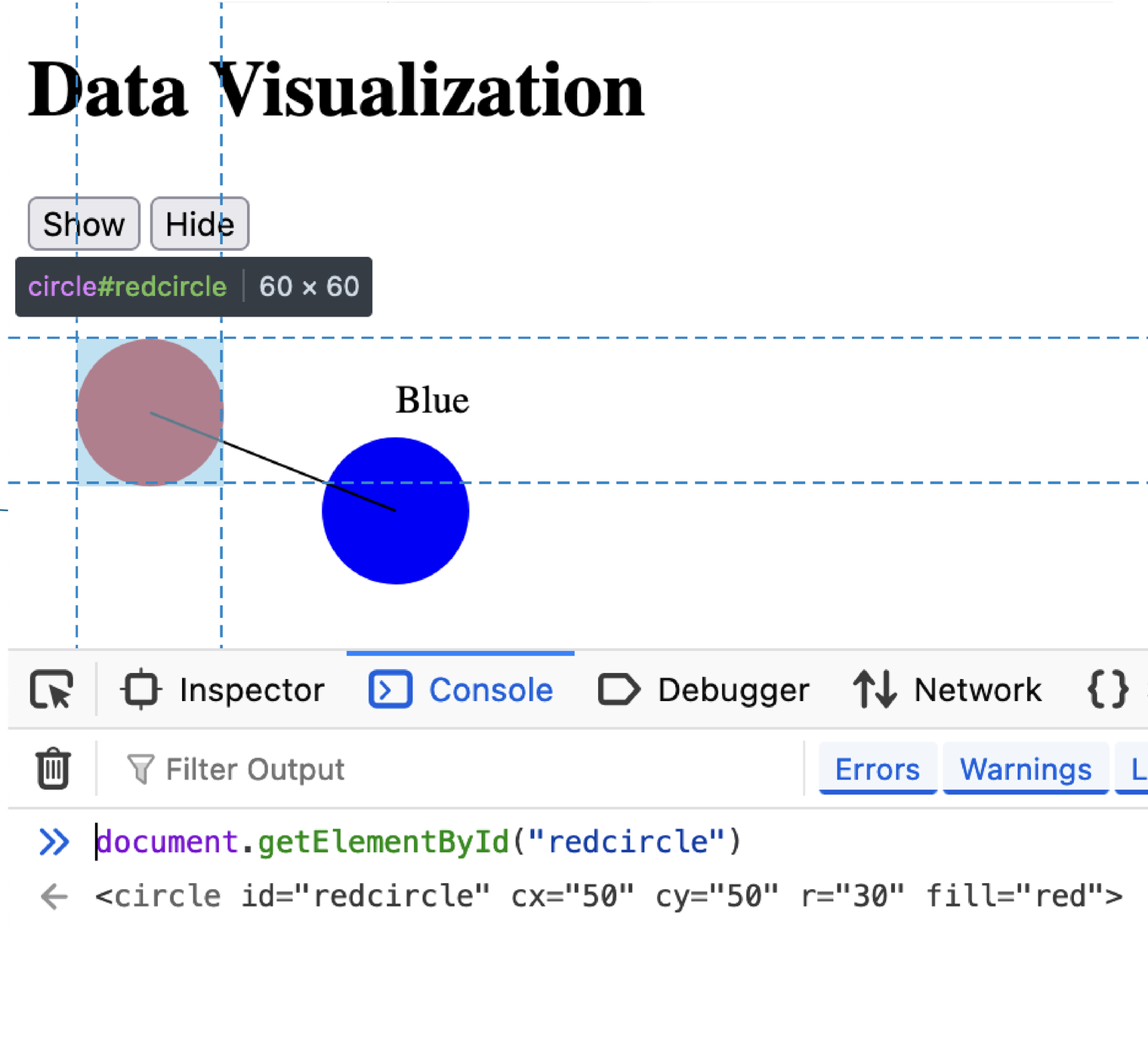

document
body
<p>
<svg width="400" height="300">
<g id=”circles">
<circle id="redcircle" cx="50" cy="50" r="30"
fill="red"></circle>
<circle id="bluecircle" cx="150" cy="90" r="30"
fill="blue"></circle>

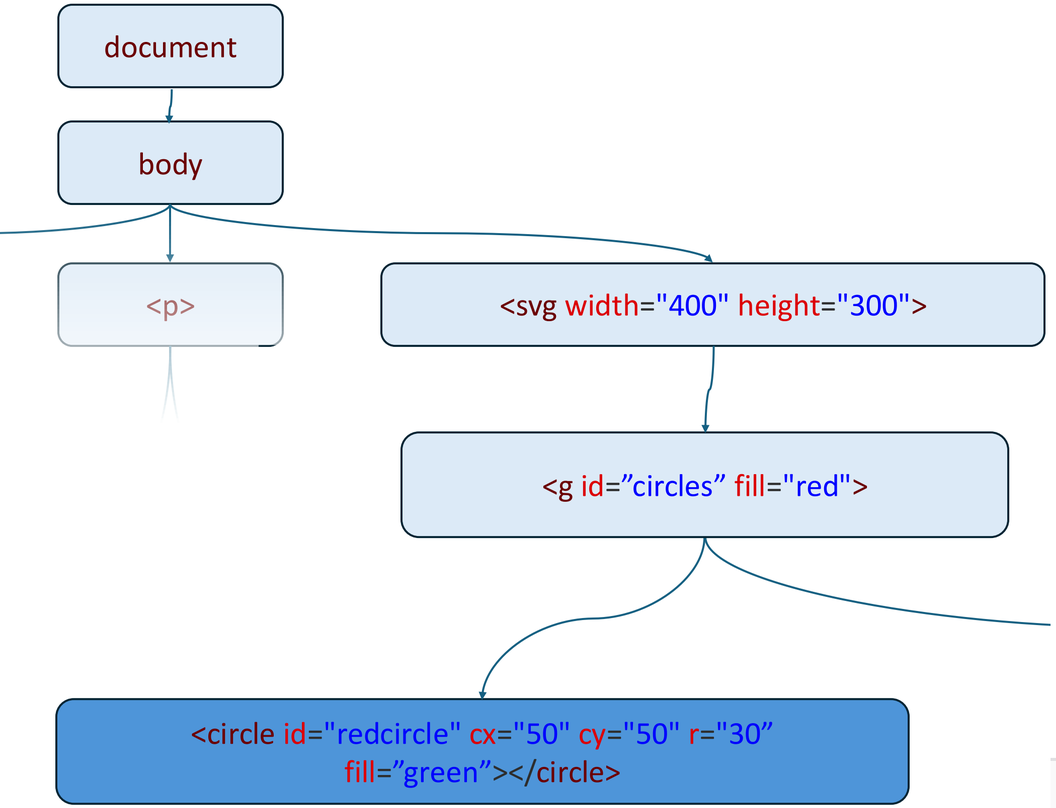

127.0.0.1:3000/modules/04-Interact
127.0.0.1:3000/modules/04-Interaction-and-web/lec
Data Visualization
Show
Hide


document
body
<p>
<svg width="400" height="300">
<g id=”circles” fill="red">
<circle id="redcircle" cx="50" cy="50" r="30”></circle>
<circle id="bluecircle" cx="150" cy="90" r="30"
fill="blue"></circle>


127.0.0.1:3000/modules/04-Interact
127.0.0.1:3000/modules/04-Interaction-and-web/lec
Data Visualization
Show
Hide

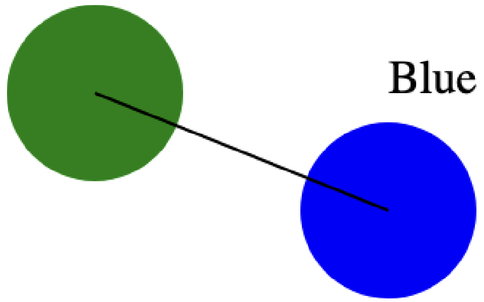

document
body
<p>
<svg width="400" height="300">
<g id=”circles” fill="red">
<circle id="redcircle" cx="50" cy="50" r="30”></circle>

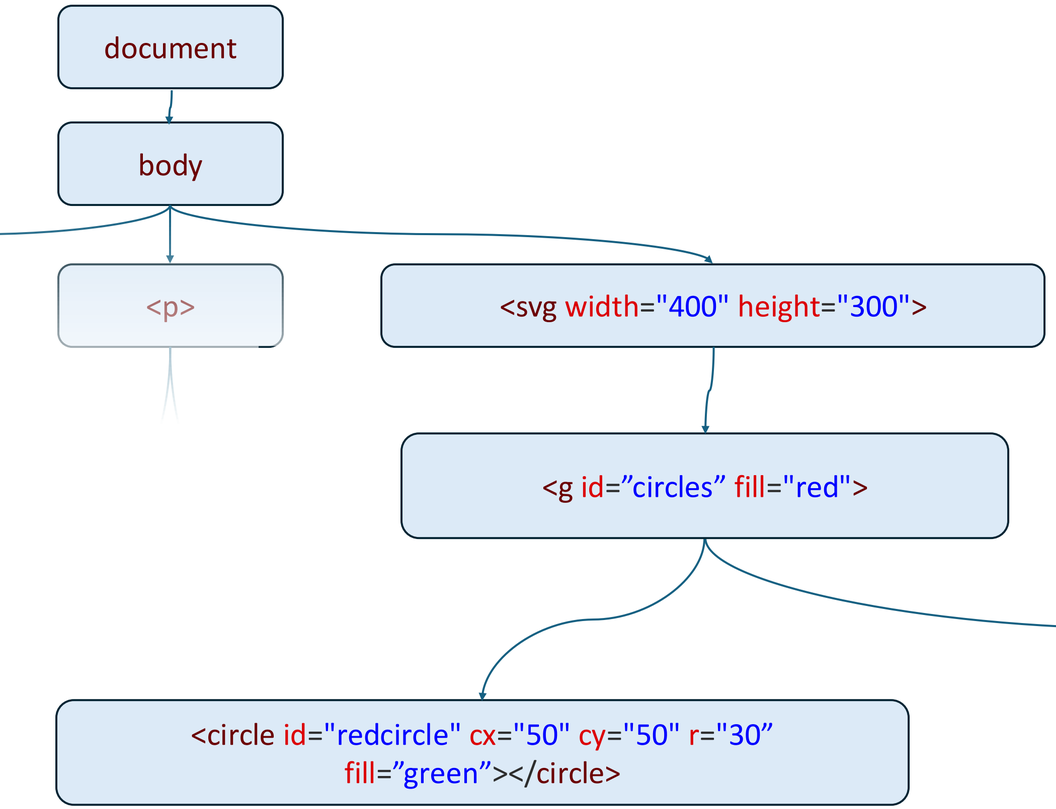

Data Visualization
Show
Hide
circle#redcircle | 60 × 60
Blue
Inspector
Console
Debugger
Network
Filter Output
Errors
Warnings
document.getElementById("redcircle")
<circle id="redcircle" cx="50" cy="50" r="30" fill="red">



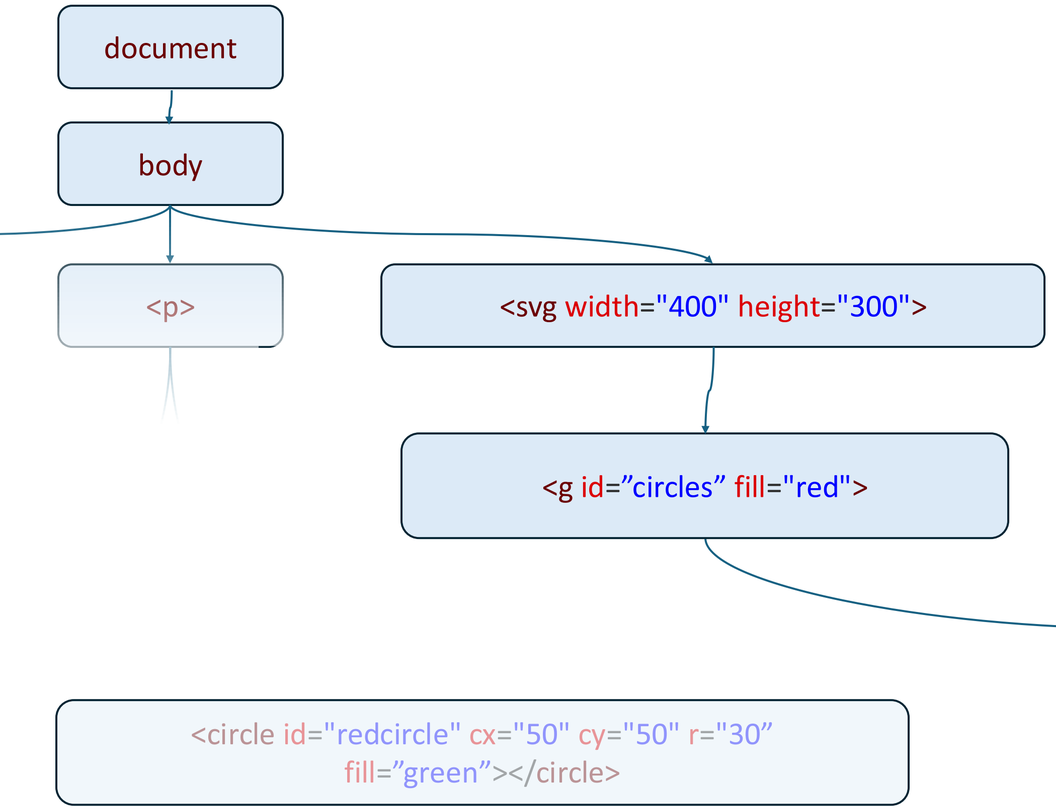
Data Visualization

Show Hide



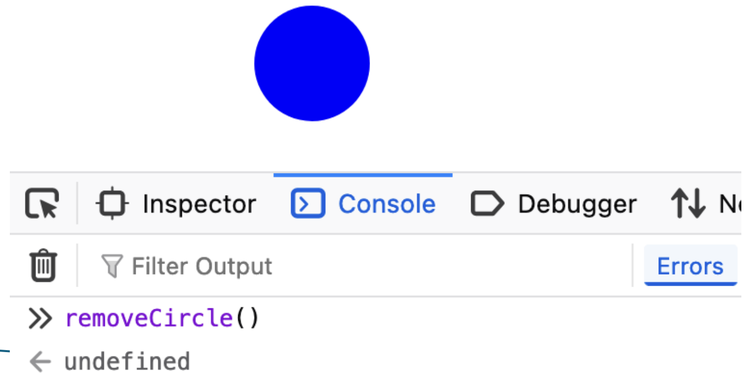



```
let redcircle = document.getElementById('redcircle');
let circles = document. getElementById(circles');

function removeCircle() {
    redcircle.remove()
}

function addCircle() {
    svg.appendChild(redcircle);
}
```



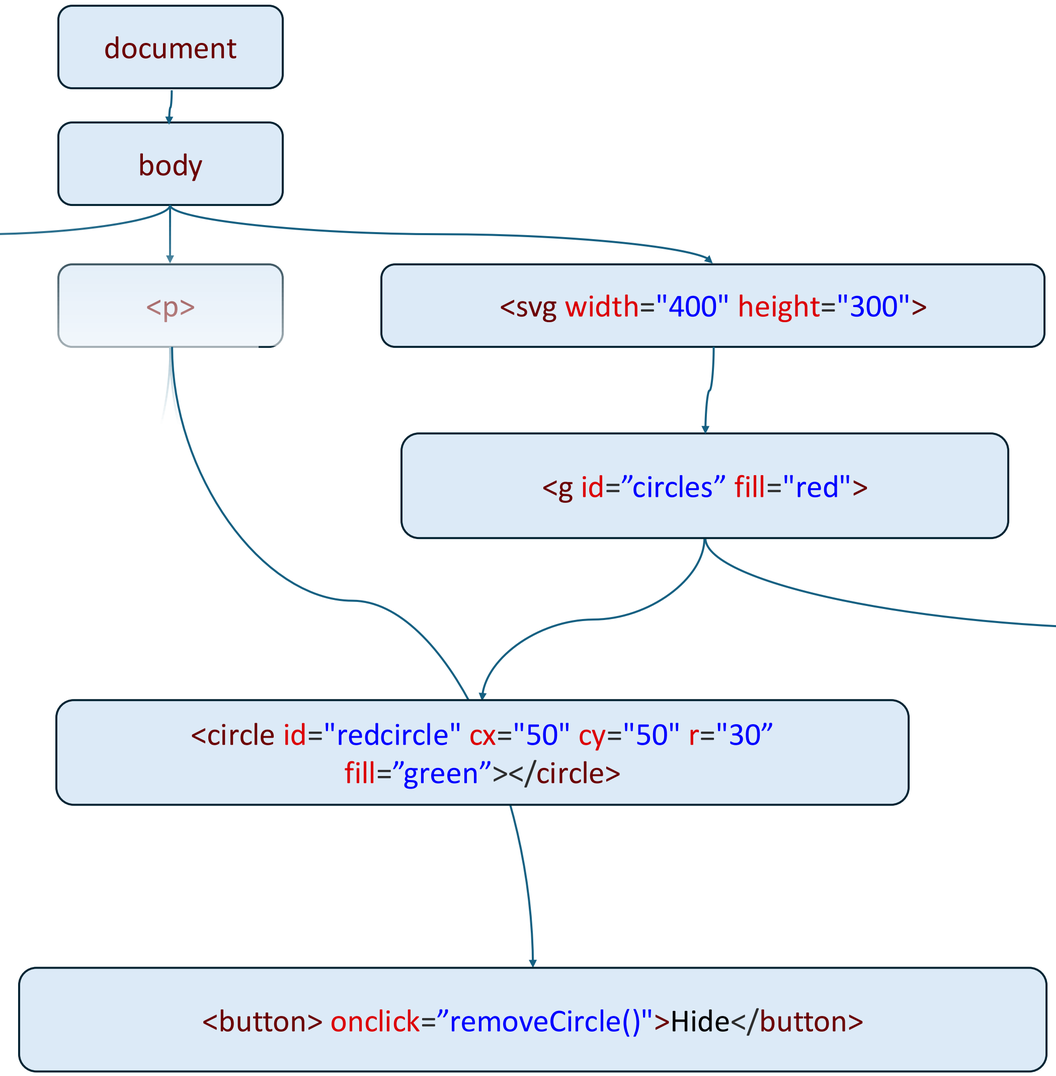
Data Visualization

Show Hide



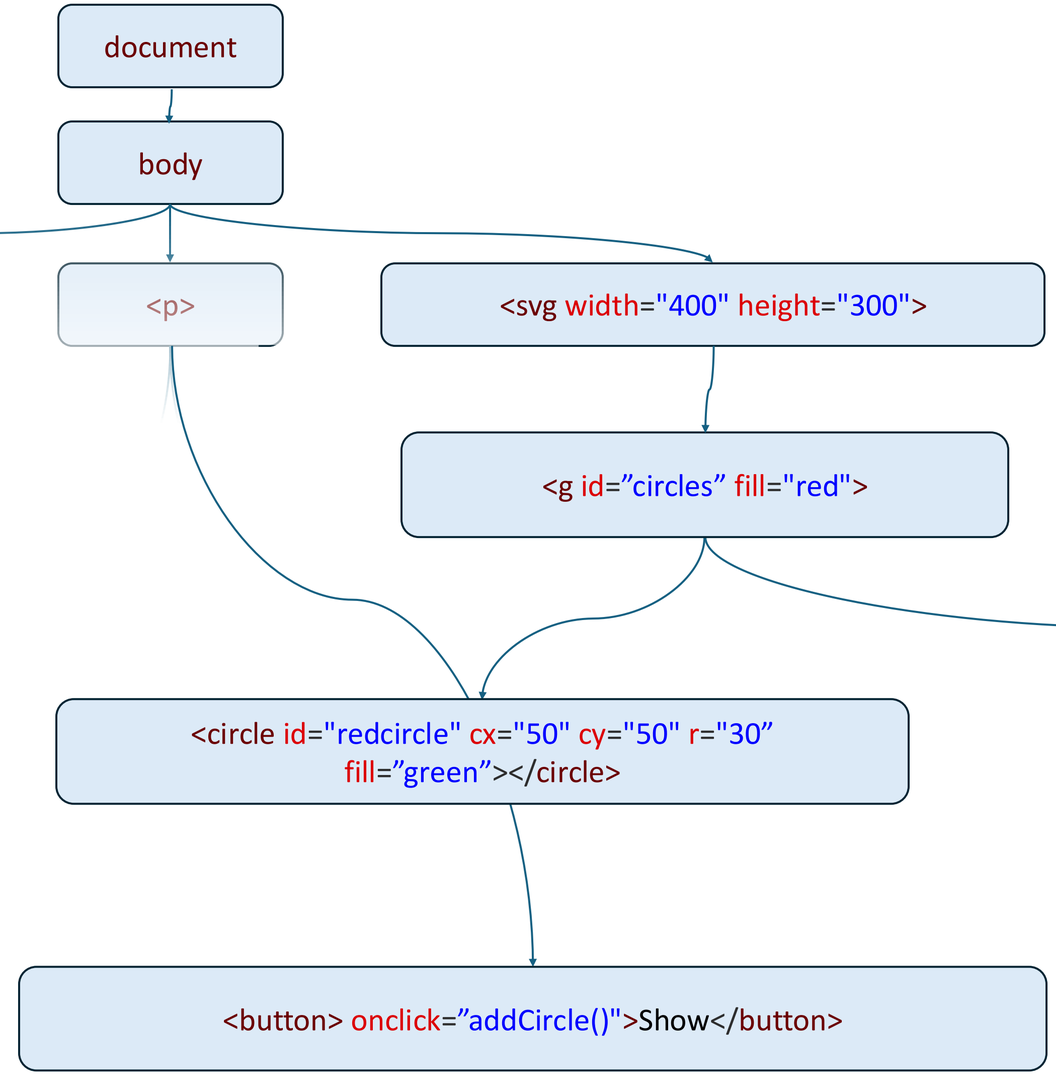

```
let redcircle = document.getElementById('redcircle');
let circles = document. getElementById(circles');

function removeCircle() {
    redcircle.remove()
}

function addCircle() {
    svg.appendChild(redcircle);
}
```

Data Visualization



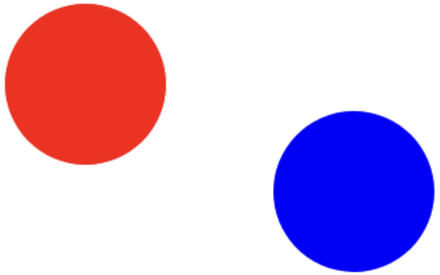
Show Hide

```
let redcircle = document.getElementById('redcircle');
let circles = document. getElementById(circles');

function removeCircle() {
    redcircle.remove()
}

function addCircle() {
    svg.appendChild(redcircle);
}
```



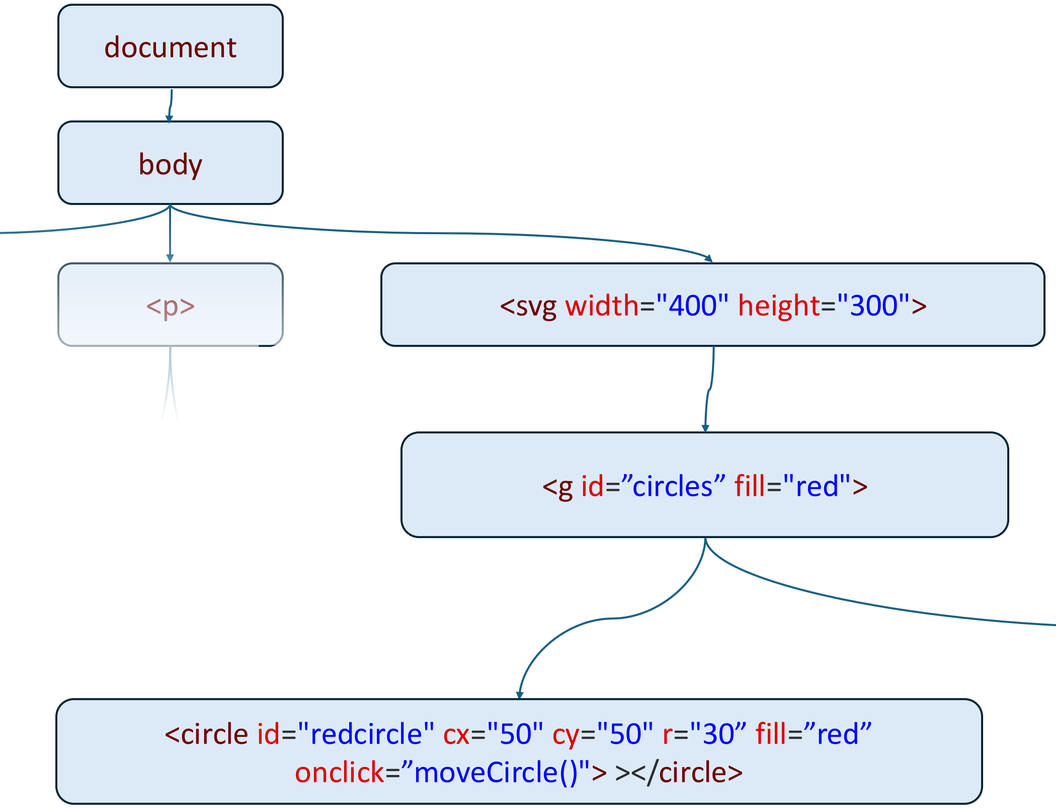
Data Visualization

Show Hide



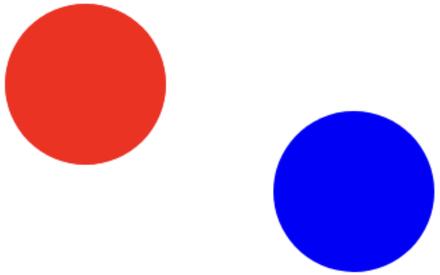

```
let redcircle = document.getElementById('redcircle');
let circles = document. getElementById(circles');

function removeCircle() {
    redcircle.remove()
}

function addCircle() {
    svg.appendChild(redcircle);
}
```

Basic Animation



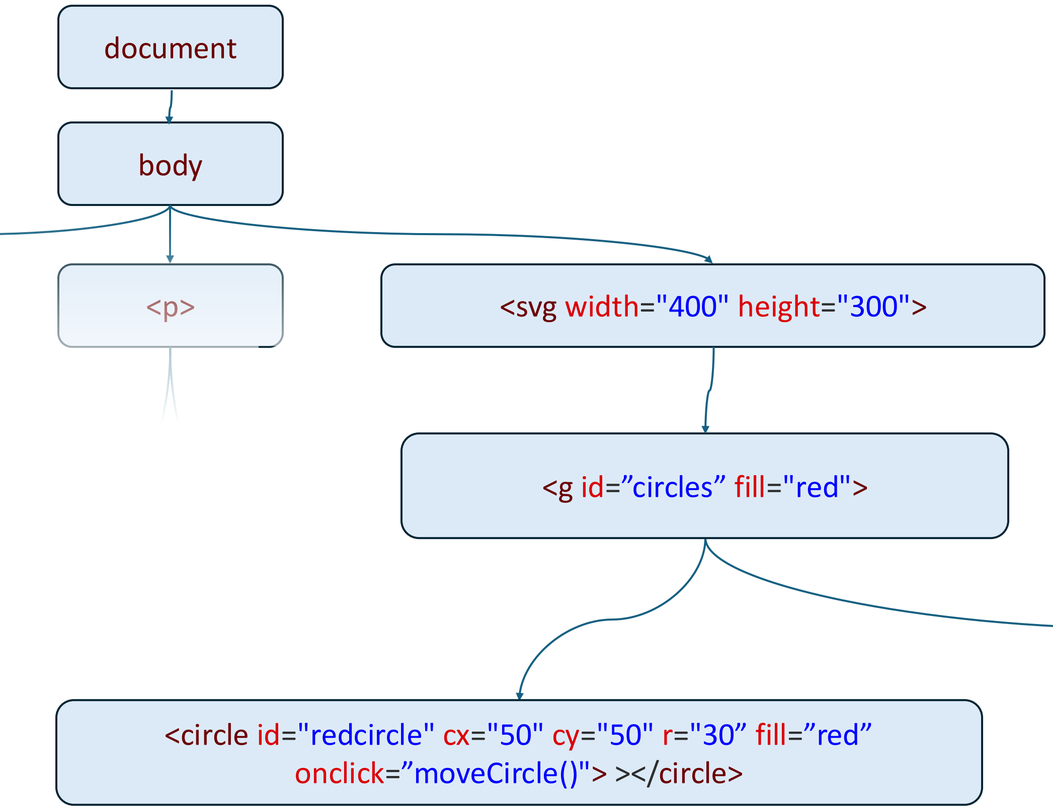
Data Visualization

Show Hide

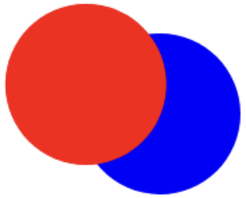


```
function moveCircle() {
    const duration = 1000;
    let i = 0;
    setInterval(() => {
        const t = Math.min(i++ / duration, 1);
        redcircle.setAttribute('cx', t * 150 + (1 - t) * 50);
        redcircle.setAttribute('cy', t * 90 + (1 - t) * 50);
        }, 1)
        }

redcircle.addEventListener('click', moveCircle);
```

Data Visualization



Show Hide

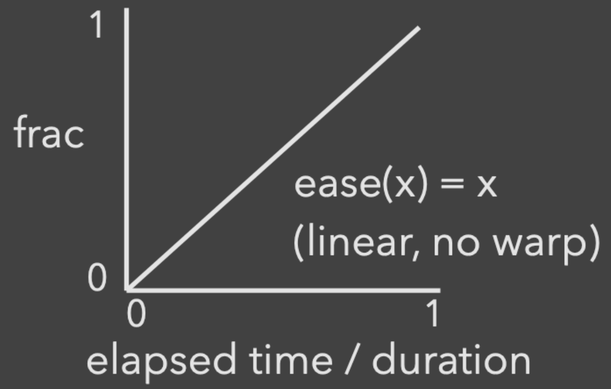


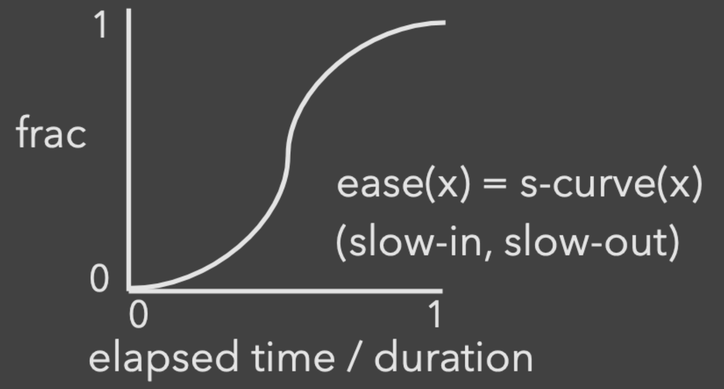
```
function moveCircle() {
    const duration = 1000;
    let i = 0;
    setInterval(() => {
        const t = Math.min(i++ / duration, 1);
        redcircle.setAttribute('cx', t * 150 + (1 - t) * 50);
        redcircle.setAttribute('cy', t * 90 + (1 - t) * 50);
        }, 1)
        }

redcircle.addEventListener('click', moveCircle);
```

Easing (or “Pacing”) Functions

Goals: stylize animation, improve perception.

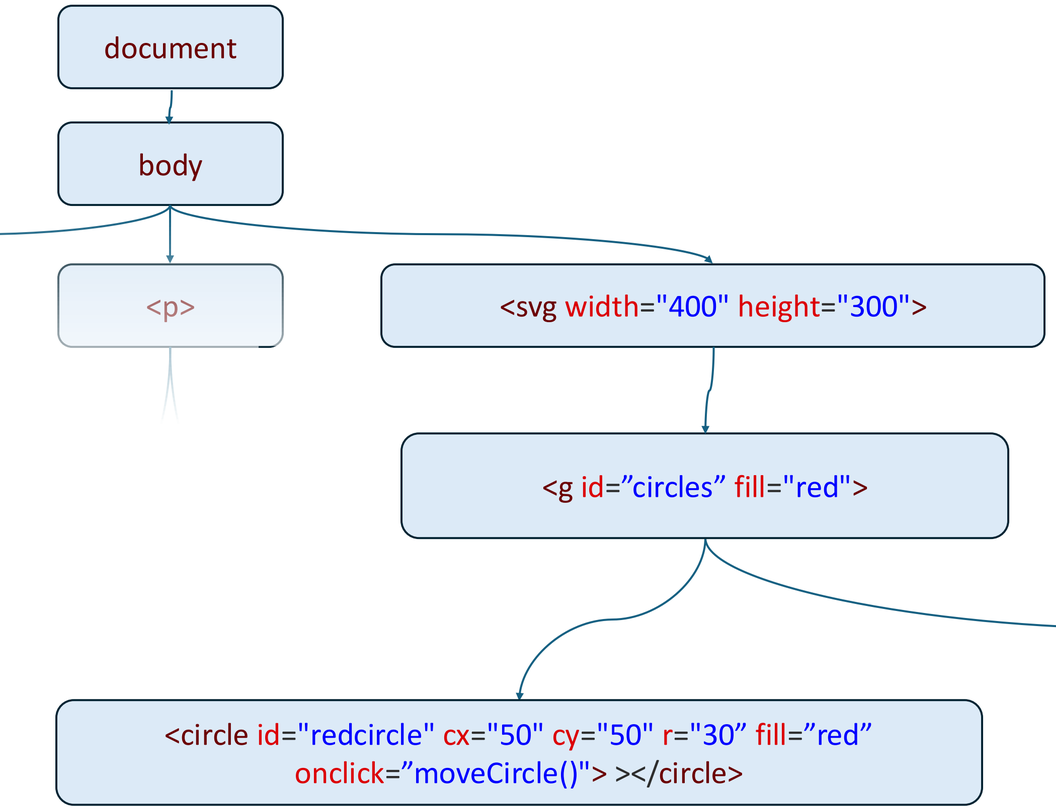
Basic idea is to warp time: as *duration* goes from start (0%) to end (100%), dynamically adjust the *interpolation fraction* using an **easing function**.



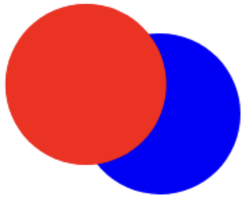


From Jeff Heer

Data Visualization



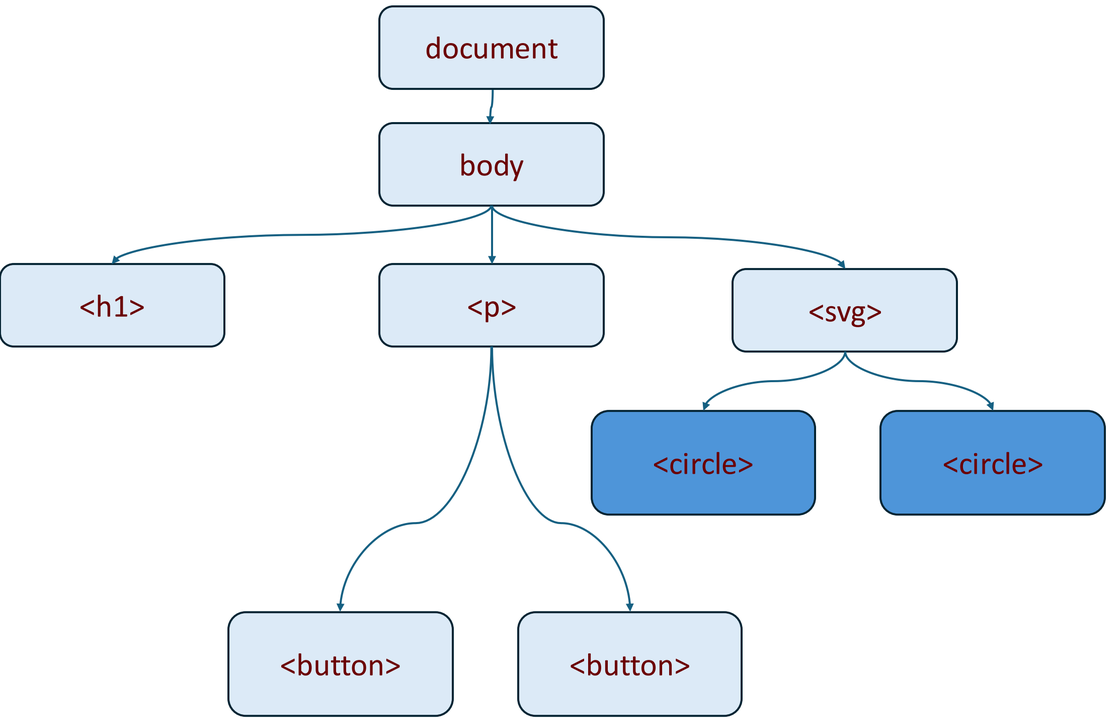
Show Hide



https://easings.net/

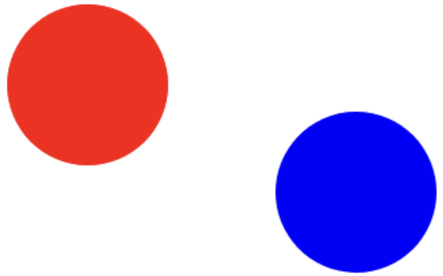
```
function moveCircle() {
    const duration = 1000;
    let i = 0;
    setInterval(() => {
        const t = Math.min(i++ / duration, 1);

        t = ease(t); //
        redcircle.setAttribute('cx', t * 150 + (1 - t) * 50);
        redcircle.setAttribute('cy', t * 90 + (1 - t) * 50);
        }, 1)
        }

redcircle.addEventListener('click', moveCircle);
```

Selecting Elements



Data Visualization

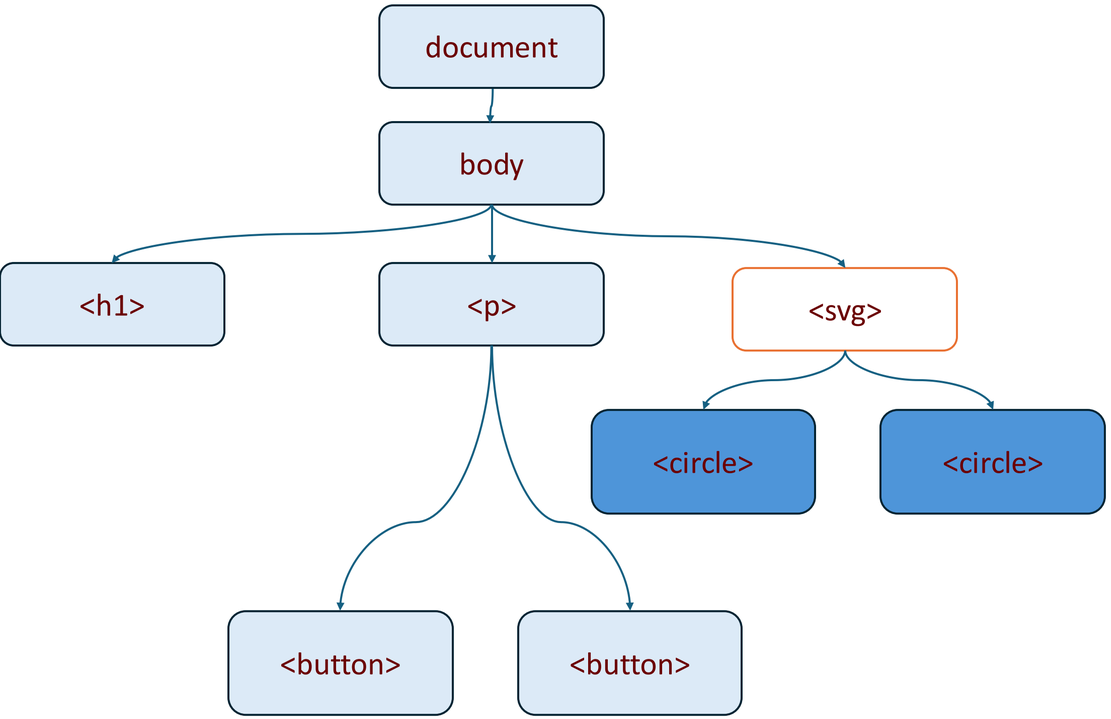
Show Hide



Inspector Console Debugger Network

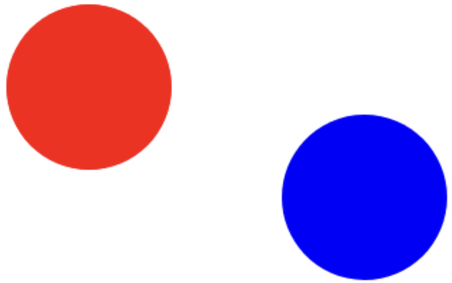
Filter Output Errors Warnings

```
document.querySelectorAll('circle')
NodeList [ circle#redcircle , circle#bluecircle ]
```



Data Visualization

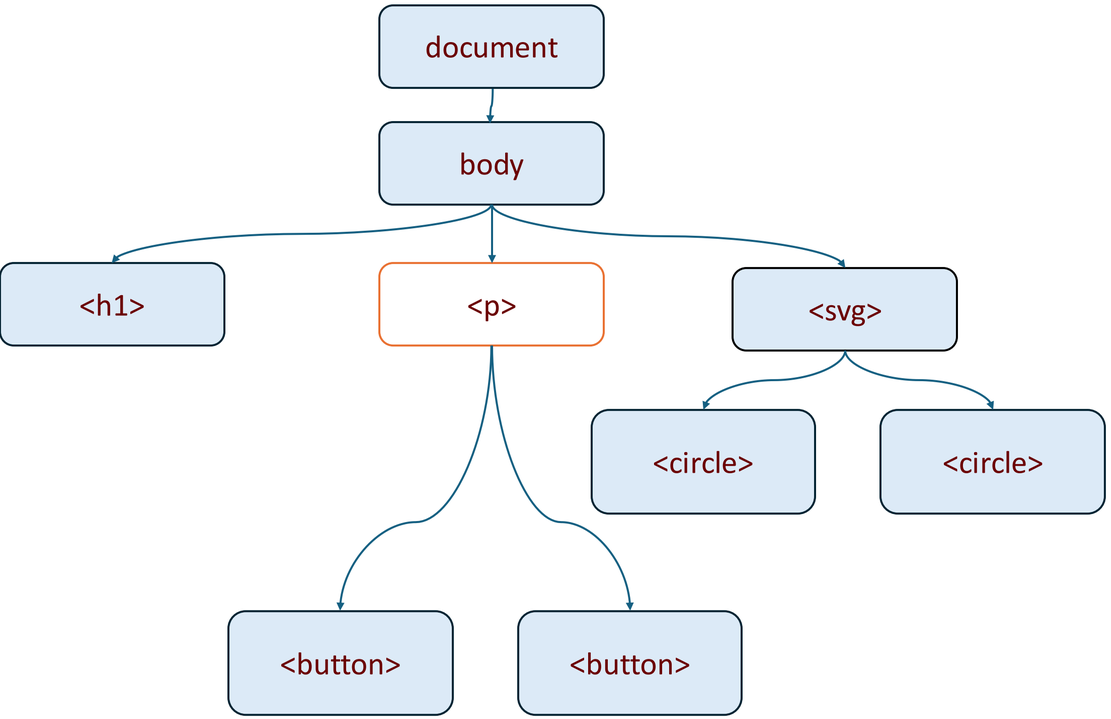
Show Hide



Inspector Console Debugger Network

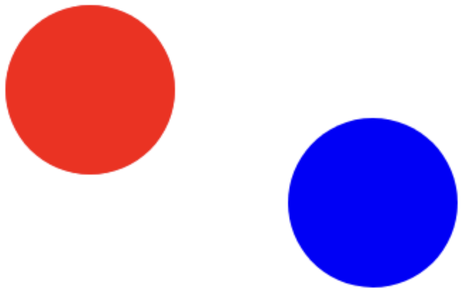
Filter Output Errors Warnings

```
>> svg.querySelectorAll('circle')
<- ▶ NodeList [ circle#redcircle , circle#bluecircle ]
```



Data Visualization

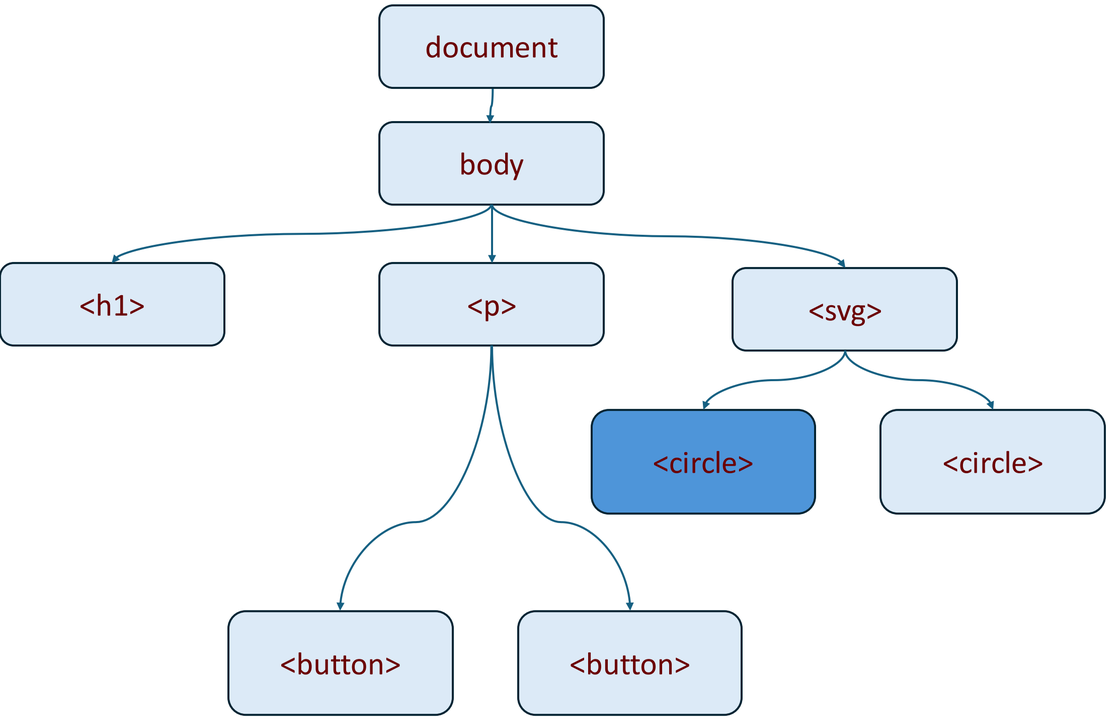
Show Hide

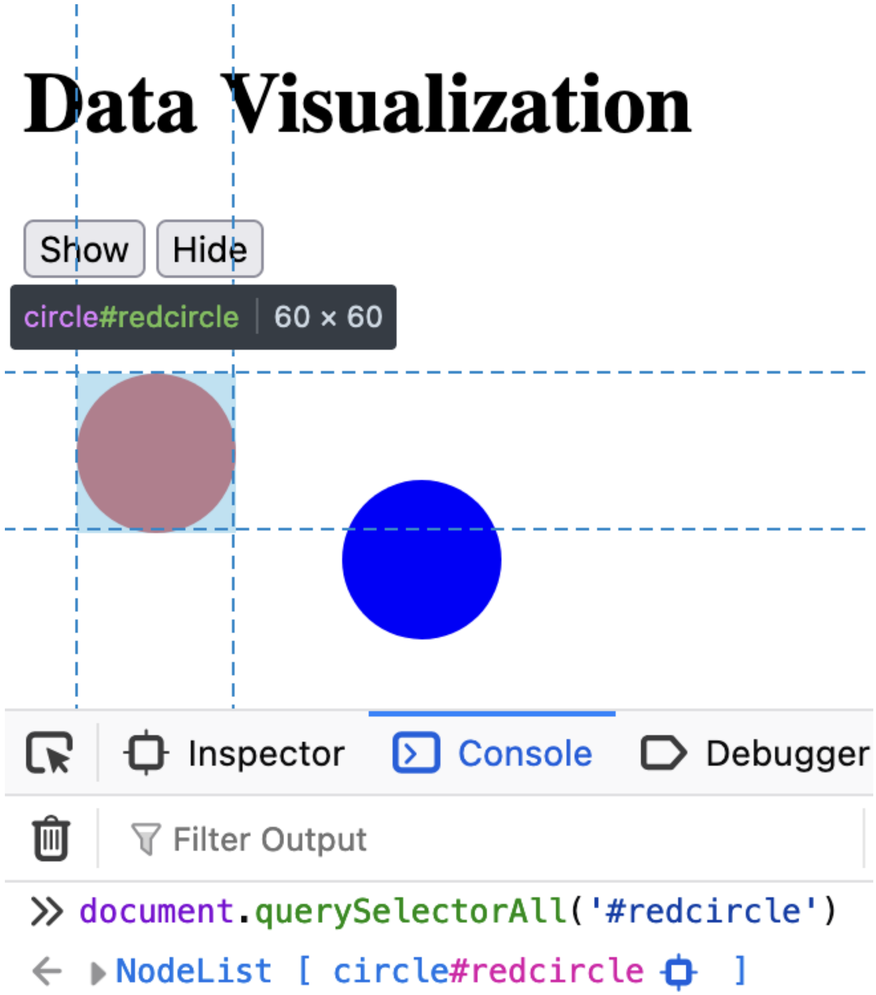


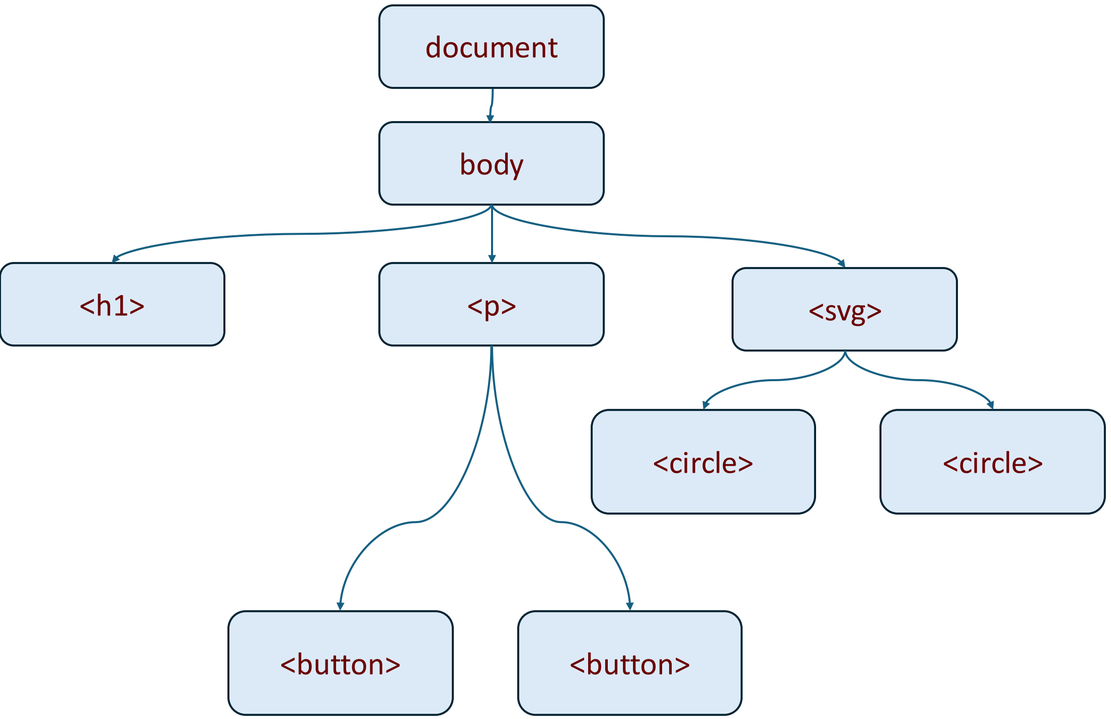
Inspector Console Debugger Network {

Filter Output Errors Warnings

```
p.querySelectorAll('circle')
NodeList []
```


document
body
<h1>
<p>
<svg>
<circle>
<circle>
<button>
<button>

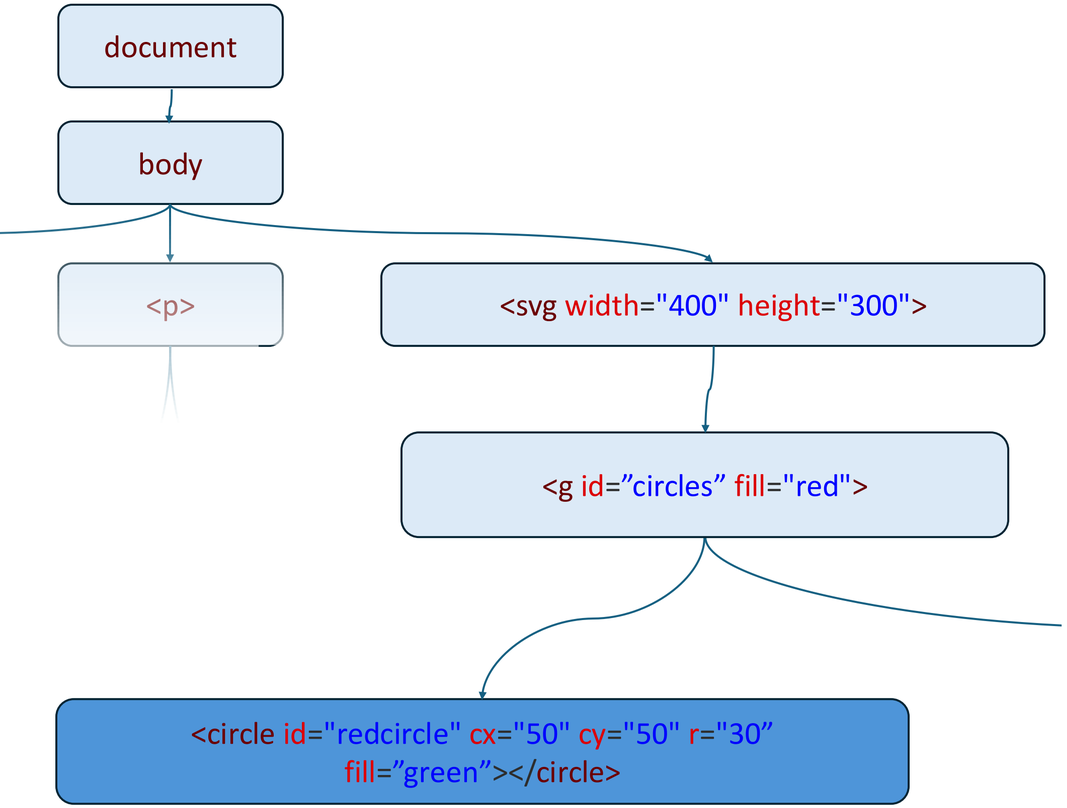

Data Visualization
Show
Hide
circle#redcircle
60 × 60
Inspector
Console
Debugger
Filter Output
document.querySelectorAll('#redcircle')
NodeList [circle#redcircle]

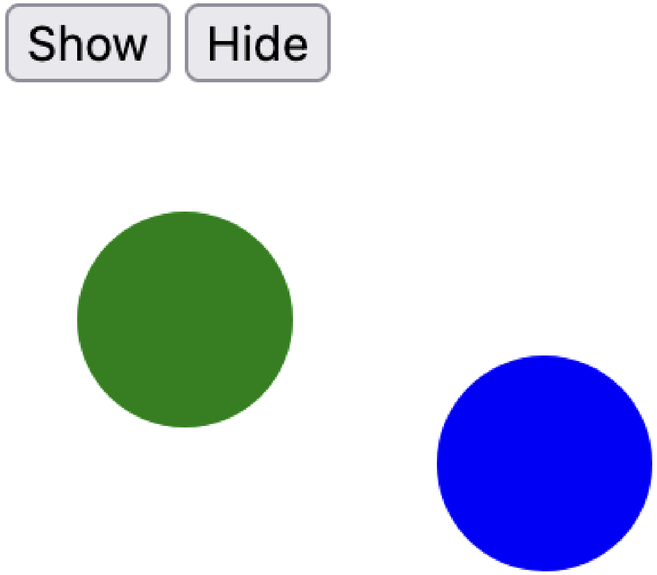


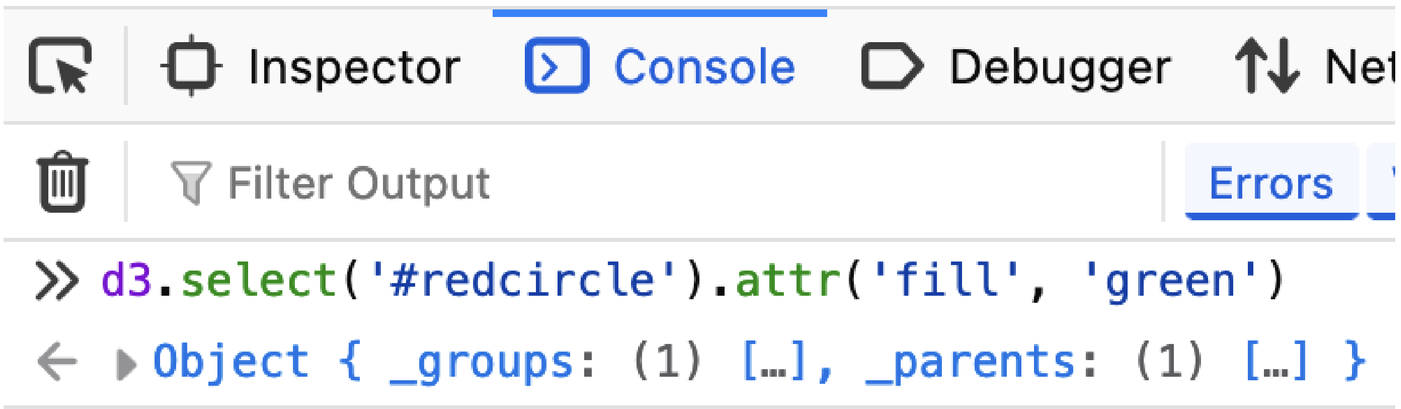

```
#foo        // <any id="foo">
foo         // <foo>
.foo        // <any class="foo">
[foo=bar]     // <any foo="bar">
foo bar     // <foo><bar></foo>
```

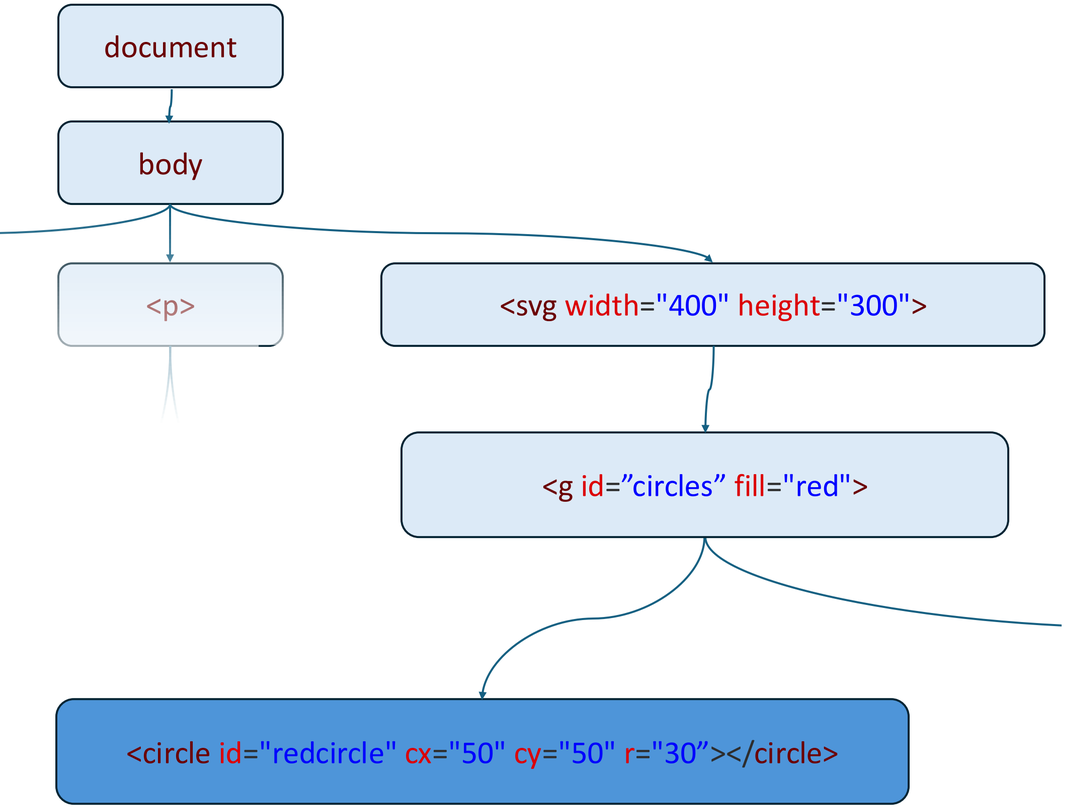
D3.js

Data Visualization



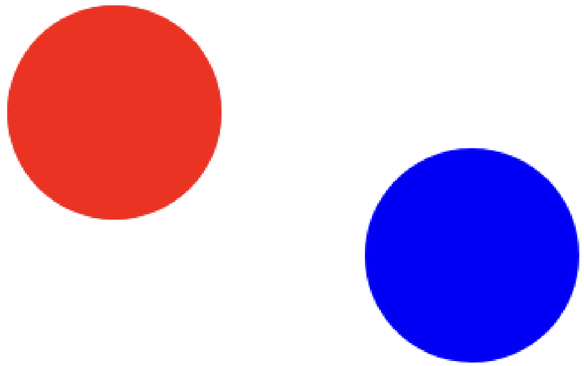


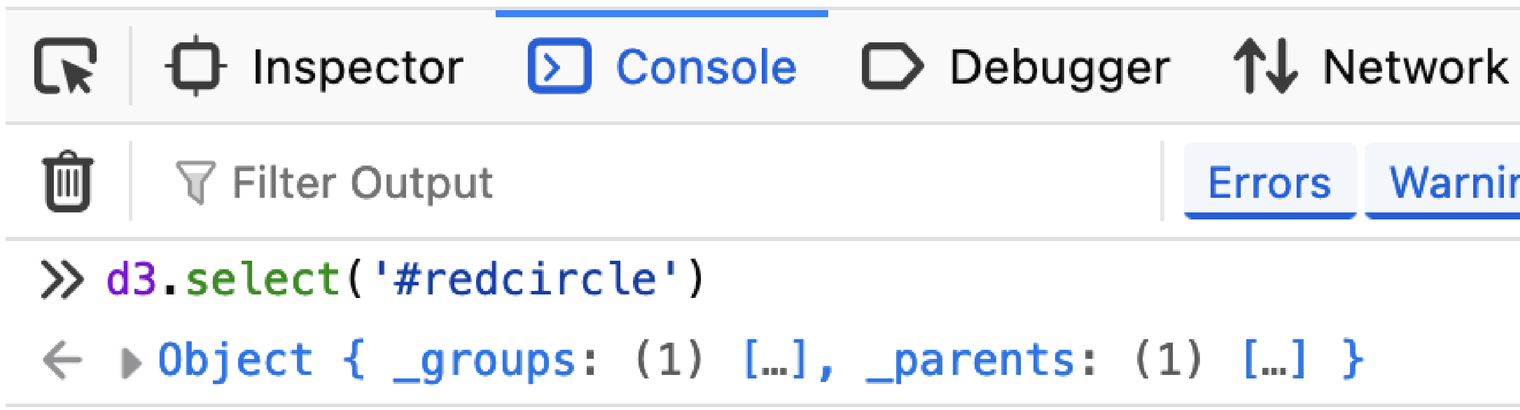


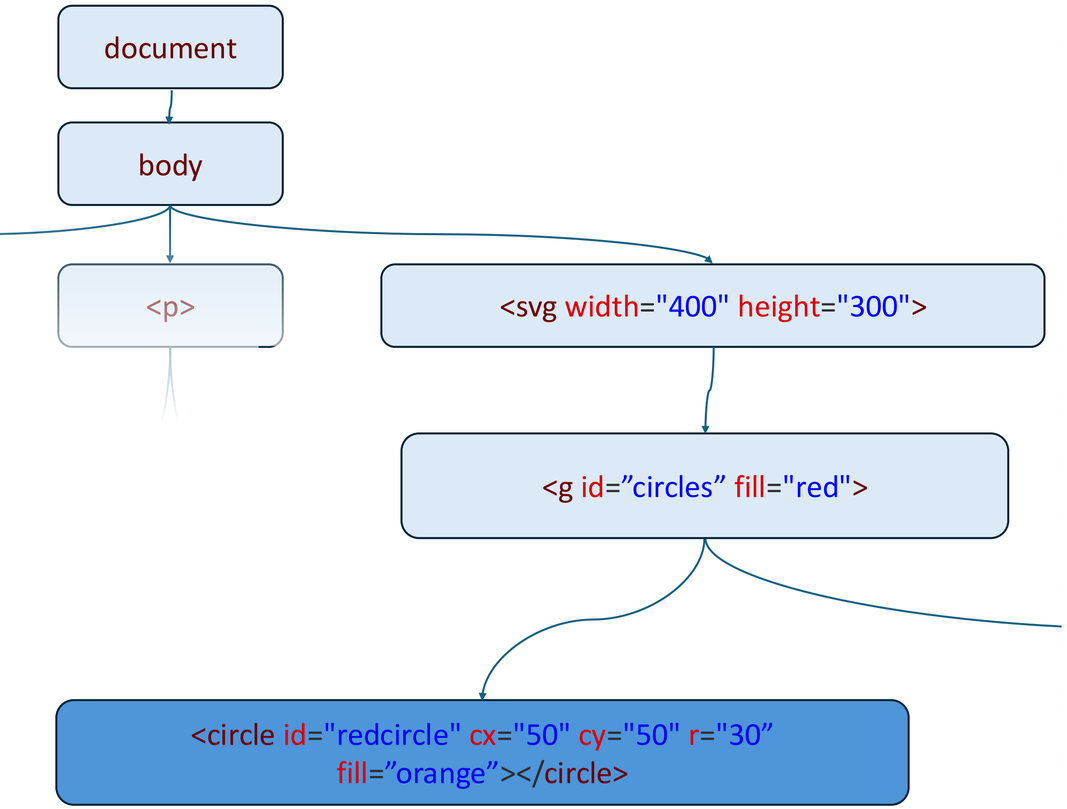


Data Visualization

Show Hide

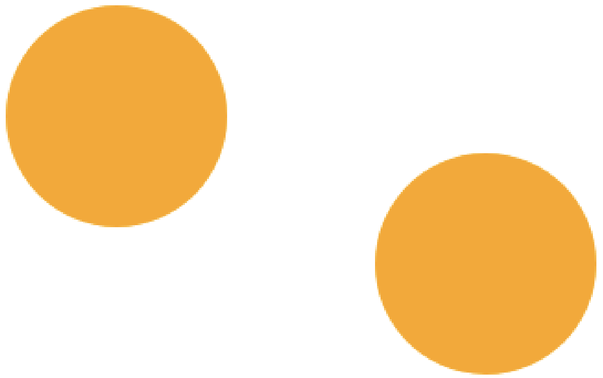


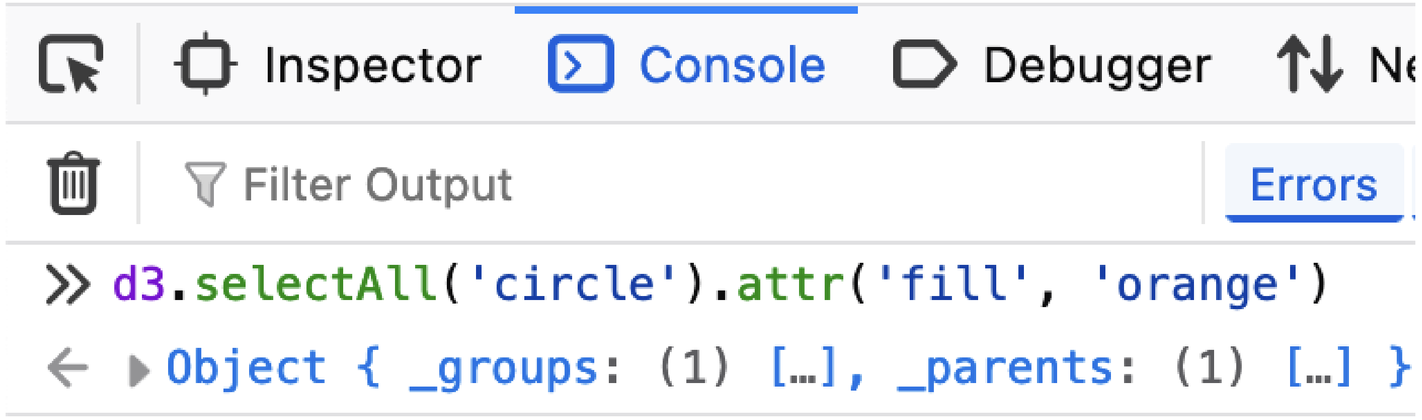


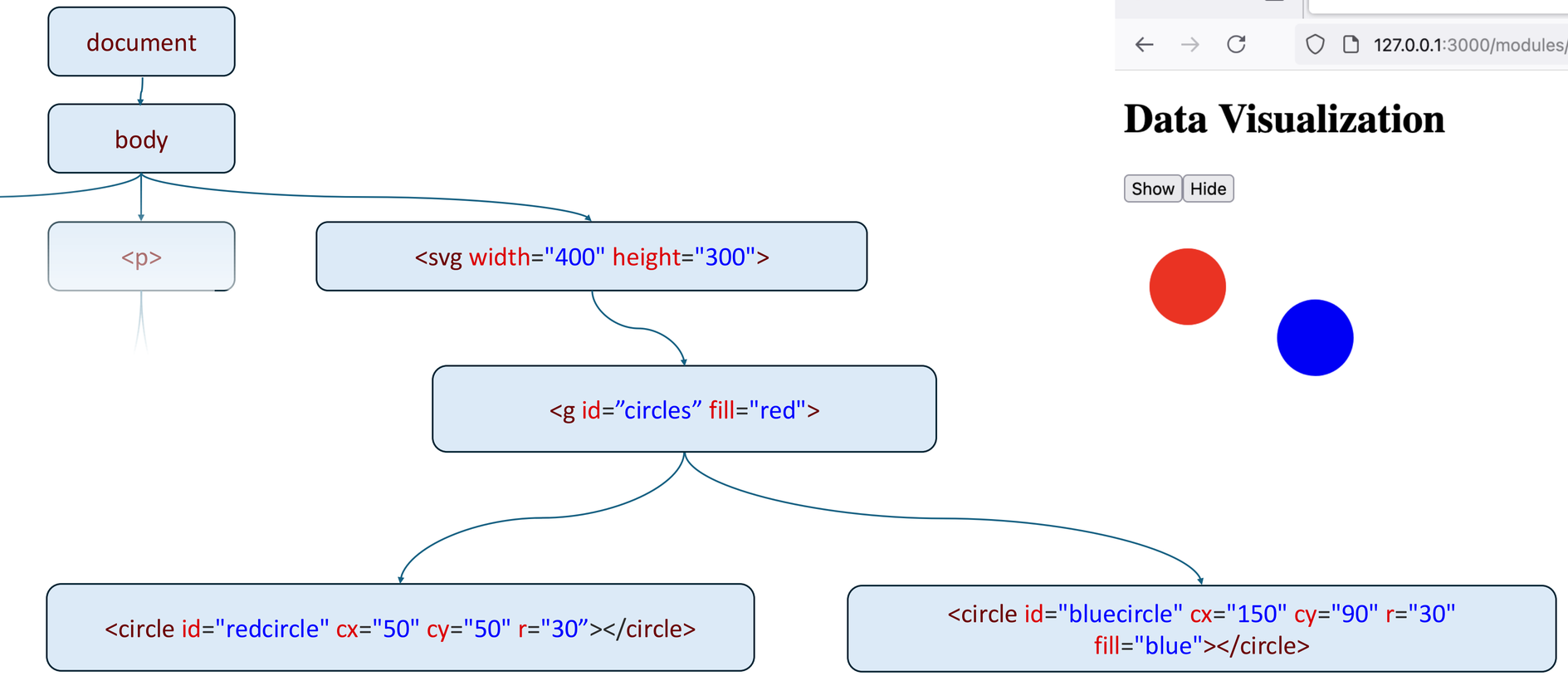


Data Visualization

Show Hide



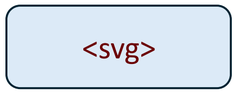



document
body
<p>
<svg width="400" height="300">
<g id=”circles” fill="red">
<circle id="redcircle" cx="50" cy="50" r="30”></circle>
<circle id="bluecircle" cx="150" cy="90" r="30"
fill="blue"></circle>


127.0.0.1:3000/modules/04-Interact
127.0.0.1:3000/modules/04-Interaction-and-web/lec
Data Visualization
Show
Hide

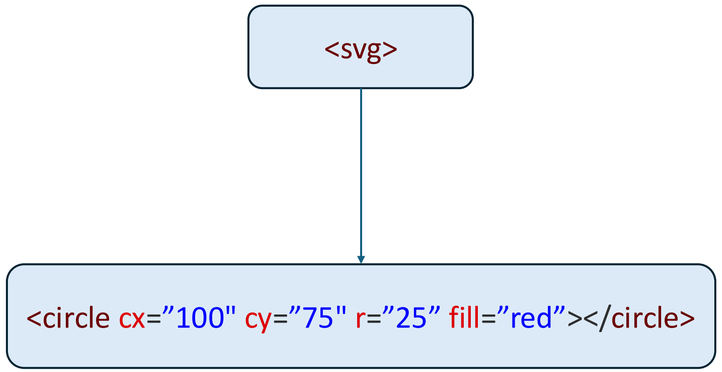
Observable and D3


2
1 + 1



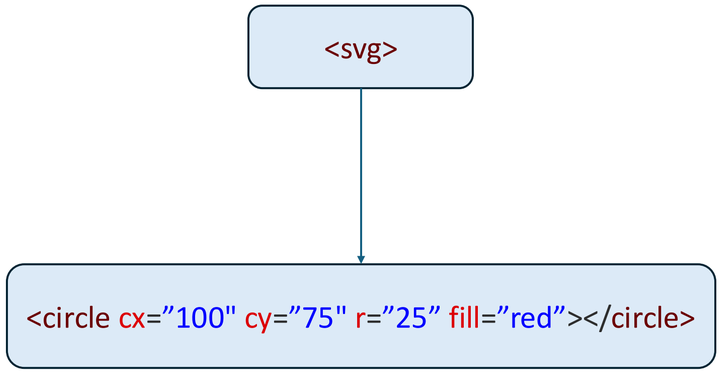

```
{
  const svg = d3.create("svg");
  return svg.node();
}
```

Creation with D3



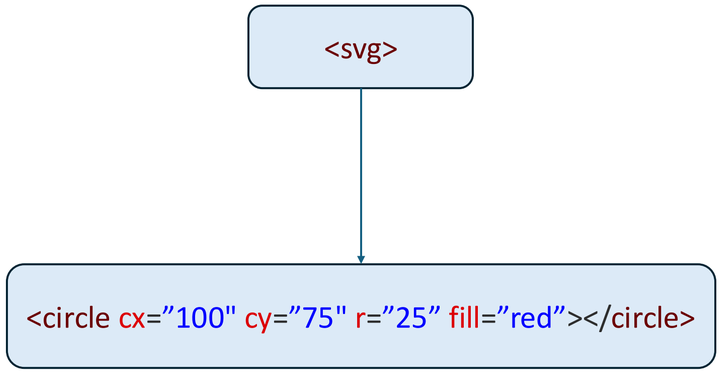
```
{
  const svg = d3.create("svg");
  const circle = svg.append('circle');
  circle.attr('cx', '100');
  circle.attr('cy', '75');
  circle.attr('r', '25');
  circle.attr('fill', 'red');
  return svg.node();
}
```

Method chaining



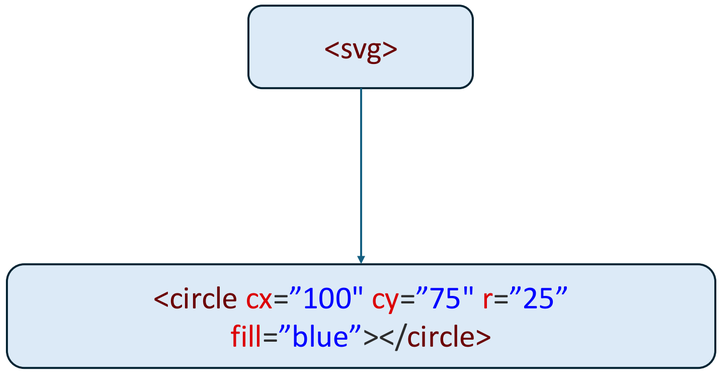
```
{
  const svg = d3.create("svg");
  const circle = svg.append('circle')
                    .attr('cx', '100')
                    .attr('cy', '75')
                    .attr('r', '25')
                    .attr('fill', 'red');
  return svg.node();
}
```

Interaction with D3



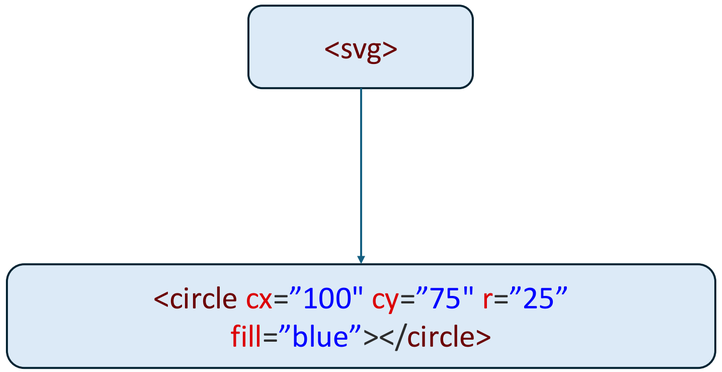
```
{
  const svg = d3.create("svg");
  const circle = svg.append('circle')
                    .attr('cx', '100')
                    .attr('cy', '75')
                    .attr('r', '25')
                    .attr('fill', 'red');

  circle.on('click',
            () => circle.attr('fill', 'blue')
            );
  return svg.node();
}
```

Interaction with D3



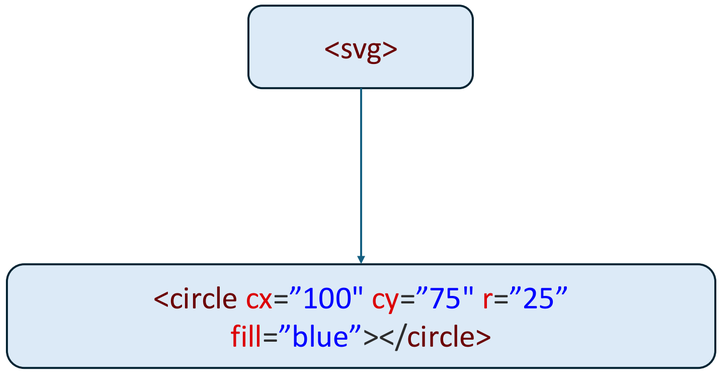
```
{
  const svg = d3.create("svg");
  const circle = svg.append('circle')
                    .attr('cx', '100')
                    .attr('cy', '75')
                    .attr('r', '25')
                    .attr('fill', 'red');

  circle.on('click',
            () => circle.attr('fill', 'blue')
            );
  return svg.node();
}
```

Animation with D3



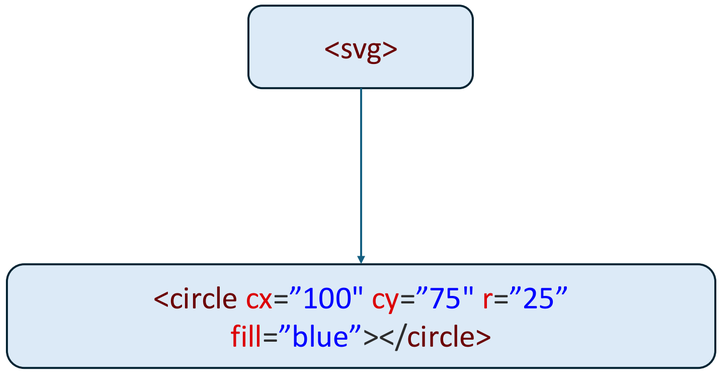
```
{
  const svg = d3.create("svg");
  const circle = svg.append('circle')
                    .attr('cx', '100')
                    .attr('cy', '75')
                    .attr('r', '25')
                    .attr('fill', 'red');

  circle.on('click',
            () => circle
              .transition().duration(10000)
              .attr('fill', 'blue')
              .attr('cx', '200')
            );
  return svg.node();
}
```

Animation with D3



```
{
  const svg = d3.create("svg");
  const circle = svg.append('circle')
                    .attr('cx', '100')
                    .attr('cy', '75')
                    .attr('r', '25')
                    .attr('fill', 'red');

  circle.on('click',
            () => circle
              .transition().duration(10000)
              .attr('fill', 'blue')
              .attr('cx', '200')
            );
  return svg.node();
}
```

Easing with D3



```
{
  const svg = d3.create("svg");
  const circle = svg.append('circle')
                    .attr('cx', '100')
                    .attr('cy', '75')
                    .attr('r', '25')
                    .attr('fill', 'red');     );

  circle.on('click',
            () => circle
              .transition().duration(2000)    )0)
              .ease(d3.easeLinear)
              .attr('fill', 'blue')
              .attr('cx', '200')
            );
  return svg.node();
}
```

Cars dataset

name	economy (mpg)	cylinders	displacement (cc)	power (hp)	weight (lb)	0-60 mph (s)
AMC Ambassador ...	13	8	360	175	3,821	11
AMC Ambassador ...	15	8	390	190	3,850	8.5
AMC Ambassador ...	17	8	304	150	3,672	11.5
AMC Concord DL 6	20.2	6	232	90	3,265	18.2
AMC Concord DL	18.1	6	258	120	3,410	15.1
AMC Concord DL	23	4	151		3,035	20.5
AMC Concord	19.4	6	232	90	3,210	17.2
AMC Concord	24.3	4	151	90	3,003	20.1
AMC Gremlin	18	6	232	100	2,789	15
AMC Gremlin	19	6	232	100	2,634	13
AMC Gremlin	20	6	232	100	2,914	16

```
Inputs.table(cars)
```

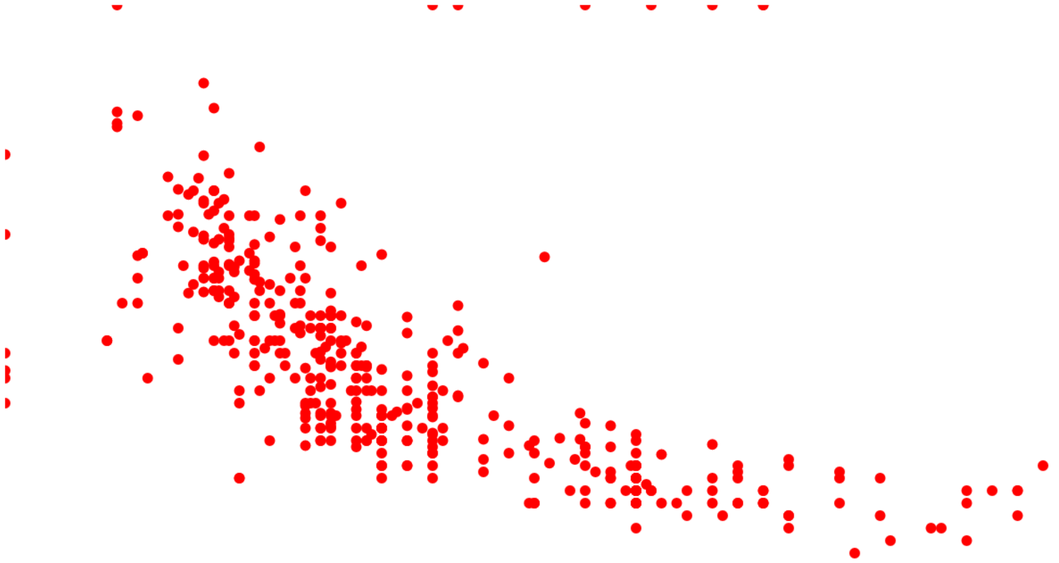
```
▶ Array(406) [Object, Object, Object, Object,
```

```
cars
```

```
▼ Object {
    name: "AMC Ambassador Brougham"
    economy (mpg): 13
    cylinders: 8
    displacement (cc): 360
    power (hp): 175
    weight (lb): 3821
    0-60 mph (s): 11
    year: 73
  }
```

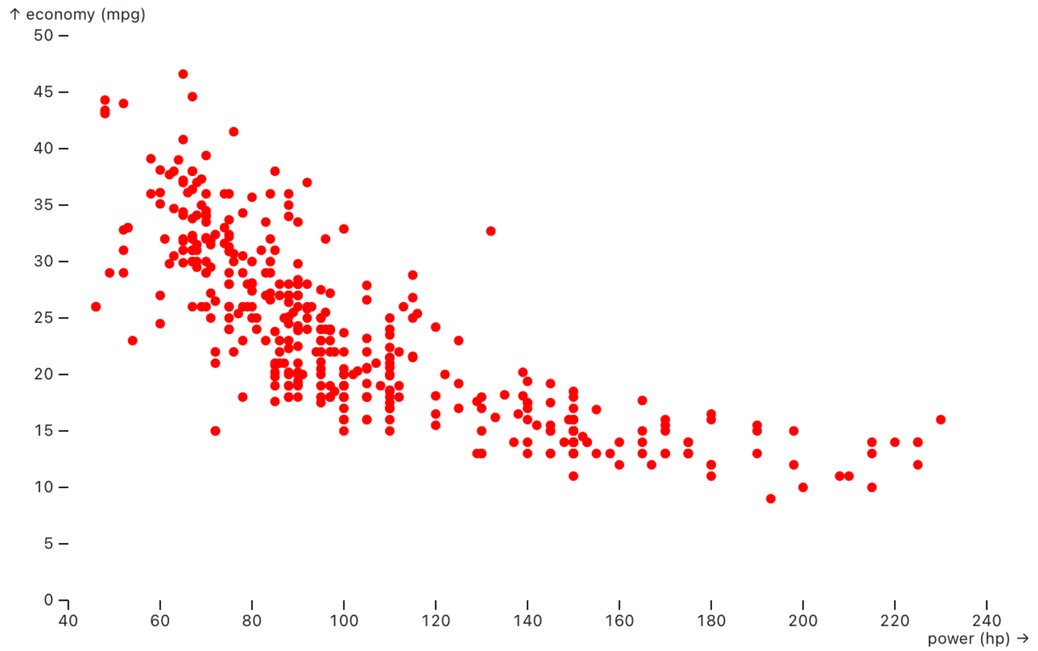
```
cars[0]
```

```
{
  const width = 640;
  const height = 400;
  const margin = {top: 20, right: 30, bottom: 30, left: 40};

  const svg = d3.create("svg")
      .attr("width", width)
      .attr("height", height);

  const x = d3.scaleLinear()
      .domain([40, 240])
      .range([margin.left, width - margin.right]);

  const y = d3.scaleLinear()
      .domain([0, 50])
      .range([height - margin.bottom, margin.top]);

  svg.selectAll('circle')
      .data(cars)
      .enter().append('circle')
      .attr("fill", "red")
      .attr("cx", (d) => x(d["power (hp)"]))
      .attr("cy", (d) => y(d["economy (mpg)"]))
      .attr("r", 3)

  return svg.node();
}
```



```d3
{
  const width = 640;
  const height = 400;
  const margin = {top: 20, right: 30, bottom: 30, left: 40};

  const svg = d3.create("svg")
      .attr("width", width)
      .attr("height", height);

  const x = d3.scaleLinear()
      .domain([40, 240])
      .range([margin.left, width - margin.right]);

  const y = d3.scaleLinear()
      .domain([0, 50])
      .range([height - margin.bottom, margin.top]);

  svg.selectAll('circle')
      .data(cars)
      .enter().append('circle')
      .attr("fill", "red")
      .attr("cx", (d) => x(d["power (hp)"]))
      .attr("cy", (d) => y(d["economy (mpg)"]))
      .attr("r", 3)

  return svg.node();
}
```

```plot
{}  Plot.plot({
      width: 640,
      height: 400,
      marginTop: 20, marginRight: 30, marginBottom: 30, marginLeft: 40,

      x: {domain: [40, 240]},
      y: {domain: [0, 50]},
      marks: [
        Plot.dot(cars, {x: "power (hp)",
                        y: "economy (mpg)",
                        r: 3,
                        fill: 'red'
                       }),
      ]
    })
```



Creating a plot

```
{
  const width = 640;
  const height = 400;
  const margin = {top: 20, right: 30, bottom: 30, left: 40};

  const svg = d3.create("svg")
      .attr("width", width)
      .attr("height", height);
```

<svg width=”640" height=”400” >

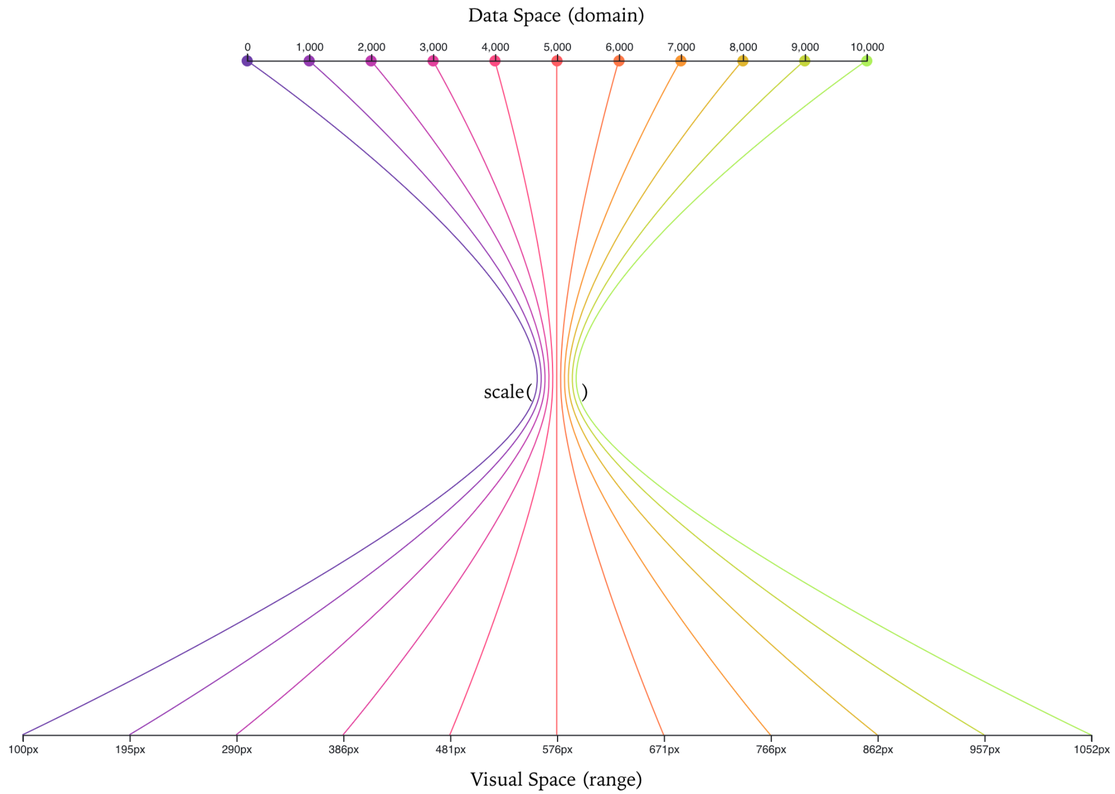
```
Plot.plot({
  width: 640,
  height: 400,
  marginTop: 20, marginRight: 30, marginBottom: 30, marginLeft: 40,
```

Creating scales

```
const x = d3.scaleLinear()
    .domain([40, 240])
    .range([margin.left, width - margin.right]);

const y = d3.scaleLinear()
    .domain([0, 50])
    .range([height - margin.bottom, margin.top]);
```

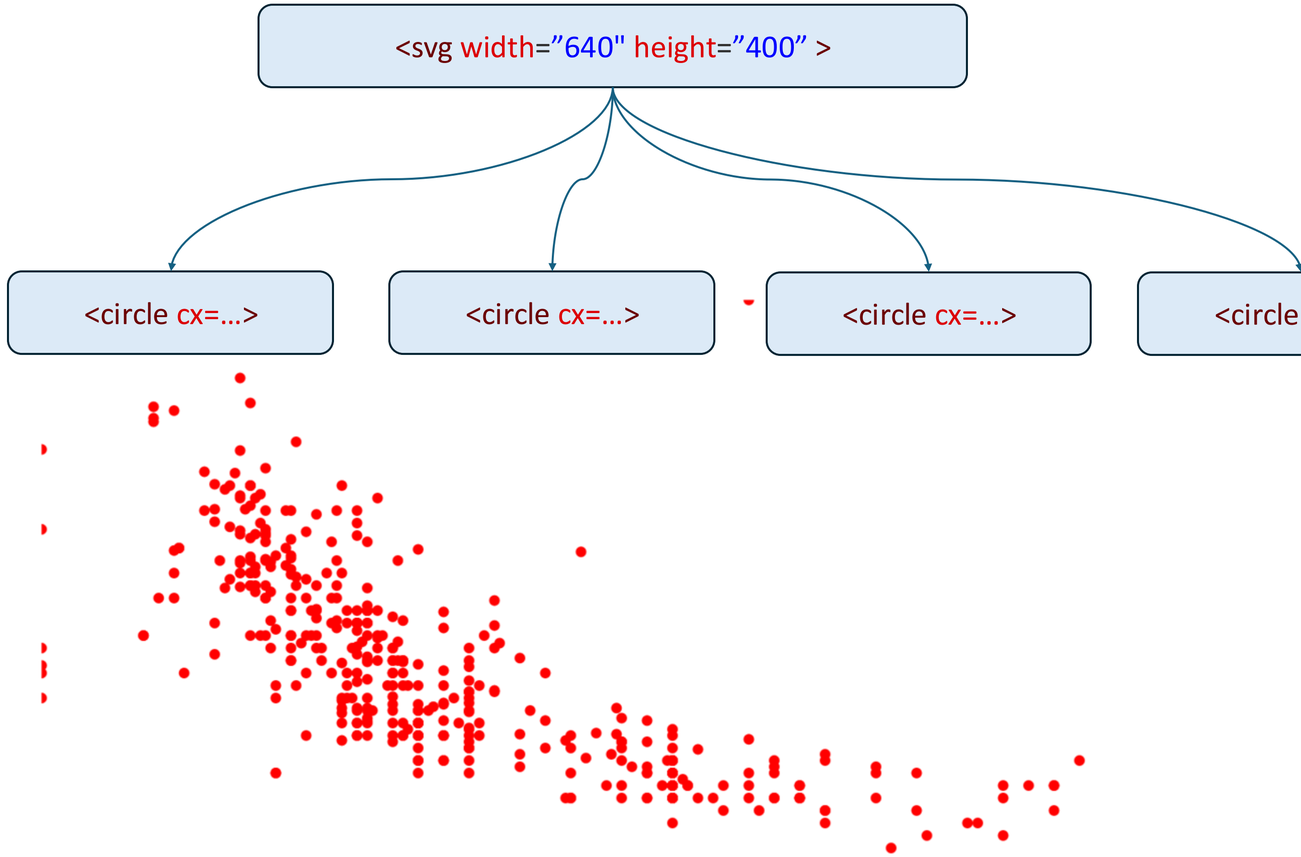
```
x: {domain: [40, 240]},
y: {domain: [0, 50]},
marks: [
```

Transition



Creating a layer

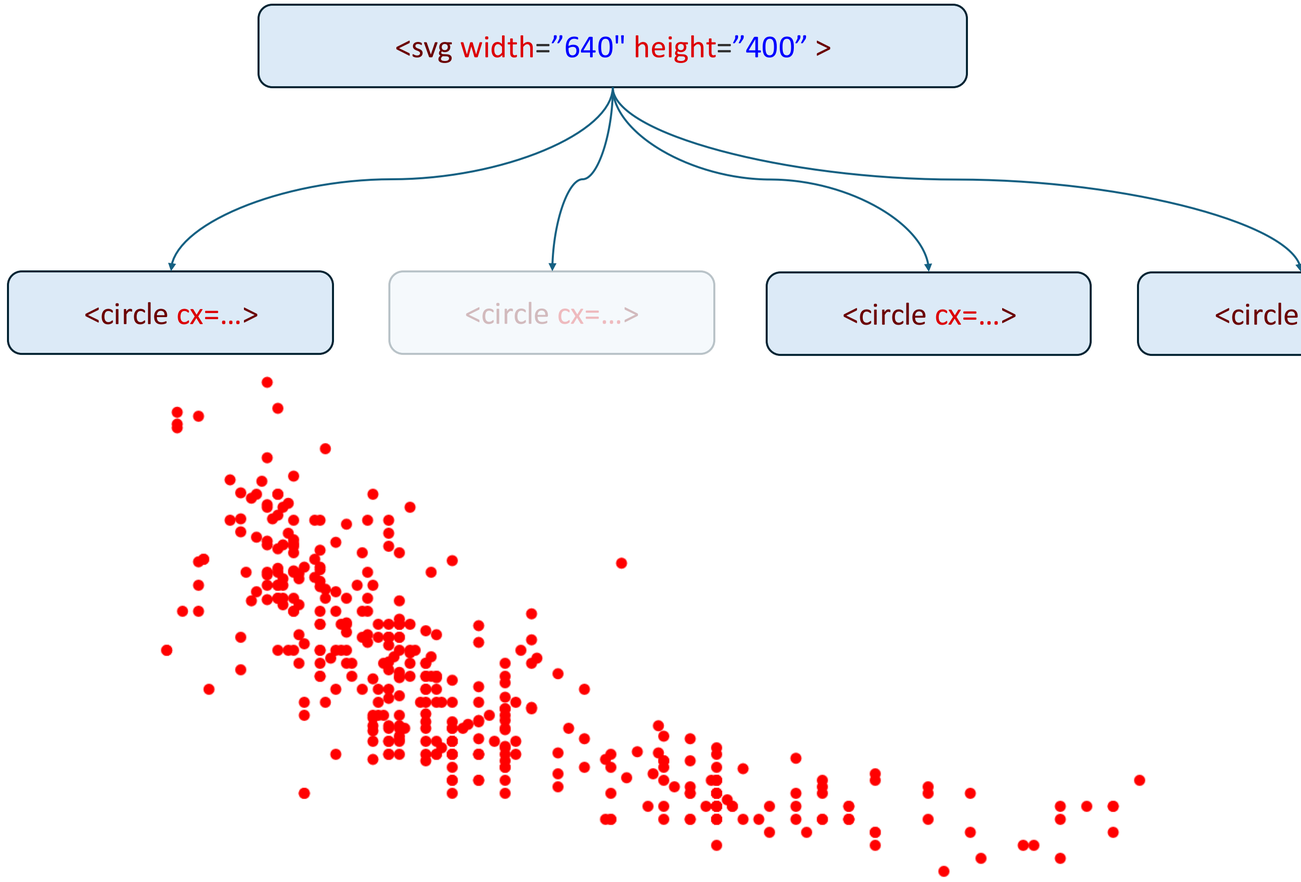
```
svg.selectAll('circle')
    .data(cars)
    .enter().append('circle')
    .filter((d) => d["power (hp)"] > 0 && d["economy (mpg)"] > 0)
    .attr("fill", "red")
    .attr("cx", (d) => x(d["power (hp)"]))
    .attr("cy", (d) => y(d["economy (mpg)"]))
    .attr("r", 3)
```




```
  marks: [
    Plot.dot(cars, {x: "power (hp)",
                    y: "economy (mpg)",
                    r: 3,
                    fill: 'red'
                   }),
  ]
```

Applying a transform (filtering)

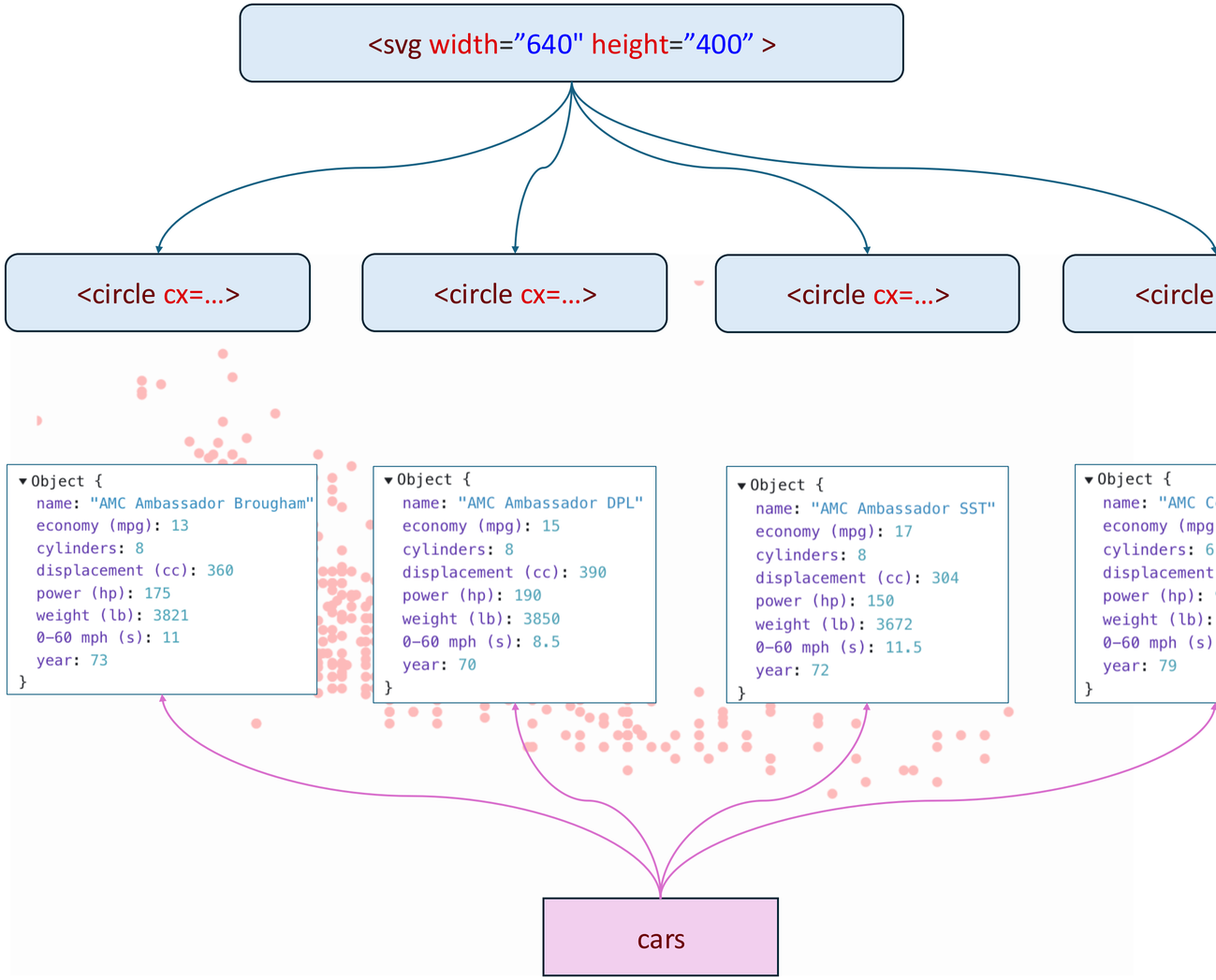
```
svg.selectAll('circle')
    .data(cars)
    .enter().append('circle')
    .filter((d) => d["power (hp)"] > 0 && d["economy (mpg)"] > 0)
    .attr("fill", "red")
    .attr("cx", (d) => x(d["power (hp)"]))
    .attr("cy", (d) => y(d["economy (mpg)"]))
    .attr("r", 3)
```



Aesthetic mappings

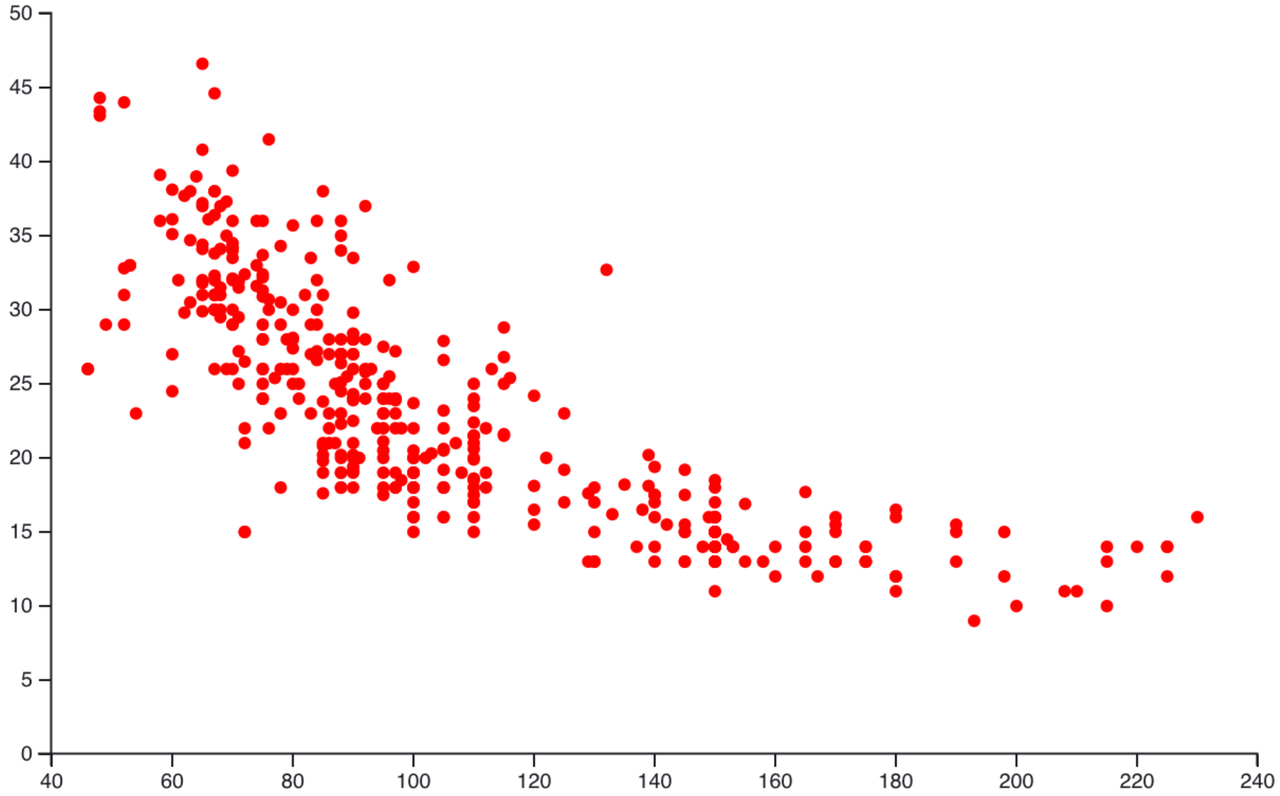
```
svg.selectAll('circle')
    .data(cars)
    .enter().append('circle')
    .filter((d) => d["power (hp)"] > 0 && d["economy (mpg)"] > 0)
    .attr("fill", "red")
    .attr("cx", (d) => x(d["power (hp)"]))
    .attr("cy", (d) => y(d["economy (mpg)"]))
    .attr("r", 3)
```

```
  marks: [
    Plot.dot(cars, {x: "power (hp)",
                    y: "economy (mpg)",
                    r: 3,
                    fill: 'red'
                   }),
  ]
```



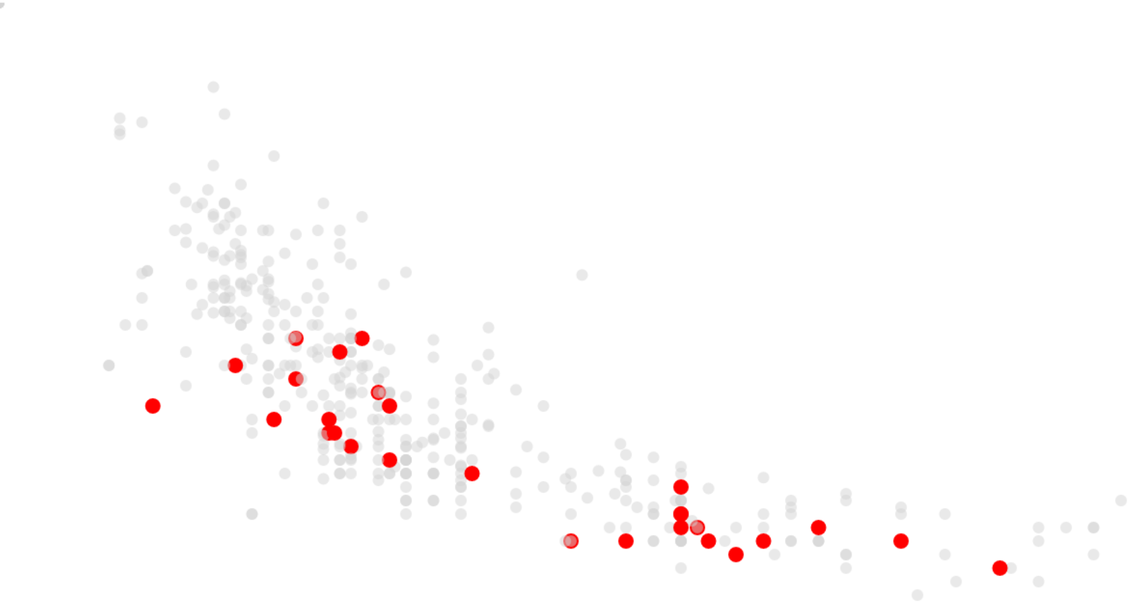
Creating axes

```
svg.append("g")
    .attr("transform", `translate(0,${height - margin.bottom})`)
    .call(d3.axisBottom(x));

svg.append("g")
    .attr("transform", `translate(${margin.left},0)`)
    .call(d3.axisLeft(y));
```

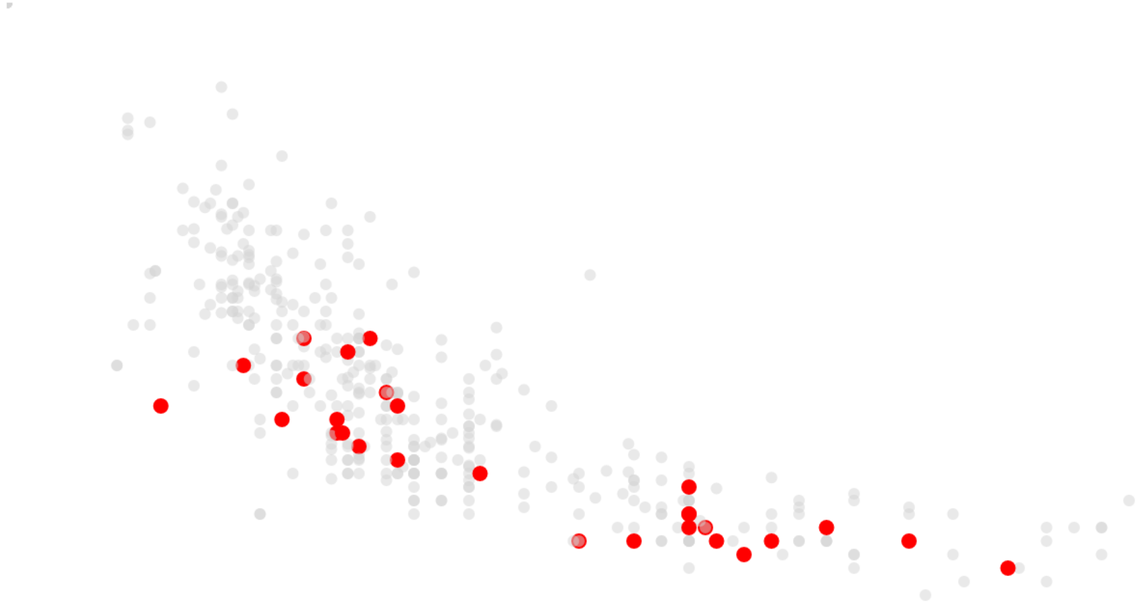


Interaction with D3

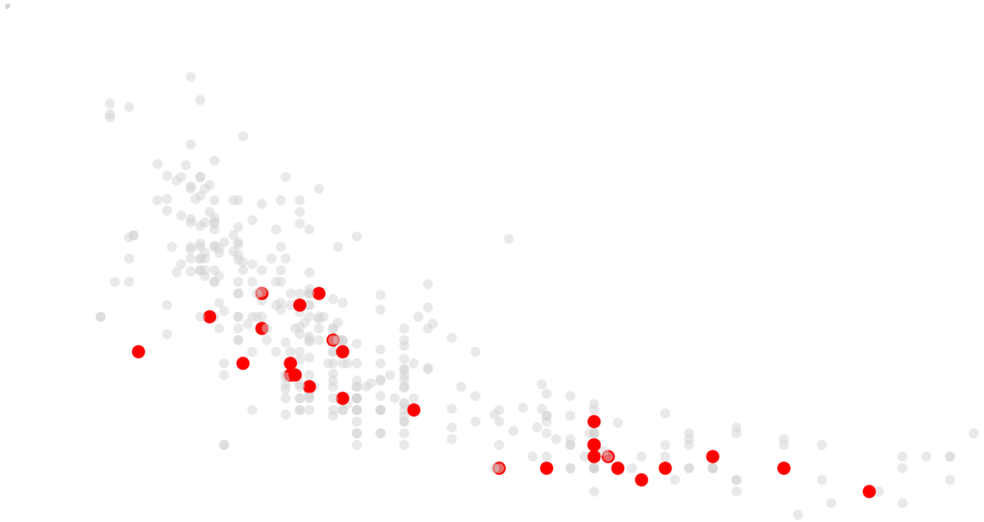
```
function onclick(event, datum) {
    svg.selectAll('circle')
      .attr('opacity', 0.5)
      .attr('fill', 'lightgrey')
      .attr('r', 3)
      .filter((d) => d.year == datum.year)
      .attr('fill', 'red')
      .attr('opacity', 1.)
      .attr('r', 4);
}
```



```
function onclick(event, datum) {
    svg.selectAll('circle')
      .attr('opacity', 0.5)
      .attr('fill', 'lightgrey')
      .attr('r', 3)
      .filter((d) => d.year == datum.year)
      .attr('fill', 'red')
      .attr('opacity', 1.)
      .attr('r', 4);
}
```

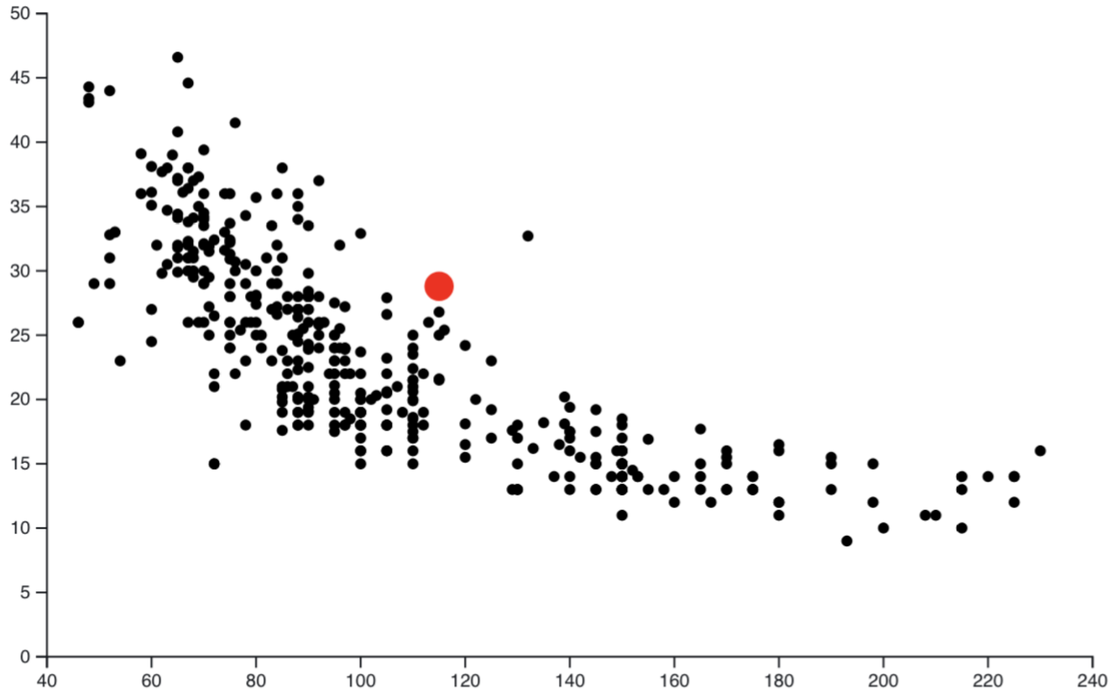


```
svg.selectAll('circle')
    .data(cars)
    .join('circle')
    .filter((d) => d["power (hp)"] > 0 && d["economy (mpg)"] > 0)
    .attr("fill", "red")
    .attr("cx", (d) => x(d["power (hp)"]))
    .attr("cy", (d) => y(d["economy (mpg)"]))
    .attr("r", 3)
    .on('click', onclick)
```



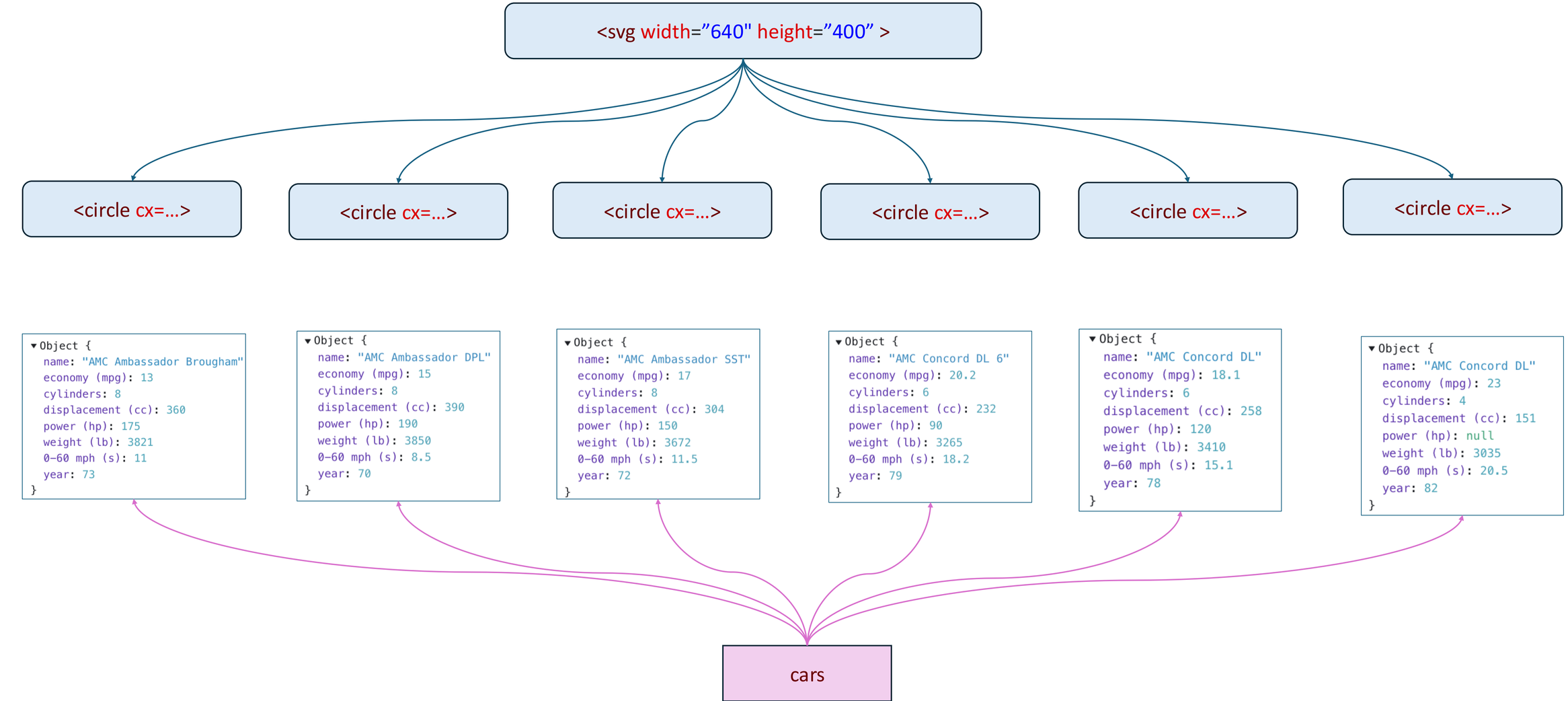
```
function onclick(event, datum) {
  svg.selectAll('circle')
    .transition().duration(500)
    .attr('opacity', 0.5)
    .attr('fill', 'lightgrey')
    .attr('r', 3)
    .filter((d) => d.year == datum.year)
    .attr('fill', 'red')
    .attr('opacity', 1.)
    .attr('r', 4);
}
```

```
svg.selectAll('circle')
    .data(cars)
    .enter().append('circle')
    .filter((d) => d["power (hp)"] > 0 && d["economy (mpg)"] > 0)
    .attr("cx", (d) => x(d["power (hp)"]))
    .attr("cy", (d) => y(d["economy (mpg)"]))
    .attr("r", 3)
    .on('mouseover', (event, datum) => d3.select(event.target).attr('fill', 'red').attr('r', 8))
    .on('mouseout', (event, datum) => d3.select(event.target).attr('fill', 'black').attr('r', 3))
```



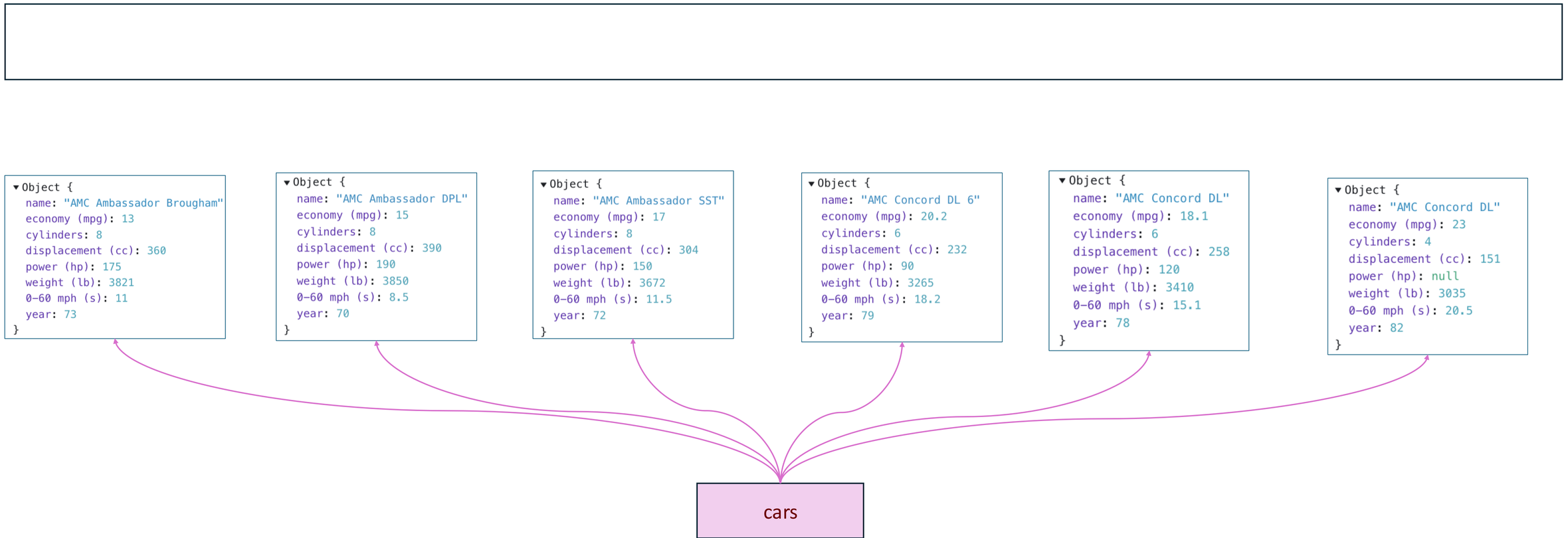
D3 Joins

```
svg.selectAll('circle')
    .data(cars)
    .enter().append('circle')
    .filter((d) => d["power (hp)"] > 0 && d["economy (mpg)"] > 0)
    .attr("fill", "red")
    .attr("cx", (d) => x(d["power (hp)"]))
    .attr("cy", (d) => y(d["economy (mpg)"]))
    .attr("r", 3)
```

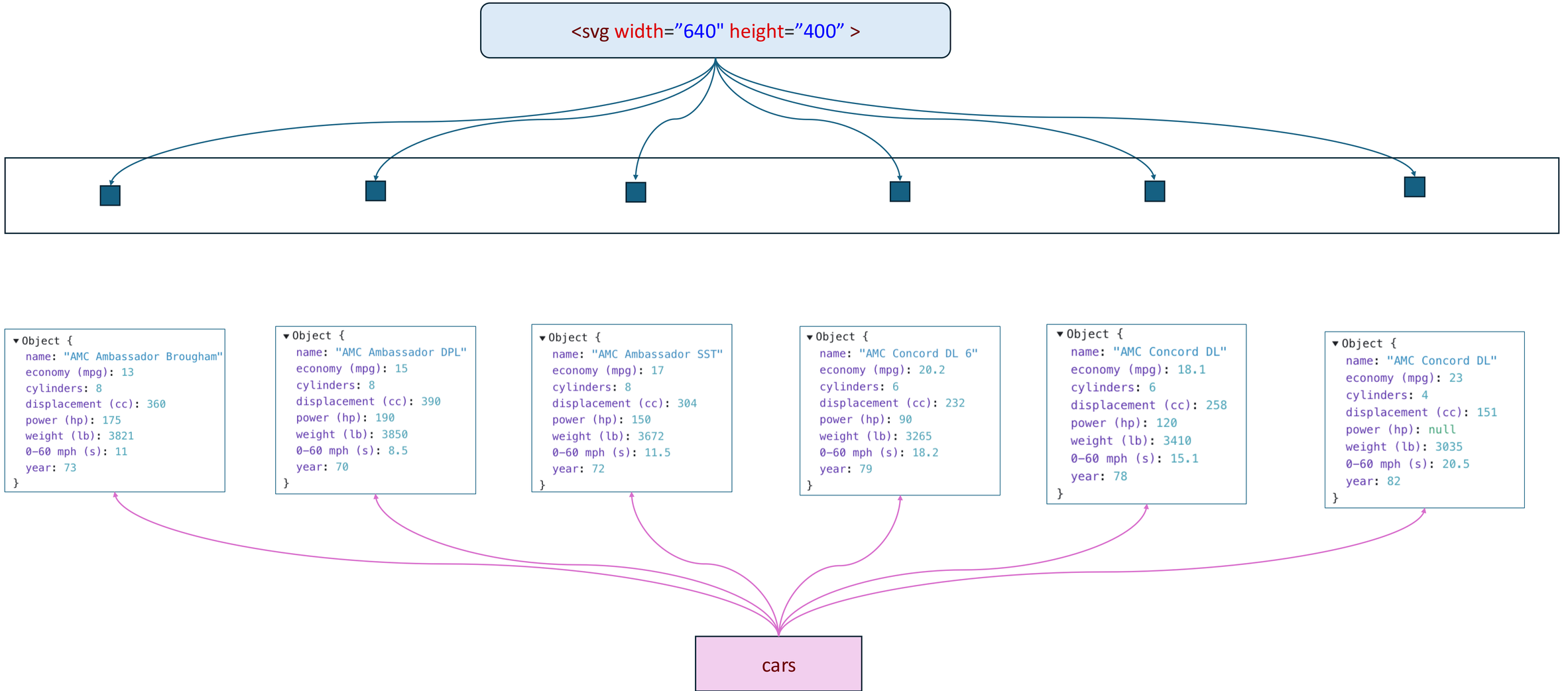


```
svg.selectAll('circle')
```

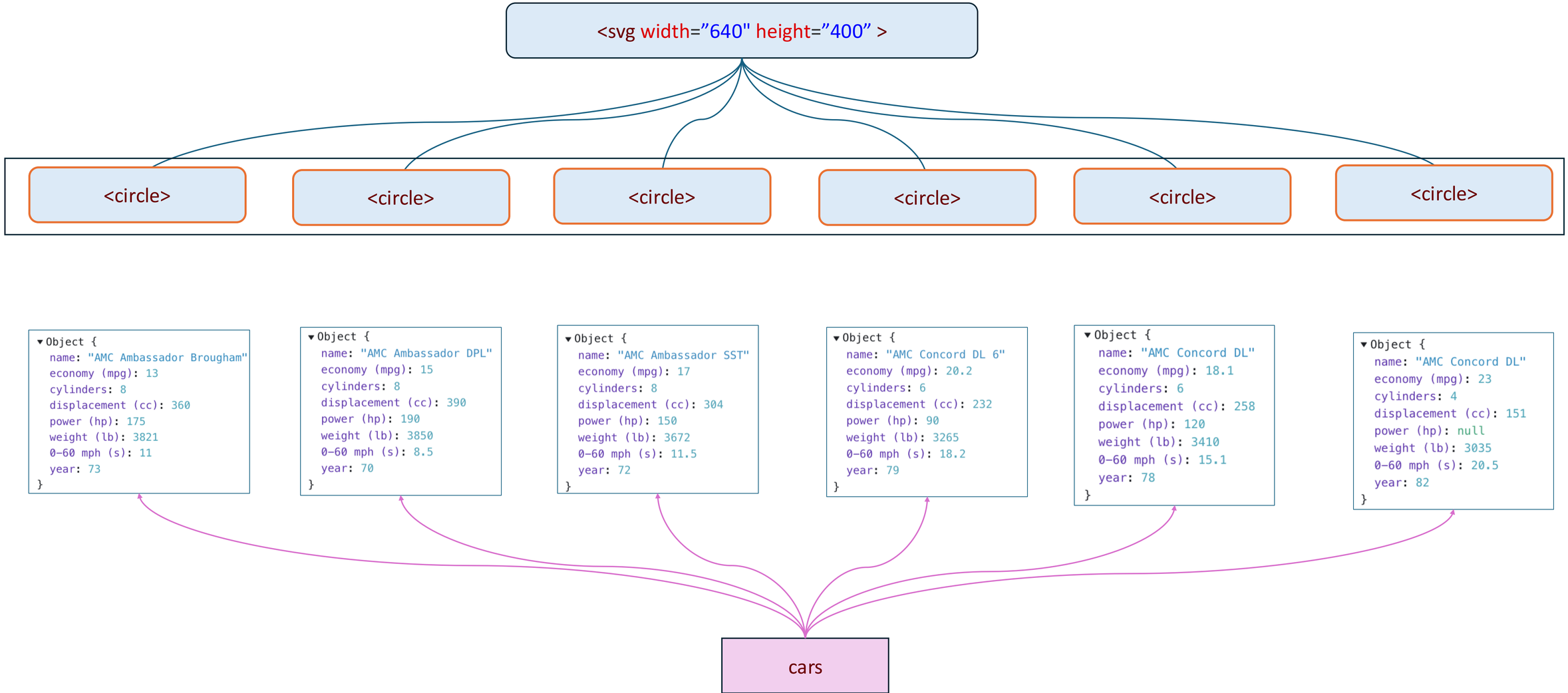
<svg width=”640" height=”400” >



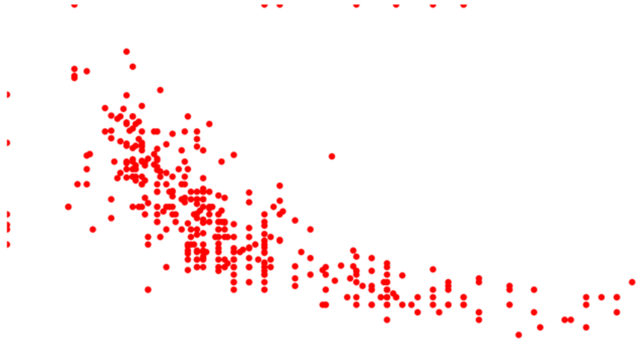
```
svg.selectAll('circle')
    .data(cars).enter()
```

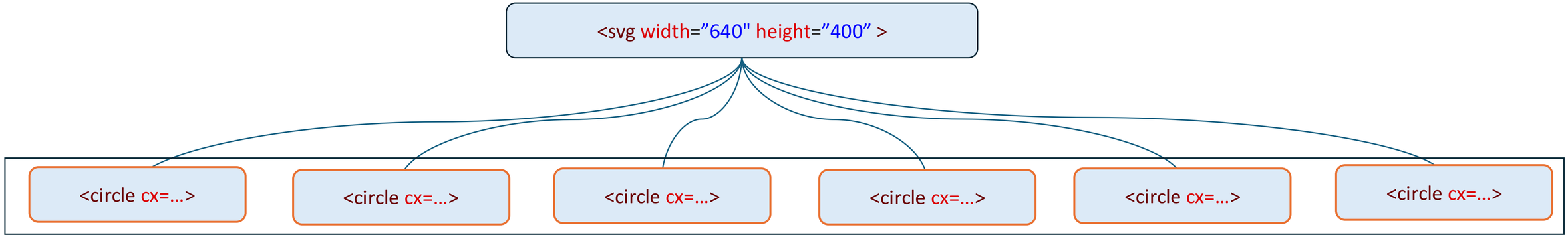


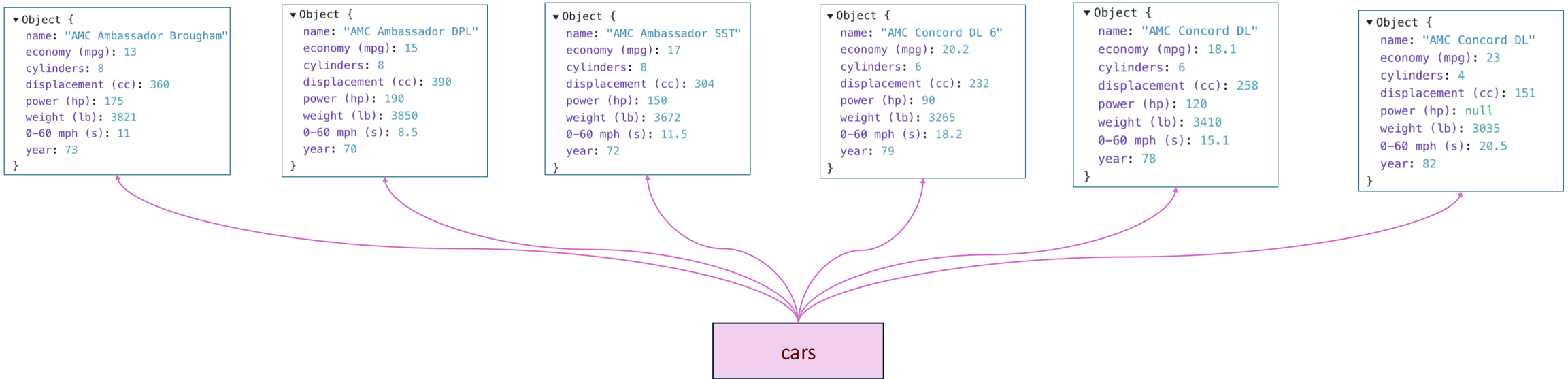
```
svg.selectAll('circle')
    .data(cars).enter()
    .append('circle')
```



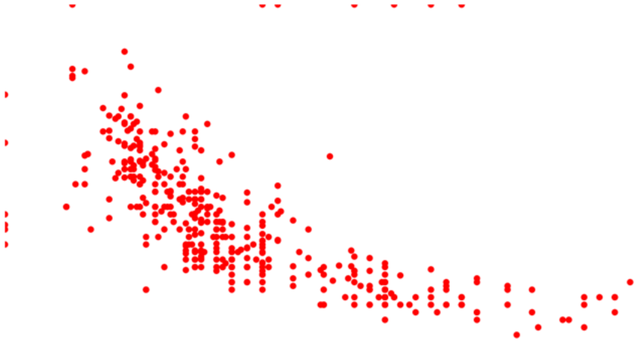
```
svg.selectAll('circle')
    .data(cars).enter()
    .append('circle')
    .attr("fill", "red")
    .attr("cx", (d) => x(d["power (hp)"]))
    .attr("cy", (d) => y(d["economy (mpg)"]))
    .attr("r", 3)
```

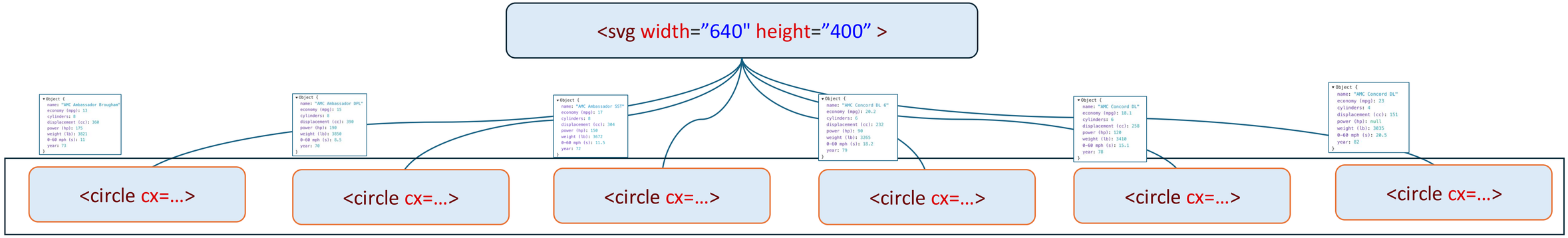


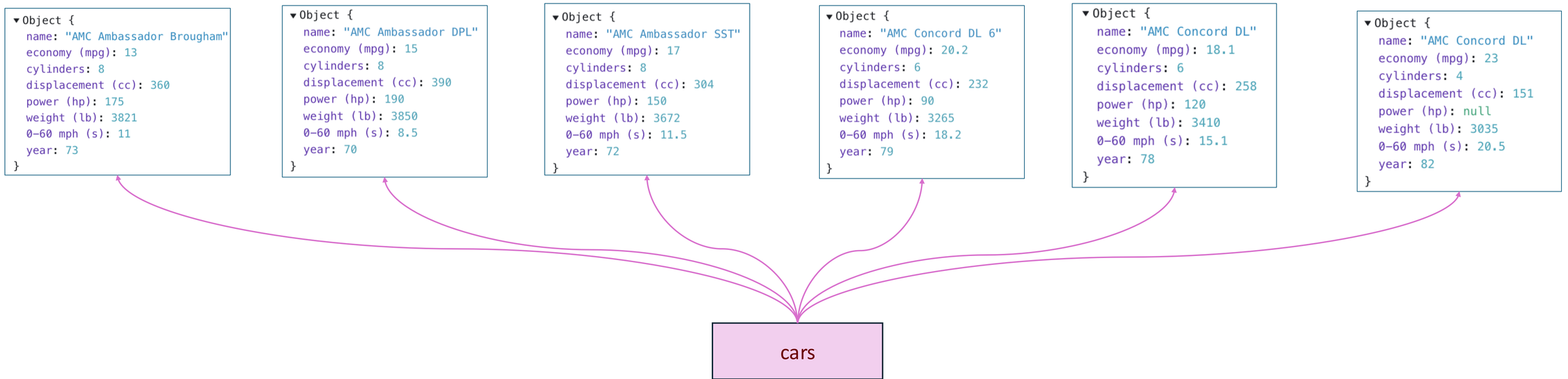




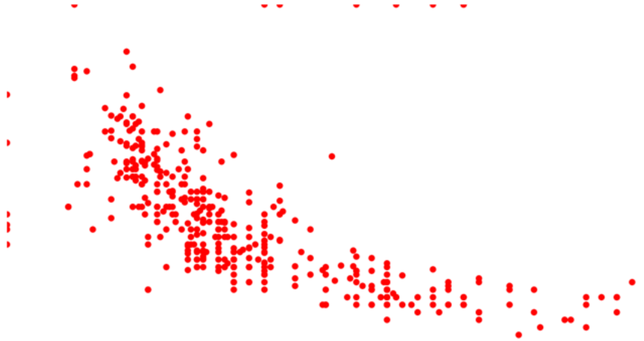
```
svg.selectAll('circle')
    .data(cars).enter()
    .append('circle')
    .attr("fill", "red")
    .attr("cx", (d) => x(d["power (hp)"]))
    .attr("cy", (d) => y(d["economy (mpg)"]))
    .attr("r", 3)
```

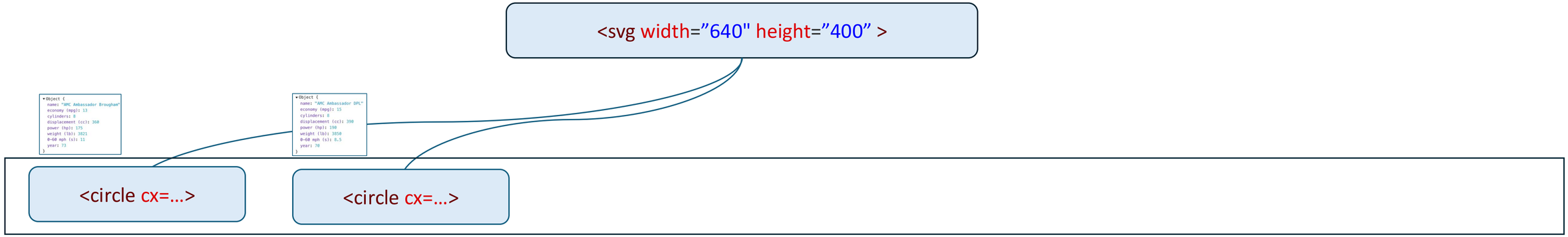


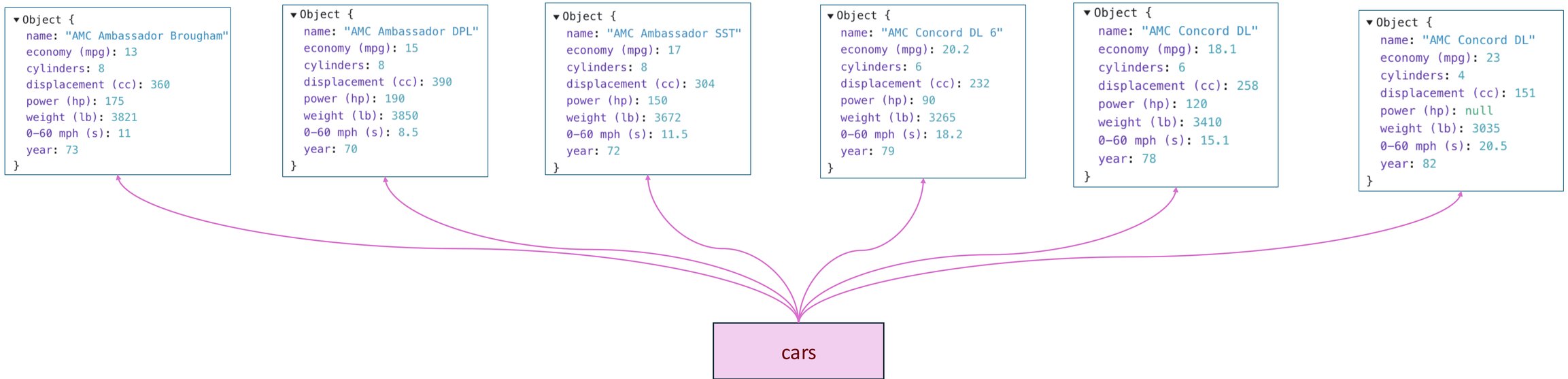




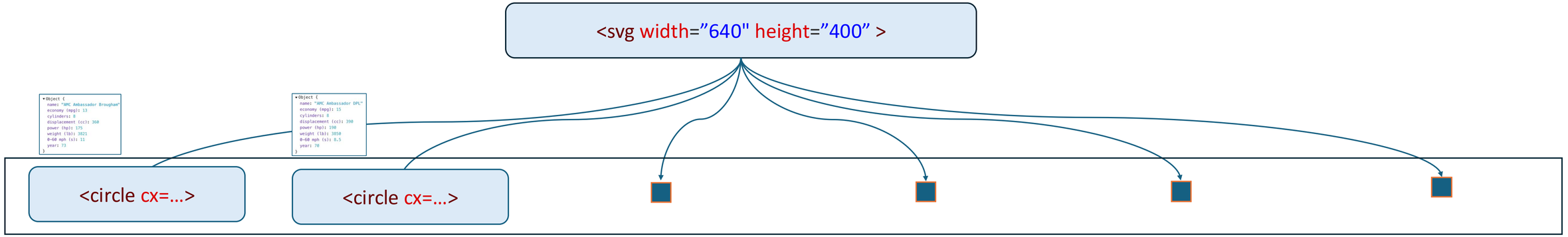
```
svg.selectAll('circle')
    .data(cars).enter()
    .append('circle')
    .attr("fill", "red")
    .attr("cx", (d) => x(d["power (hp)"]))
    .attr("cy", (d) => y(d["economy (mpg)"]))
    .attr("r", 3)
```

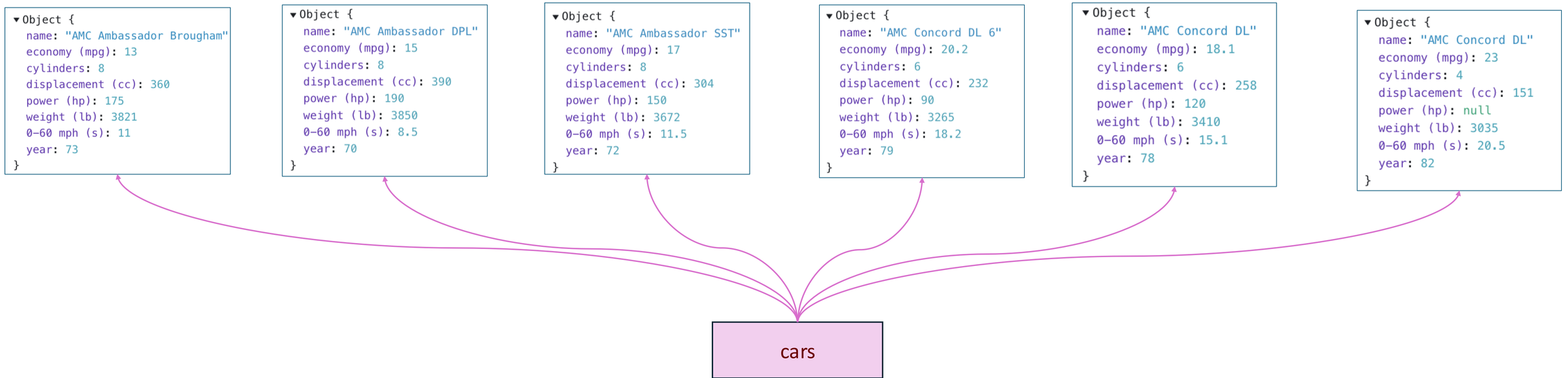






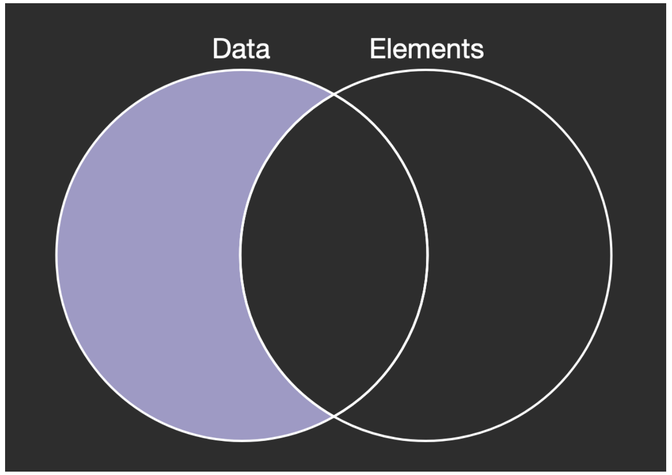
```
svg.selectAll('circle')
    .data(cars).enter()
    .append('circle')
    .attr("fill", "red")
    .attr("cx", (d) => x(d["power (hp)"]))
    .attr("cy", (d) => y(d["economy (mpg)"]))
    .attr("r", 3)
```

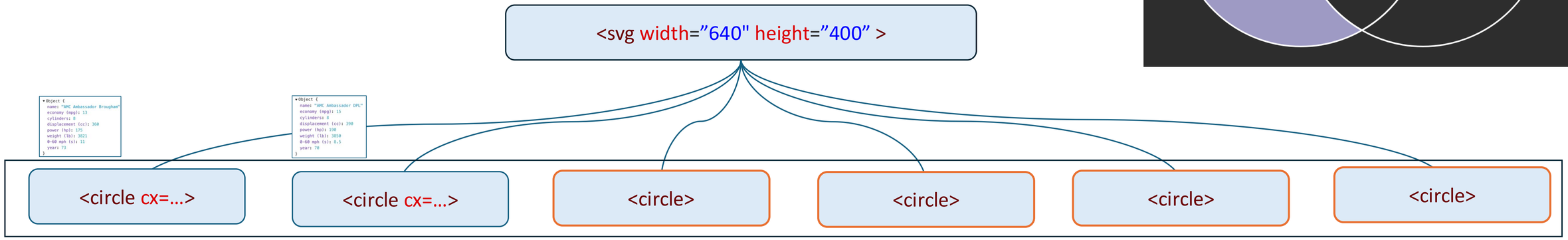


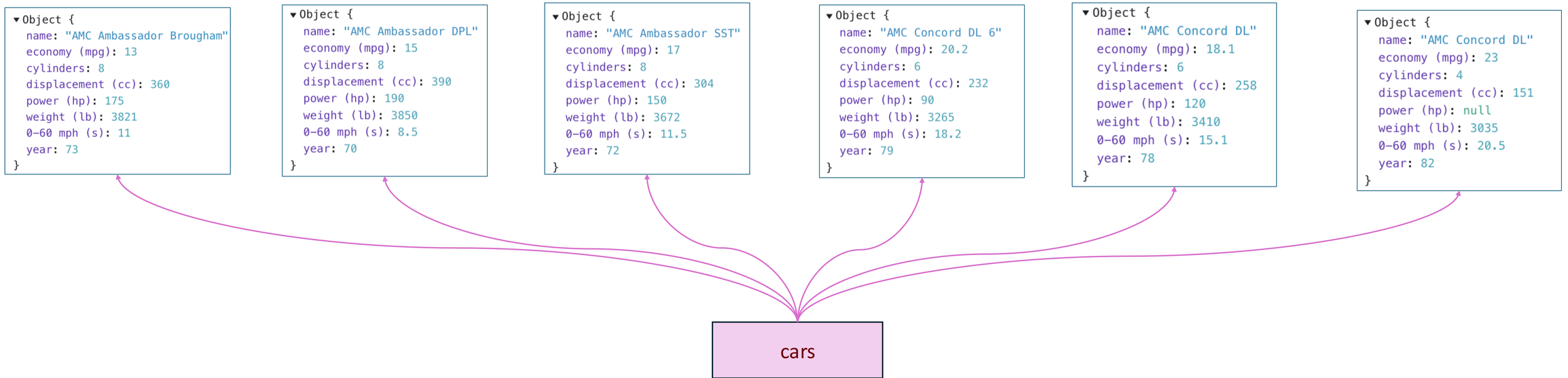


```
svg.selectAll('circle')
    .data(cars).enter()
    .append('circle')
    .attr("fill", "red")
    .attr("cx", (d) => x(d["power (hp)"]))
    .attr("cy", (d) => y(d["economy (mpg)"]))
    .attr("r", 3)
```

Enter:

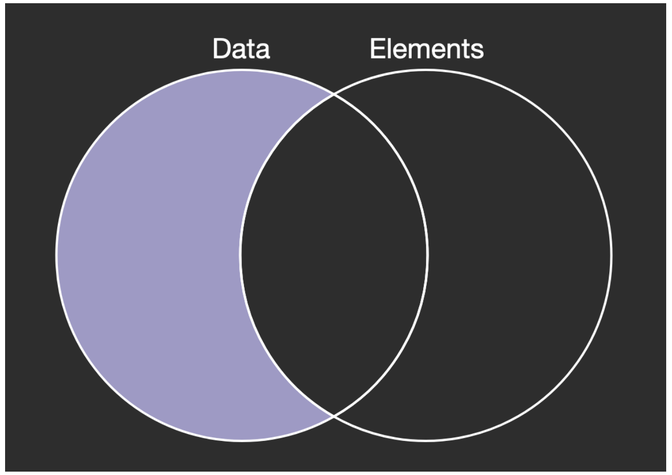


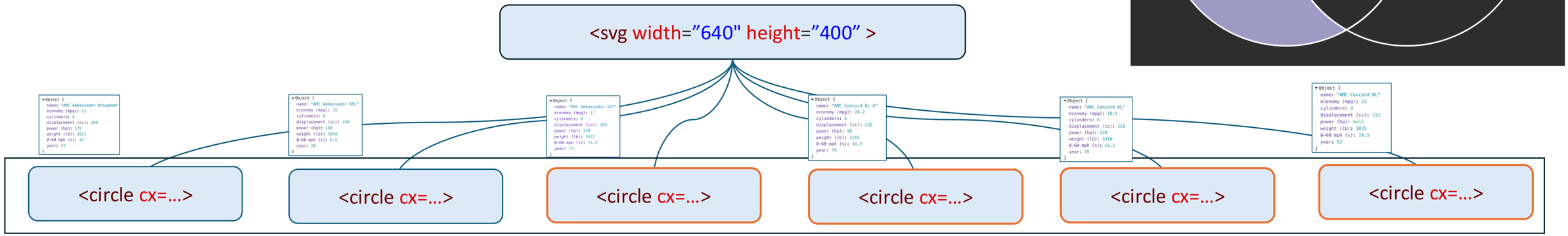


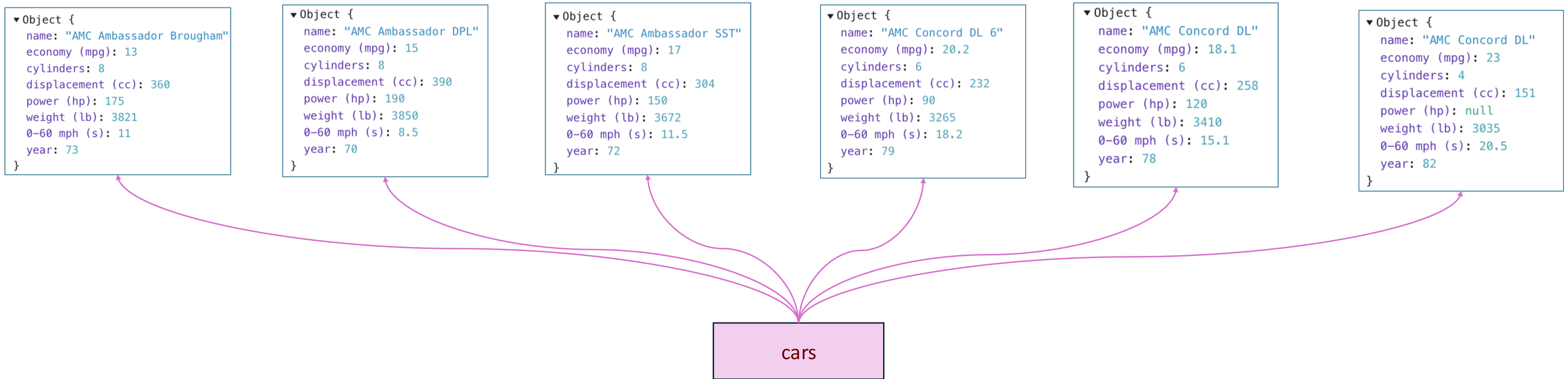


```
svg.selectAll('circle')
    .data(cars).enter()
    .append('circle')
    .attr("fill", "red")
    .attr("cx", (d) => x(d["power (hp)"]))
    .attr("cy", (d) => y(d["economy (mpg)"]))
    .attr("r", 3)
```

Enter:

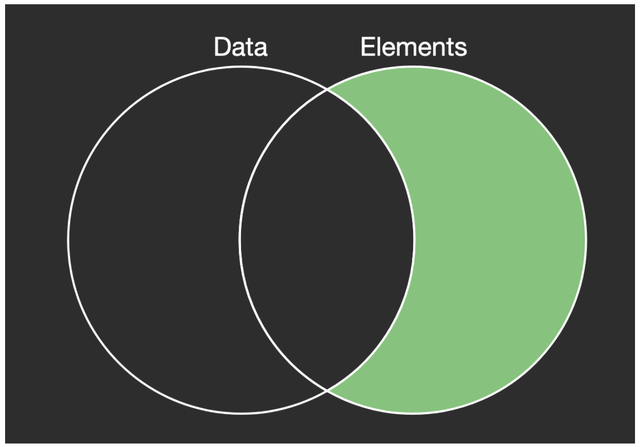


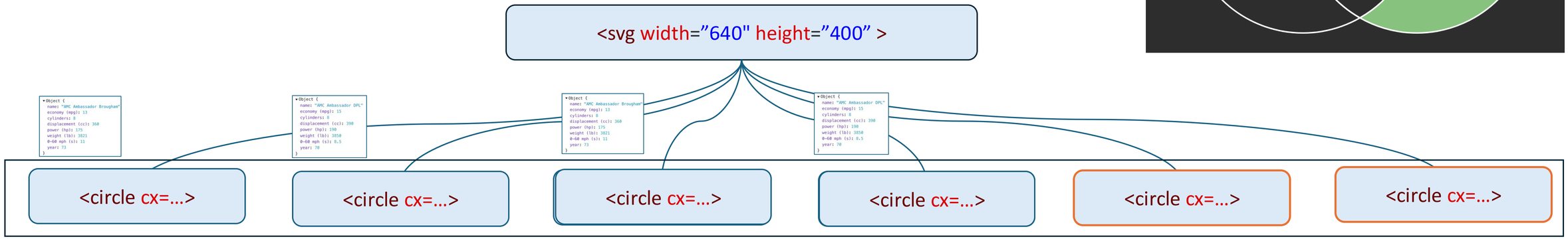


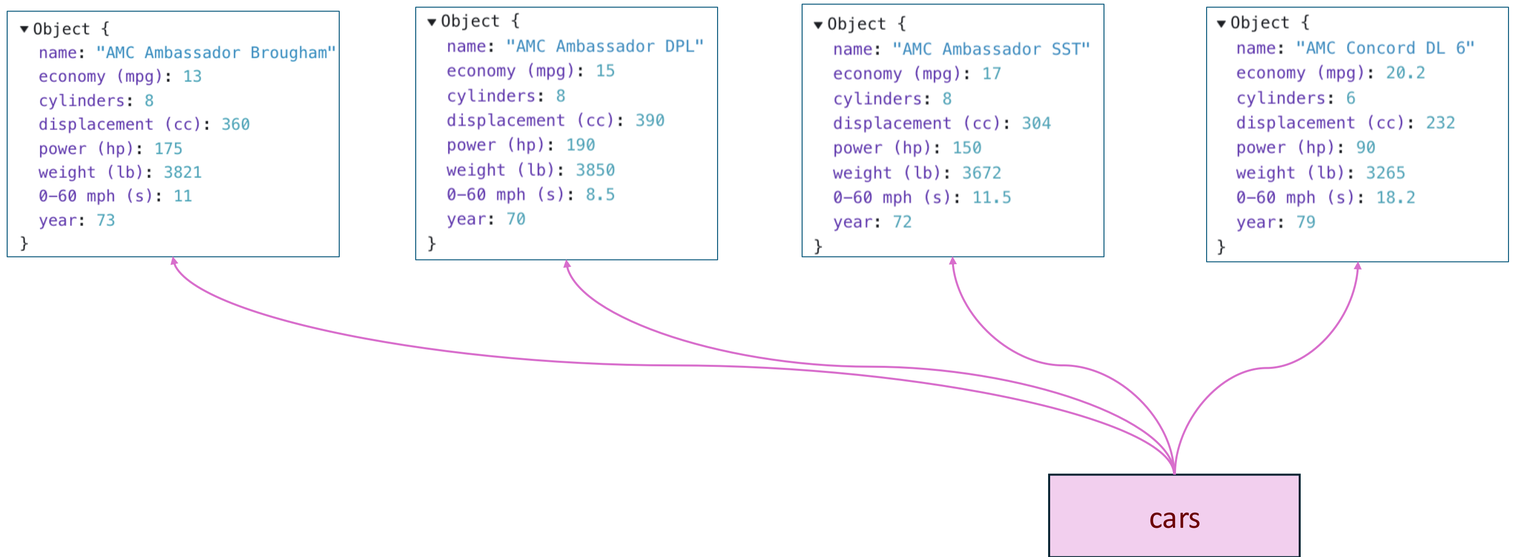


```
svg.selectAll('circle')
    .data(cars).exit()
    .remove()
```

Exit:

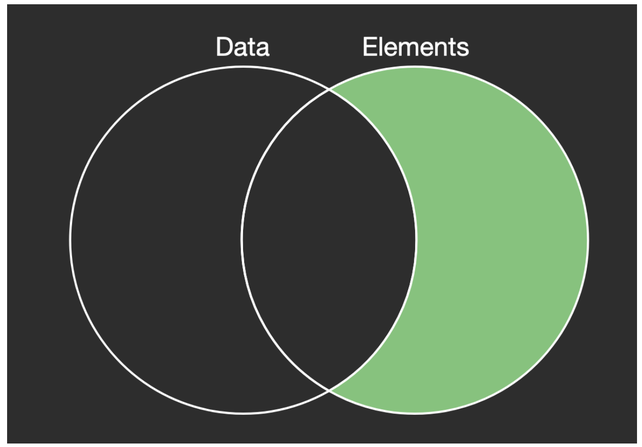


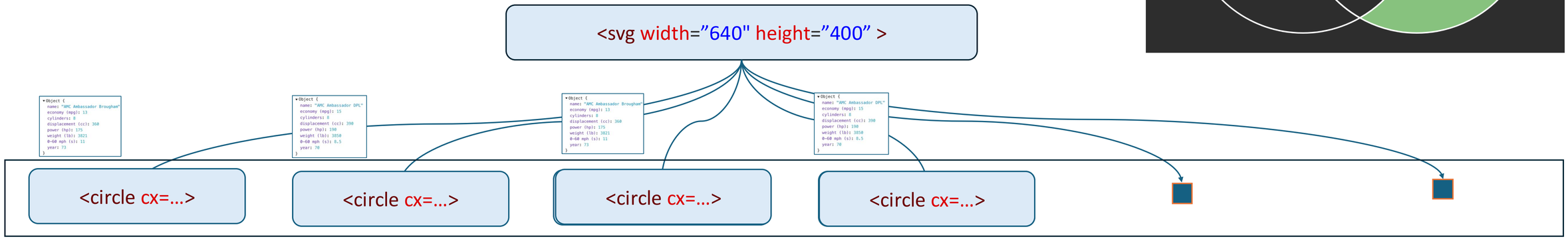


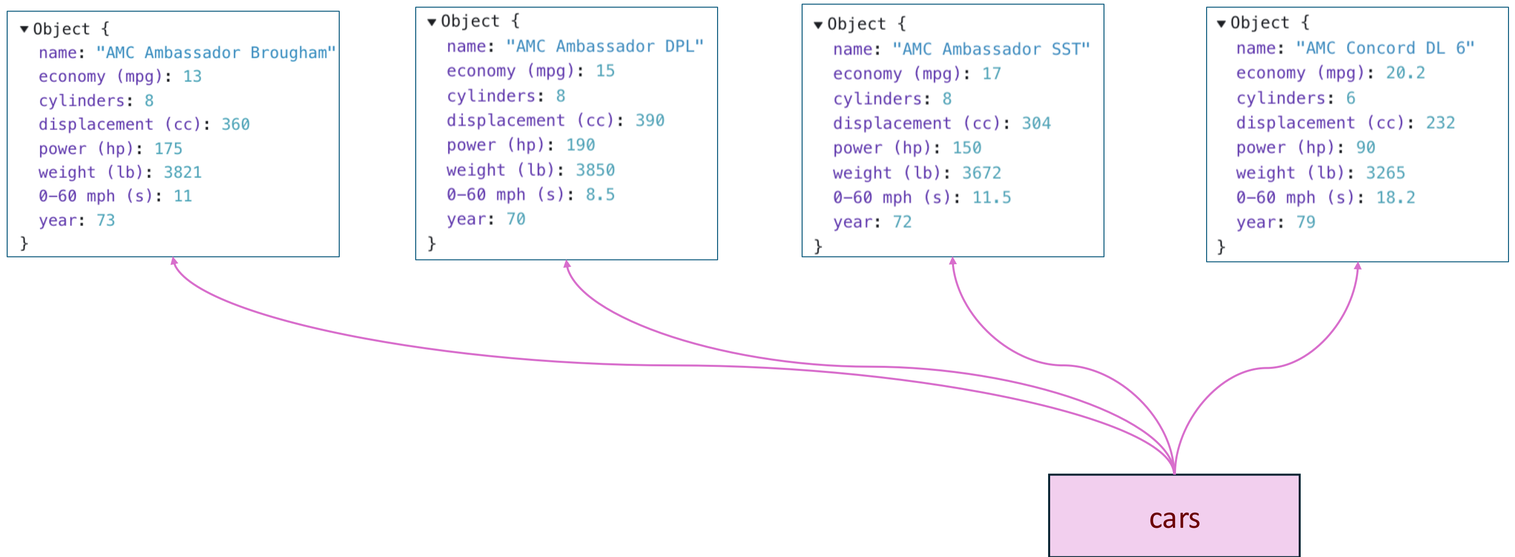


```
svg.selectAll('circle')
    .data(cars).exit()
    .remove()
```

Exit:

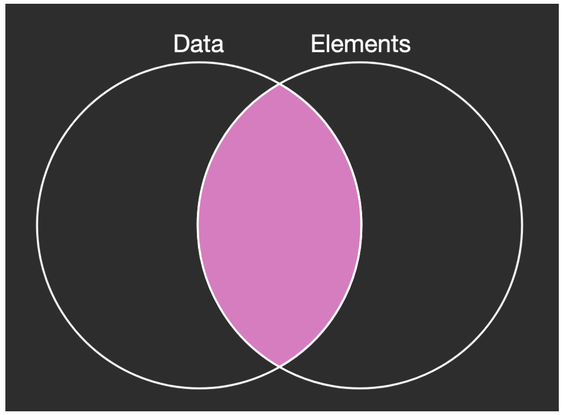


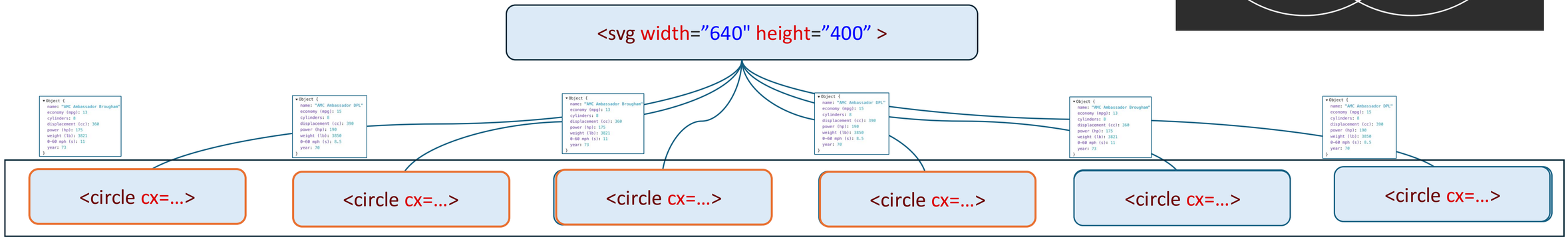


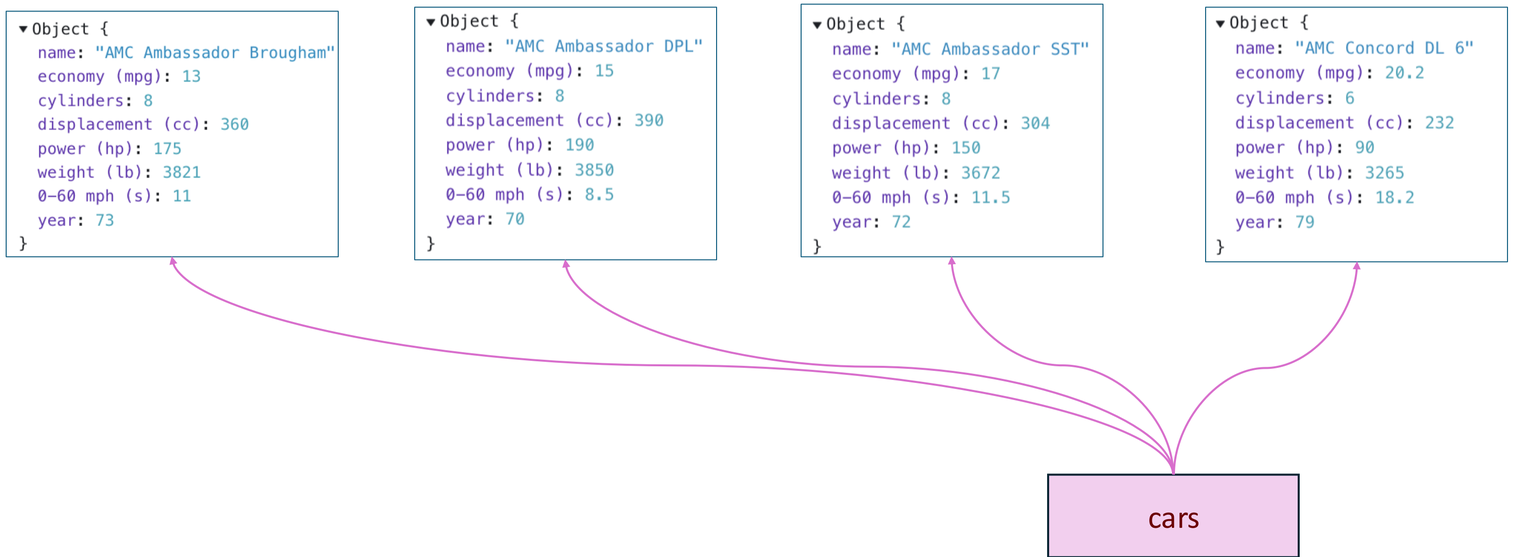


```
svg.selectAll('circle')
    .data(cars)
    .attr("fill", "red")
    .attr("cx", (d) => x(d["power (hp)"]))
    .attr("cy", (d) => y(d["economy (mpg)"]))
    .attr("r", 3)
```

Update:

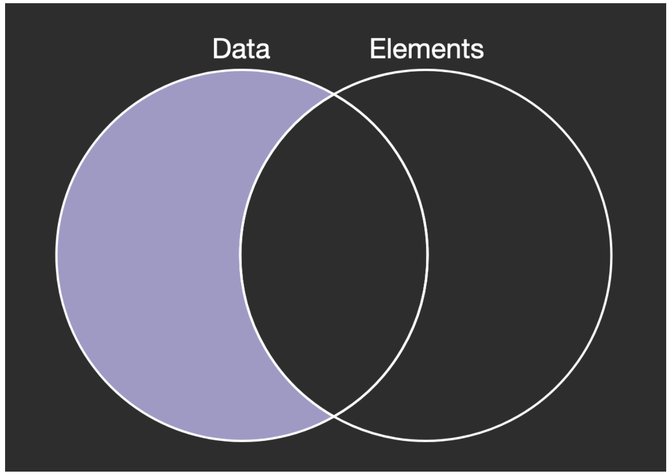


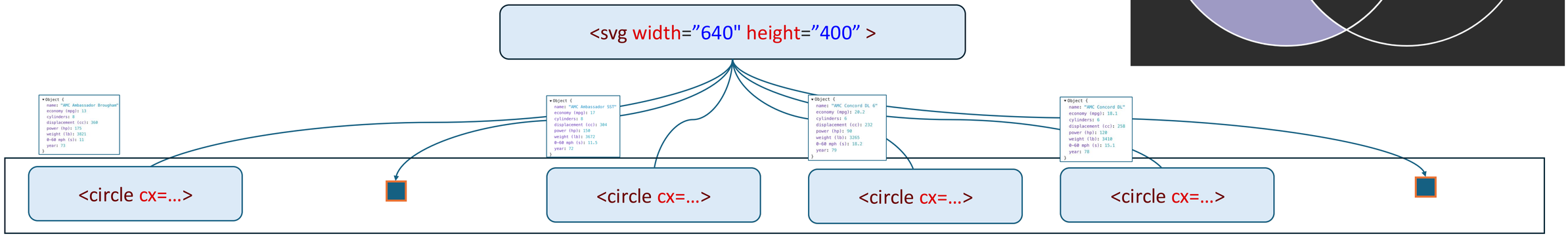


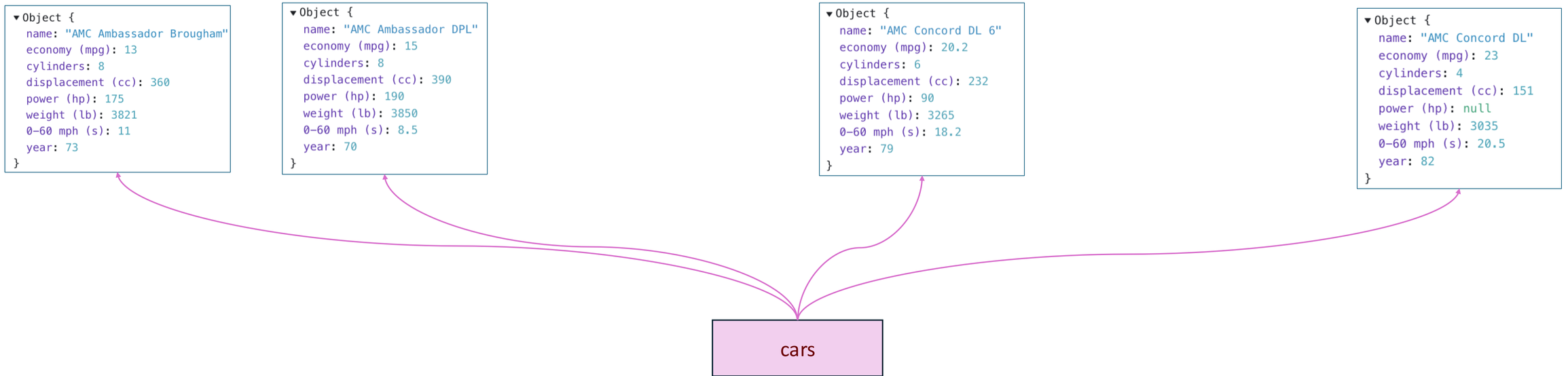


```
svg.selectAll('circle')
    .data(cars, d => d.name).enter()
    .append('circle')
    .attr("fill", "red")
```

Enter:

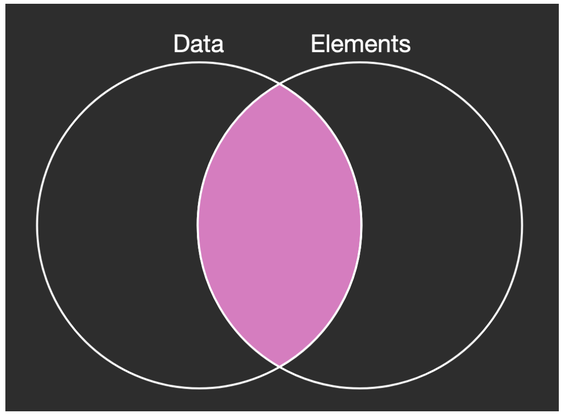


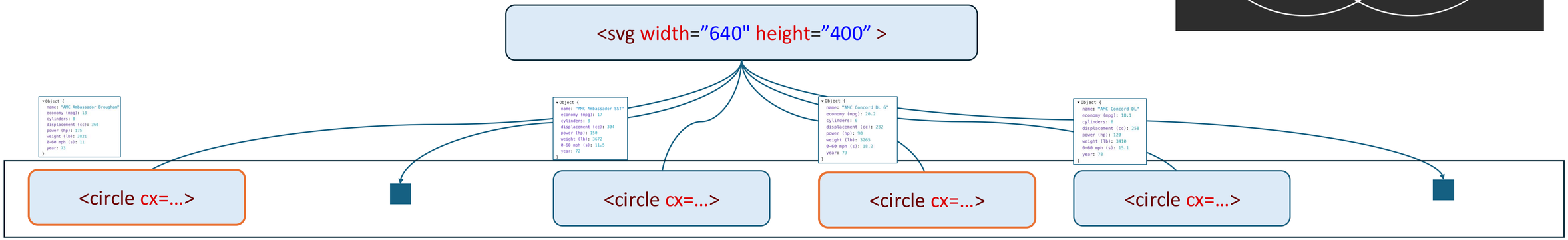


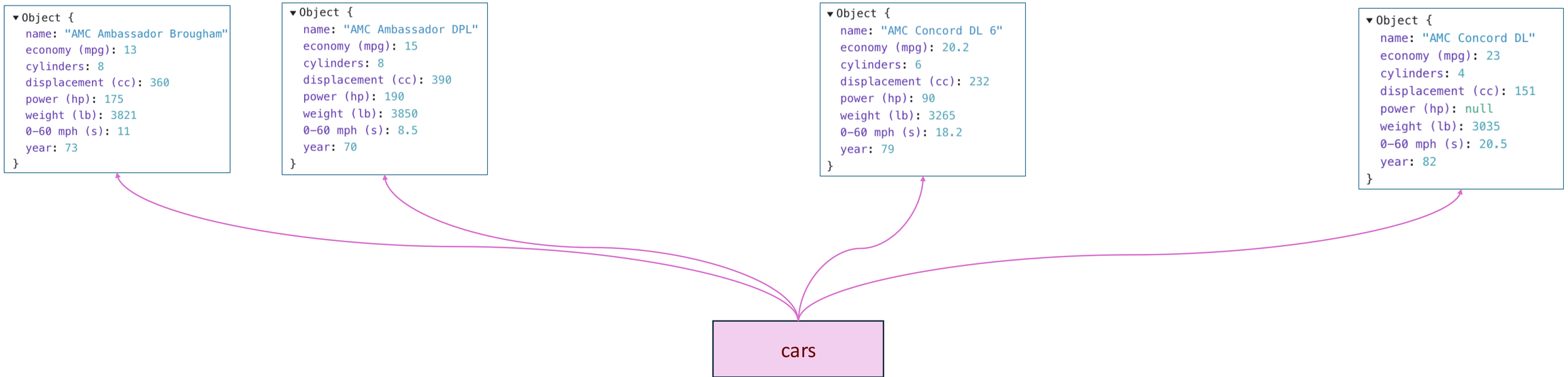


```
svg.selectAll('circle')
    .data(cars, d => d.name)
    .attr("fill", "red")
```

Update:

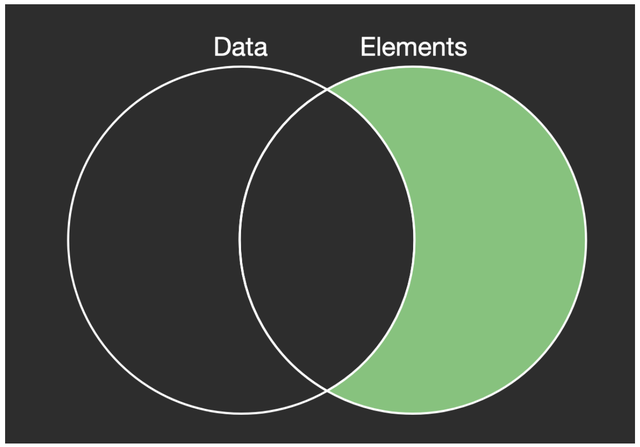


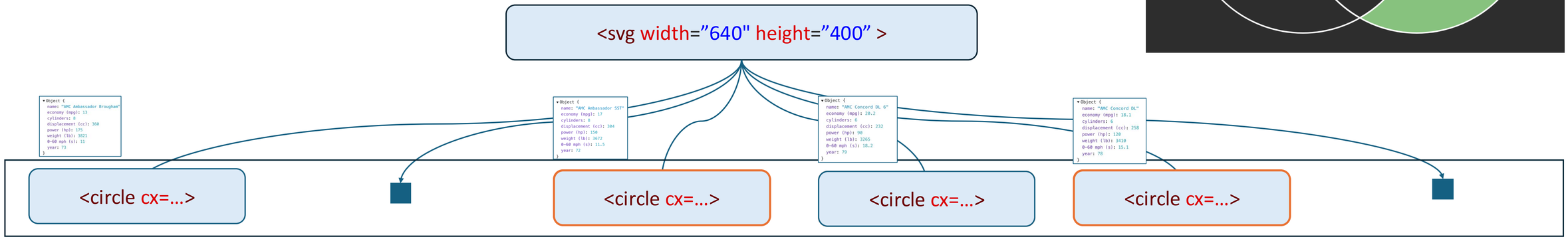


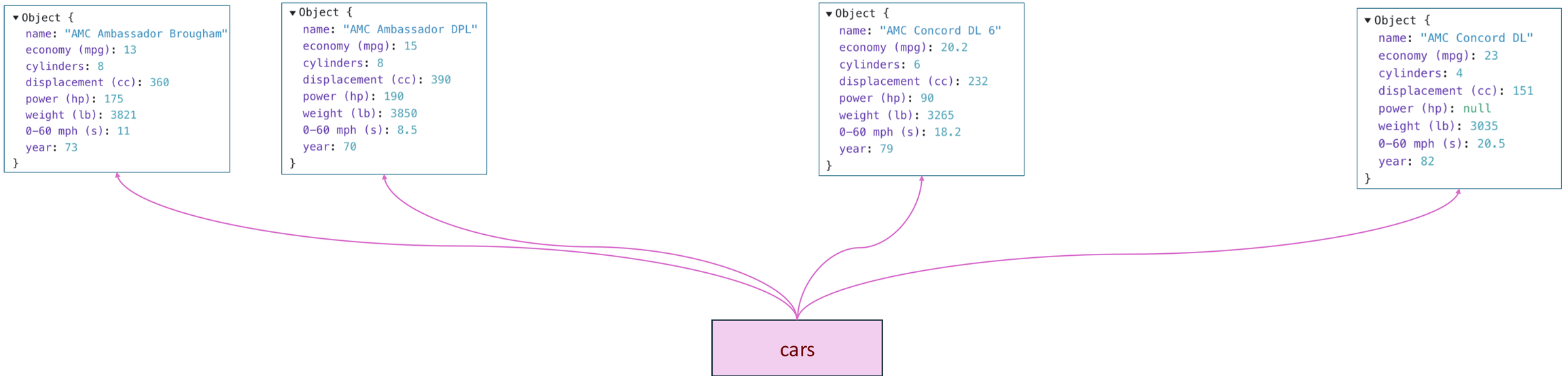


```
svg.selectAll('circle')
    .data(cars, d => d.name)
    .exit().remove()
```

Exit:

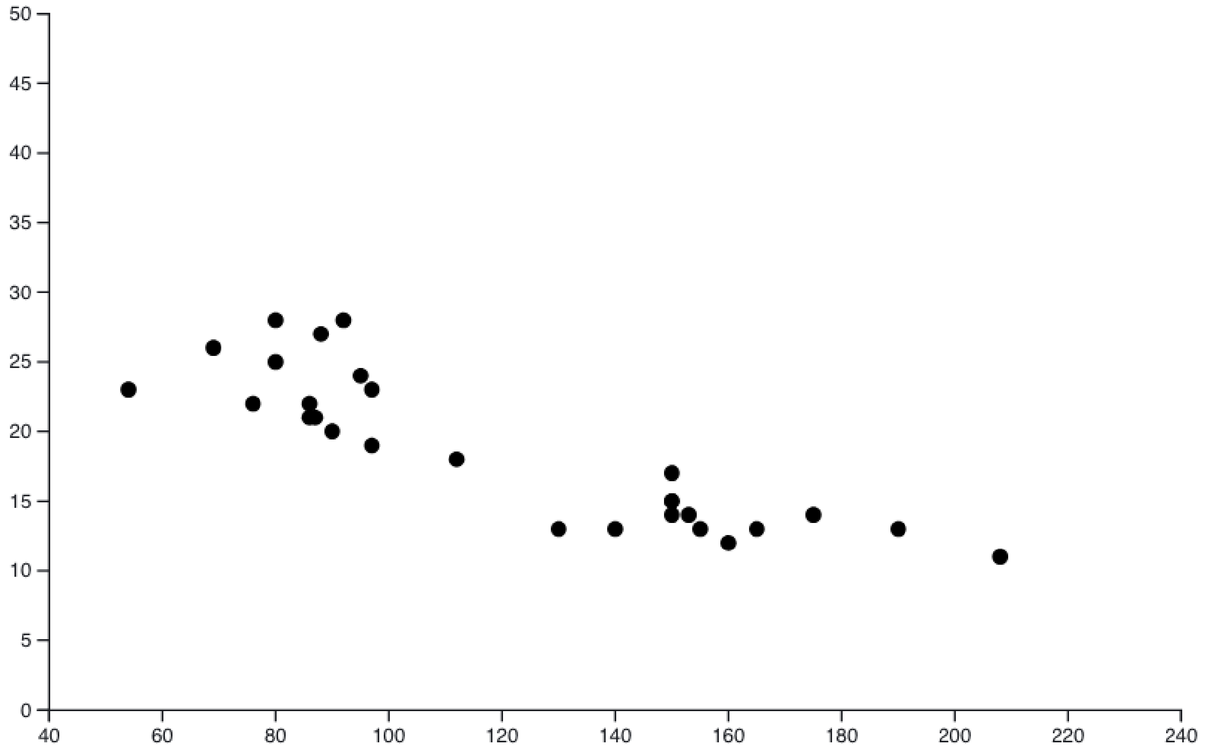




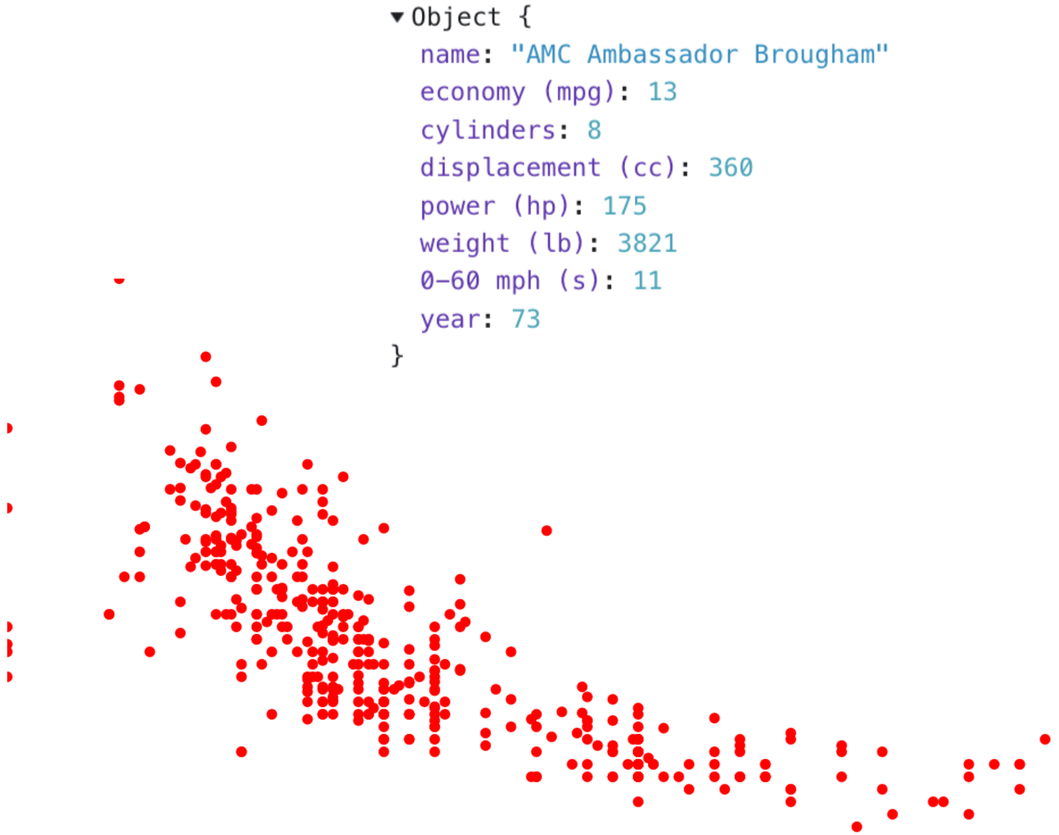


Example: Cars

```
const svg = d3.select(chart);
const yearCars = cars.filter((d) => d.year == year);

const selection = svg.selectAll('circle')
    .data(yearCars, (d) => d.name);

// Move remaining cars
selection.transition().duration(1000)
    .attr("cx", (d) => x(d["power (hp)"]))
    .attr("cy", (d) => y(d["economy (mpg)"]))

// Grow entering cars
selection.enter().append('circle')
    .filter((d) => d["power (hp)"] > 0 && d["economy (mpg)"] > 0)
    .attr("cx", (d) => x(d["power (hp)"]))
    .attr("cy", (d) => y(d["economy (mpg)"]))
    .transition().duration(1000)
    .attr("r", 4);

// Shrink and remove exiting cars
selection.exit()
  .transition().duration(1000)
  .attr("r", 0)
  .remove();
```



```
svg.selectAll('circle')
    .data(cars)
    .join('circle')
    .attr("fill", "red")
    .attr("cx", (d) => x(d["power (hp)"]))
    .attr("cy", (d) => y(d["economy (mpg)"]))
    .attr("r", 3)
```

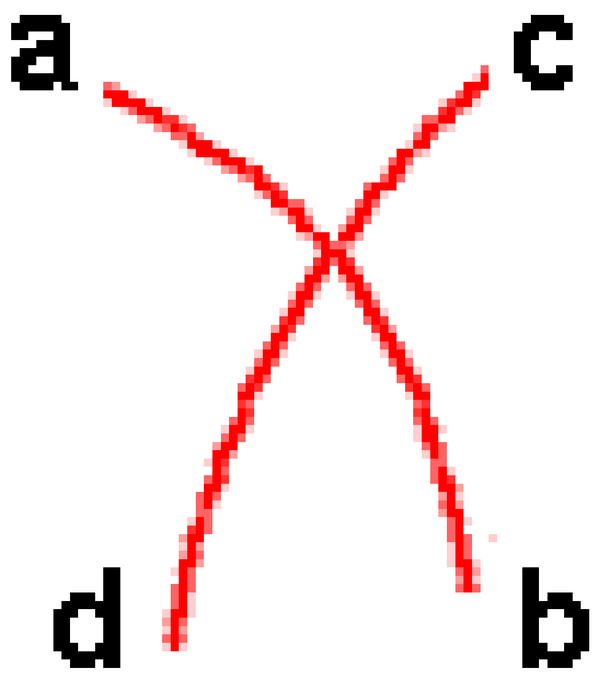



```
marks: [
  Plot.dot(cars, {x: "power (hp)",
                  y: "economy (mpg)",
                  r: 3,
                  fill: 'red'
                 }),
]
```

Animation perception

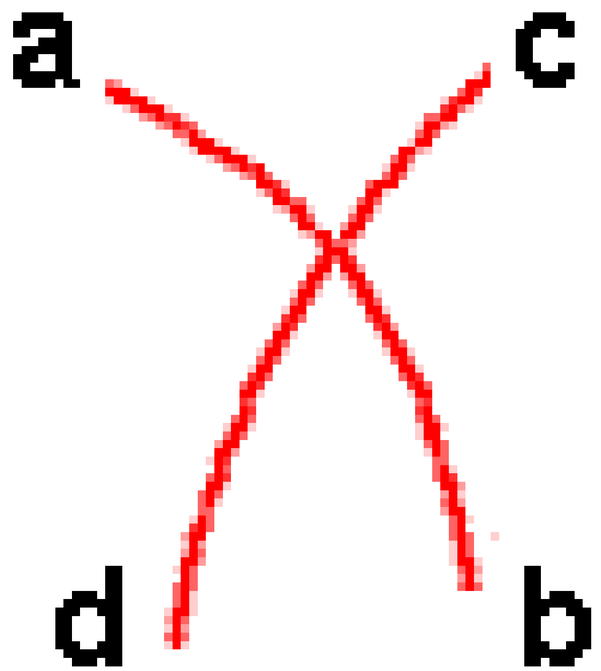
Gestalt Principles: Continuity

- Does a connect to b, c, or d?

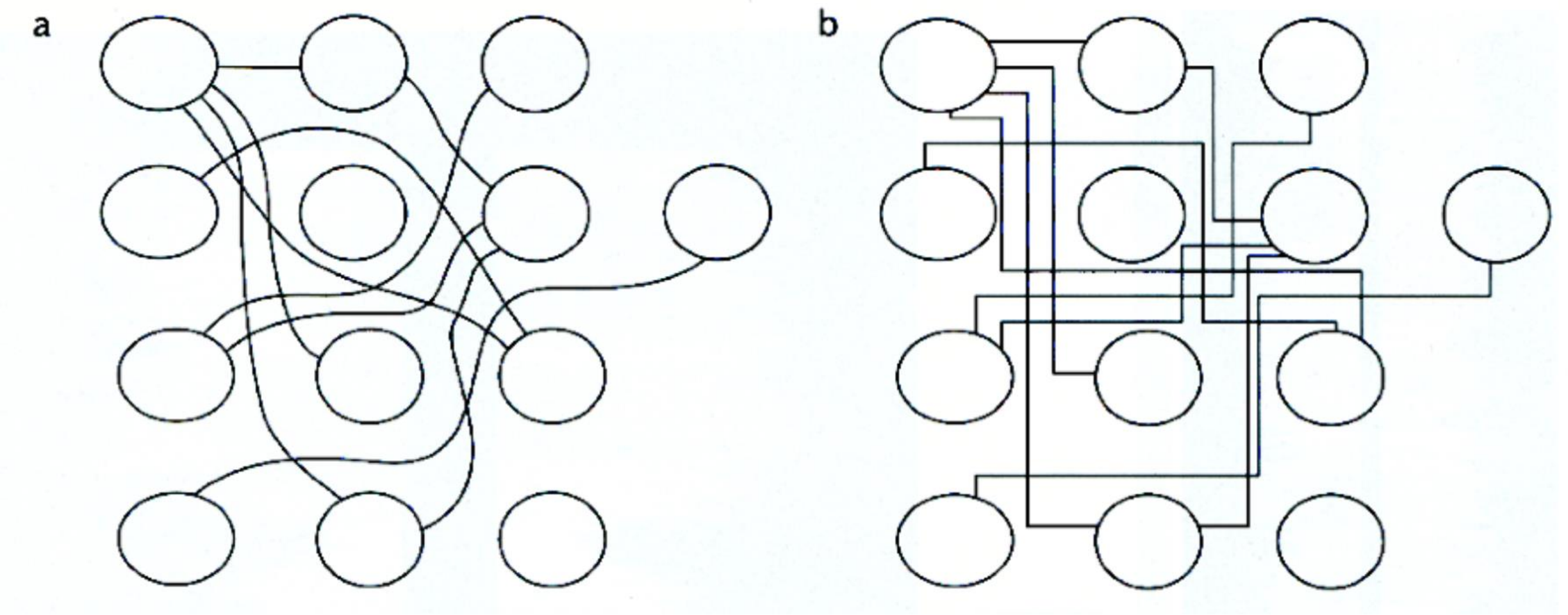


Gestalt Principles: Continuity

- Does a connect to b, c, or d?
  - Prefer **smooth** continuity

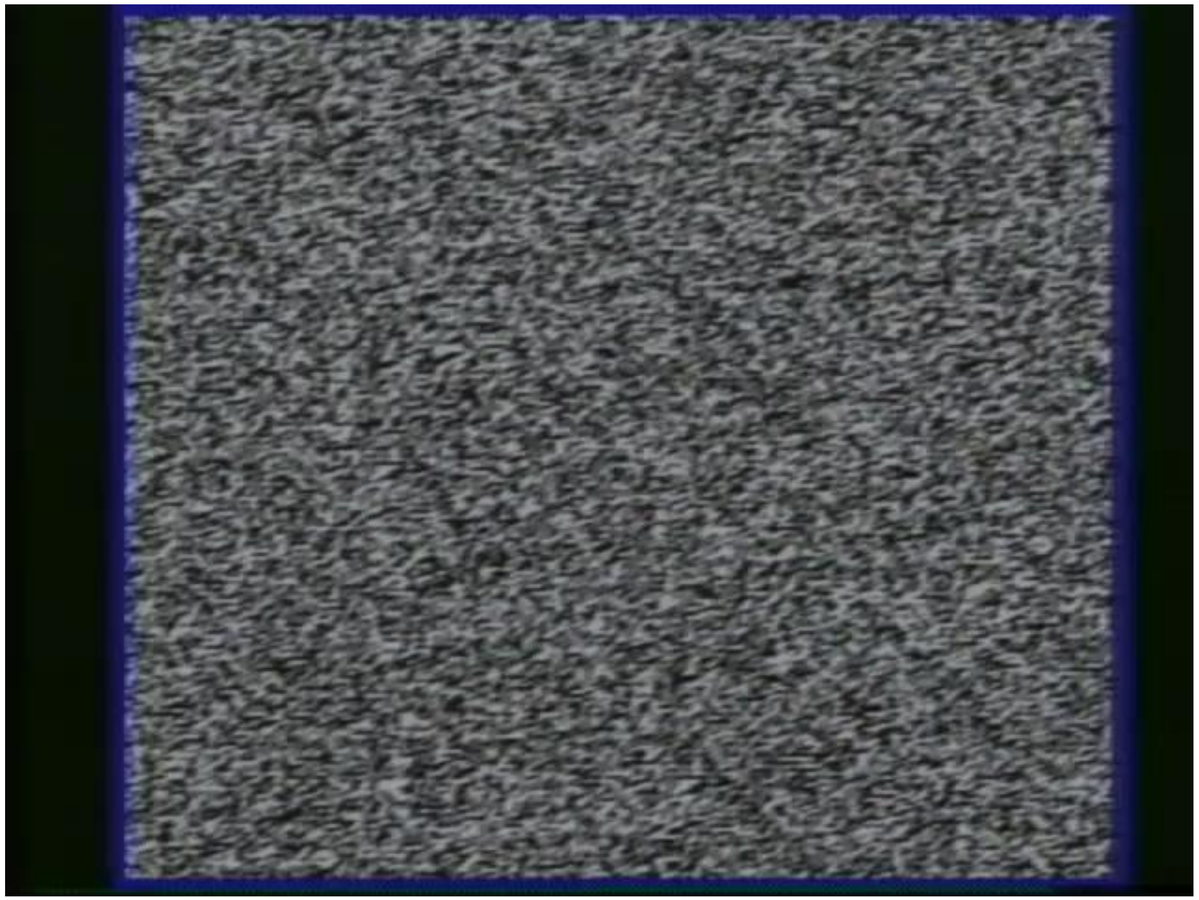


Gestalt Principles: Continuity



Gestalt Principles: animation

- Similar motions are perceived as a group!



Gestalt Principles: Continuity

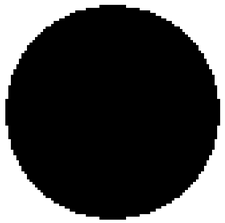
- Continuity is very important for motion!

Lack of continuity

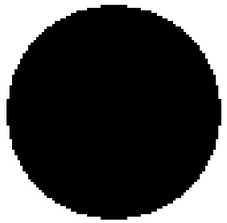
Presence of continuity

Perceiving animation

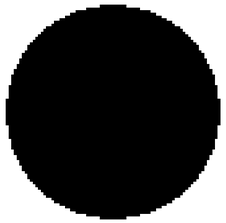
- Motion perceived at about 10 fps
  - Need at least 20-30 fps for *smoothness*



8fps (0.125... s)
0/8



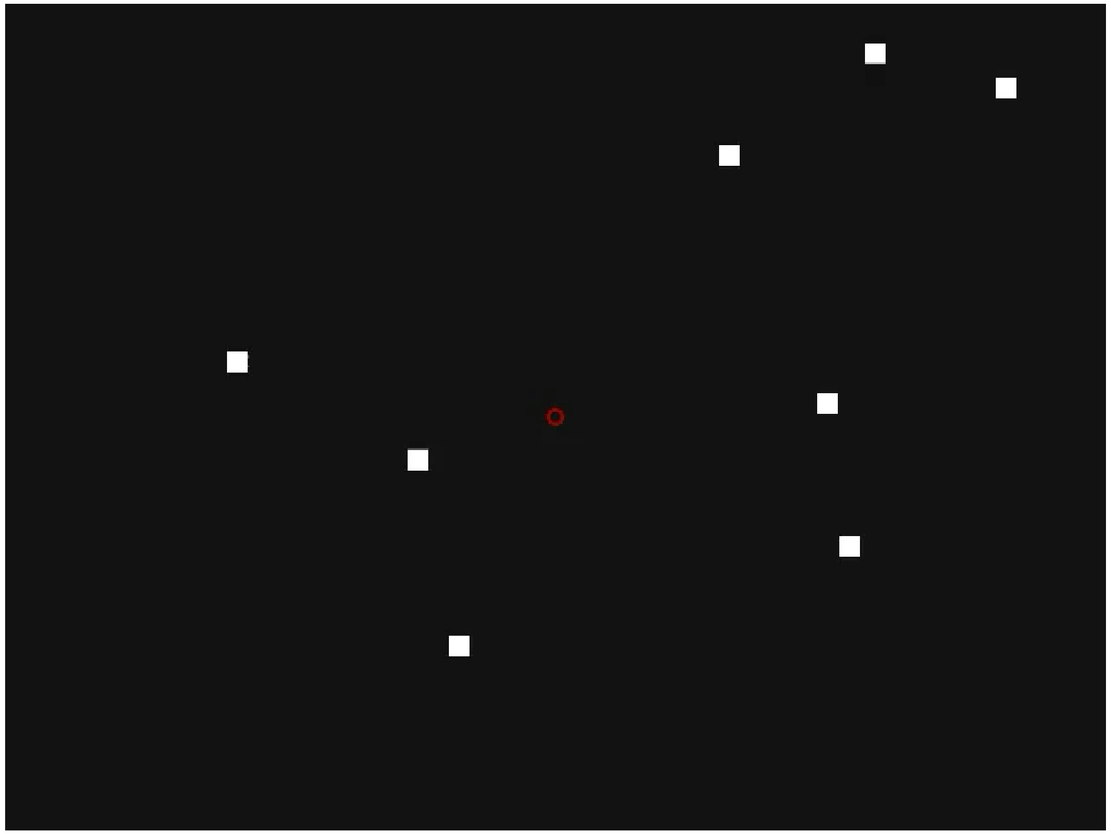
12fps (0.083... s)
0/12



24fps (0.041... s)
0/24

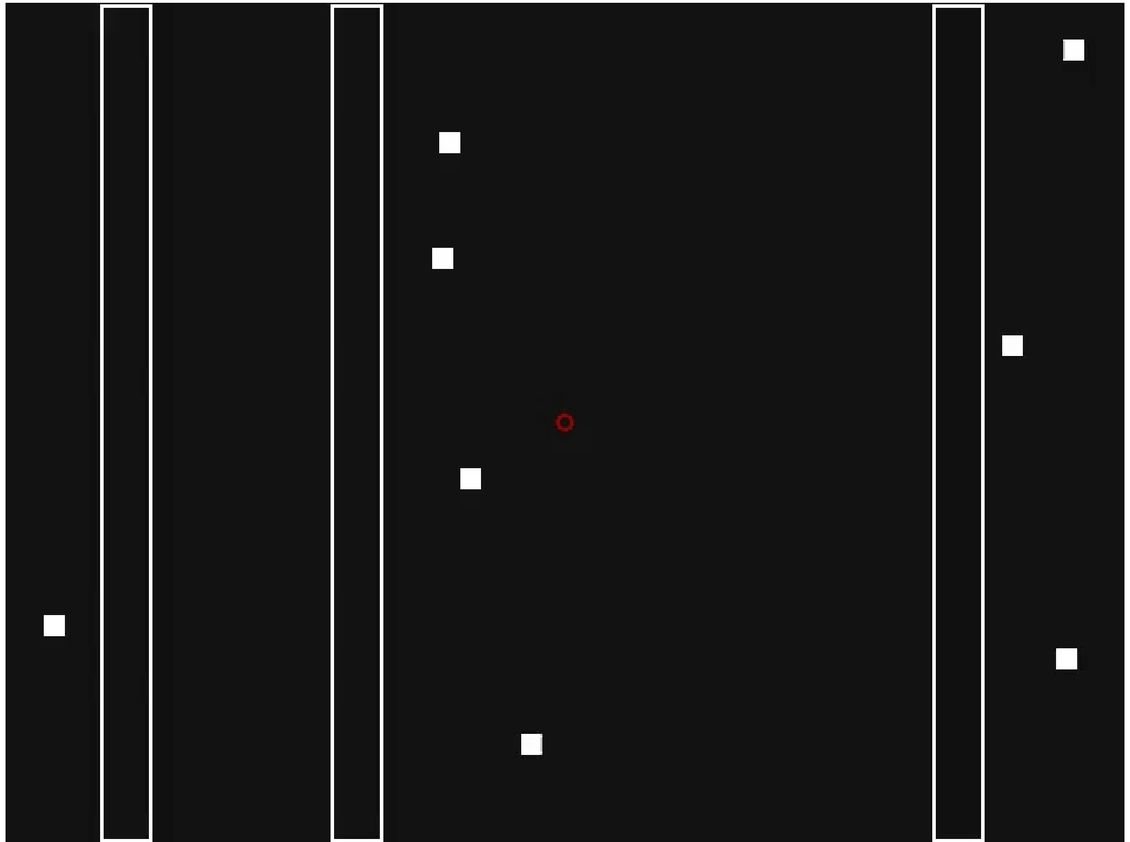
Tracking motion

- We are limited in our ability to track uncorrelated motion
  - About 4 objects is the limit



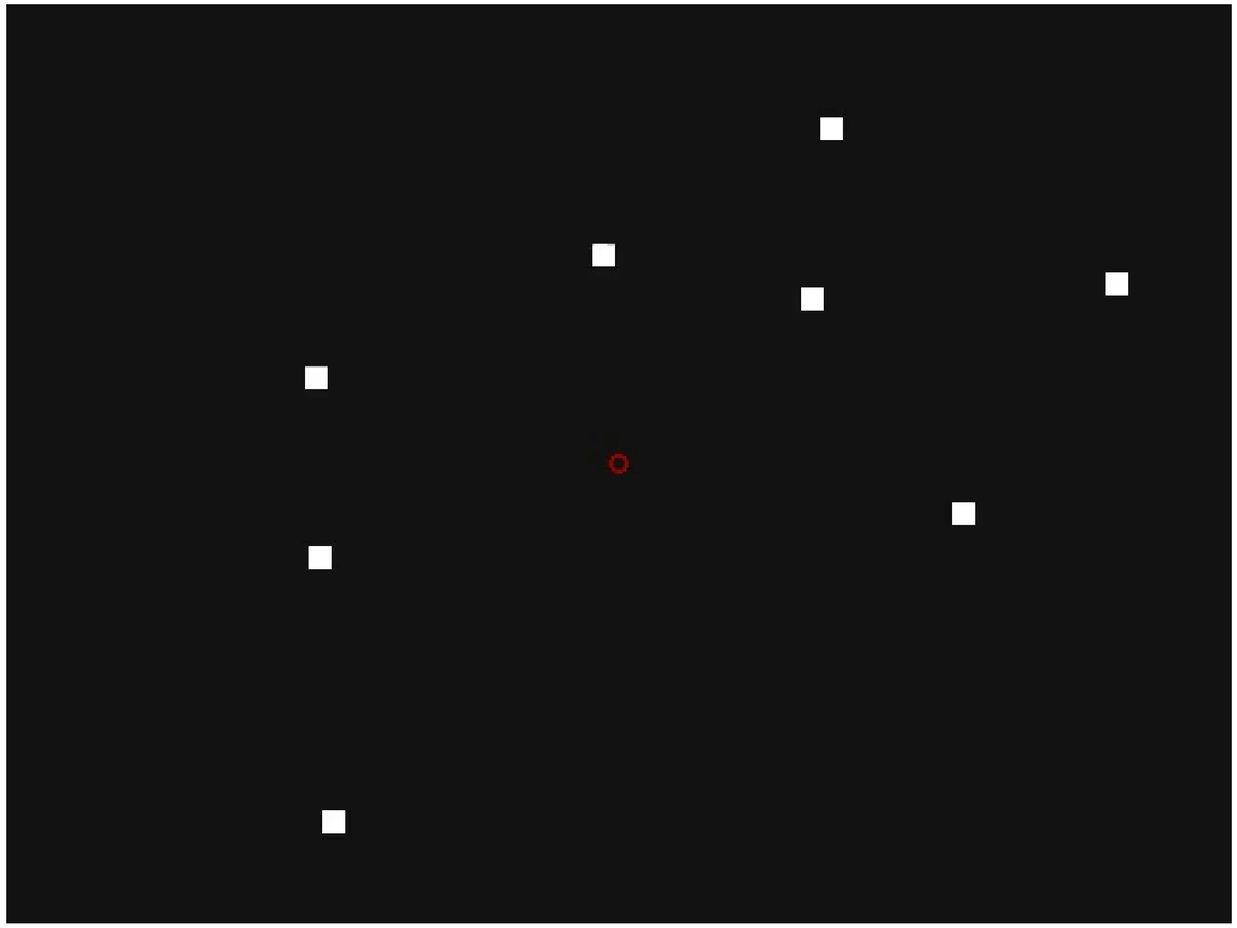
Tracking motion

- Can usually deal with occlusion



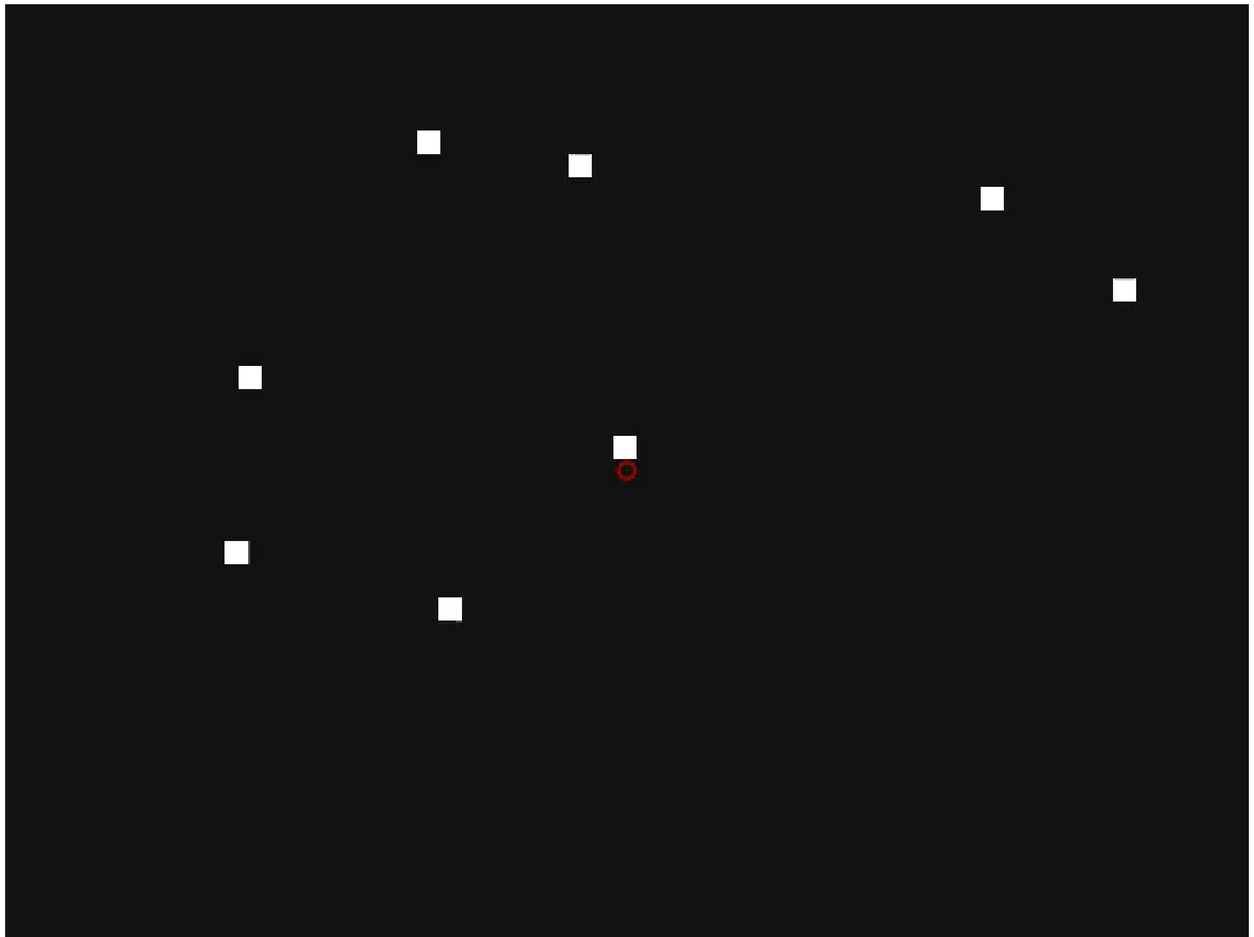
Tracking motion

- Even virtual occlusion



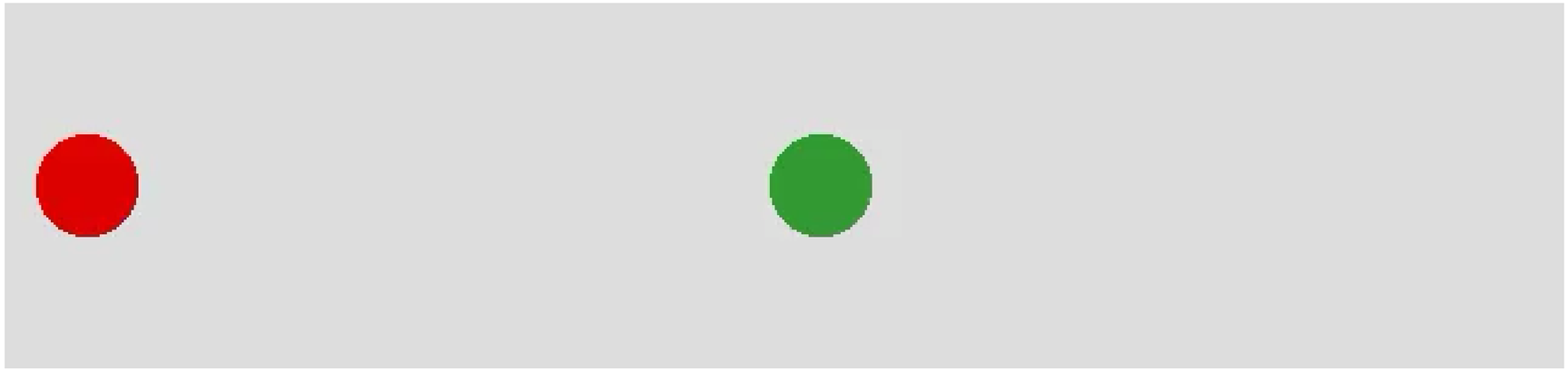
Tracking motion

- Other animations make this much harder



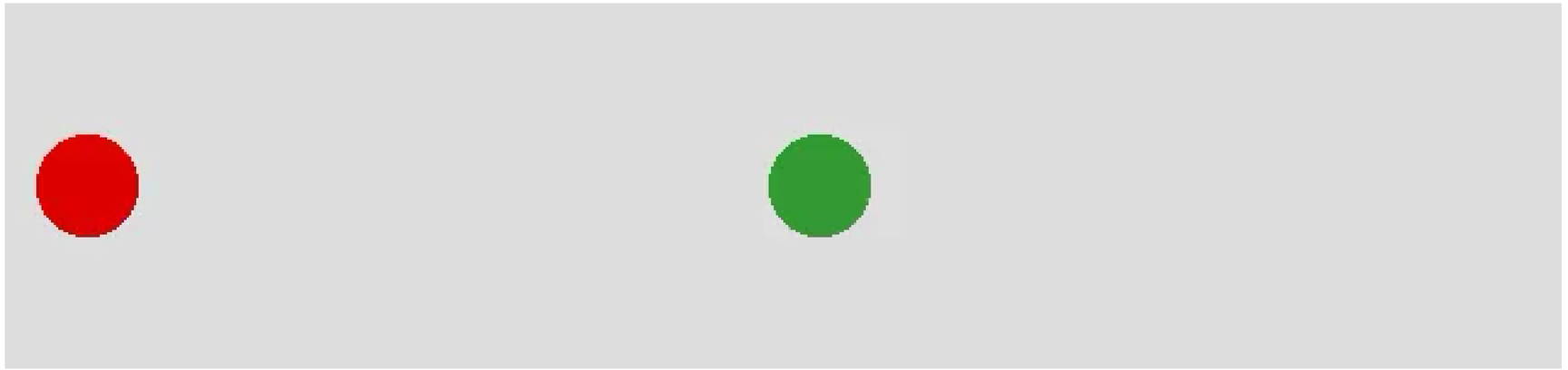
Causality

- Hard-wired to infer causality from motion

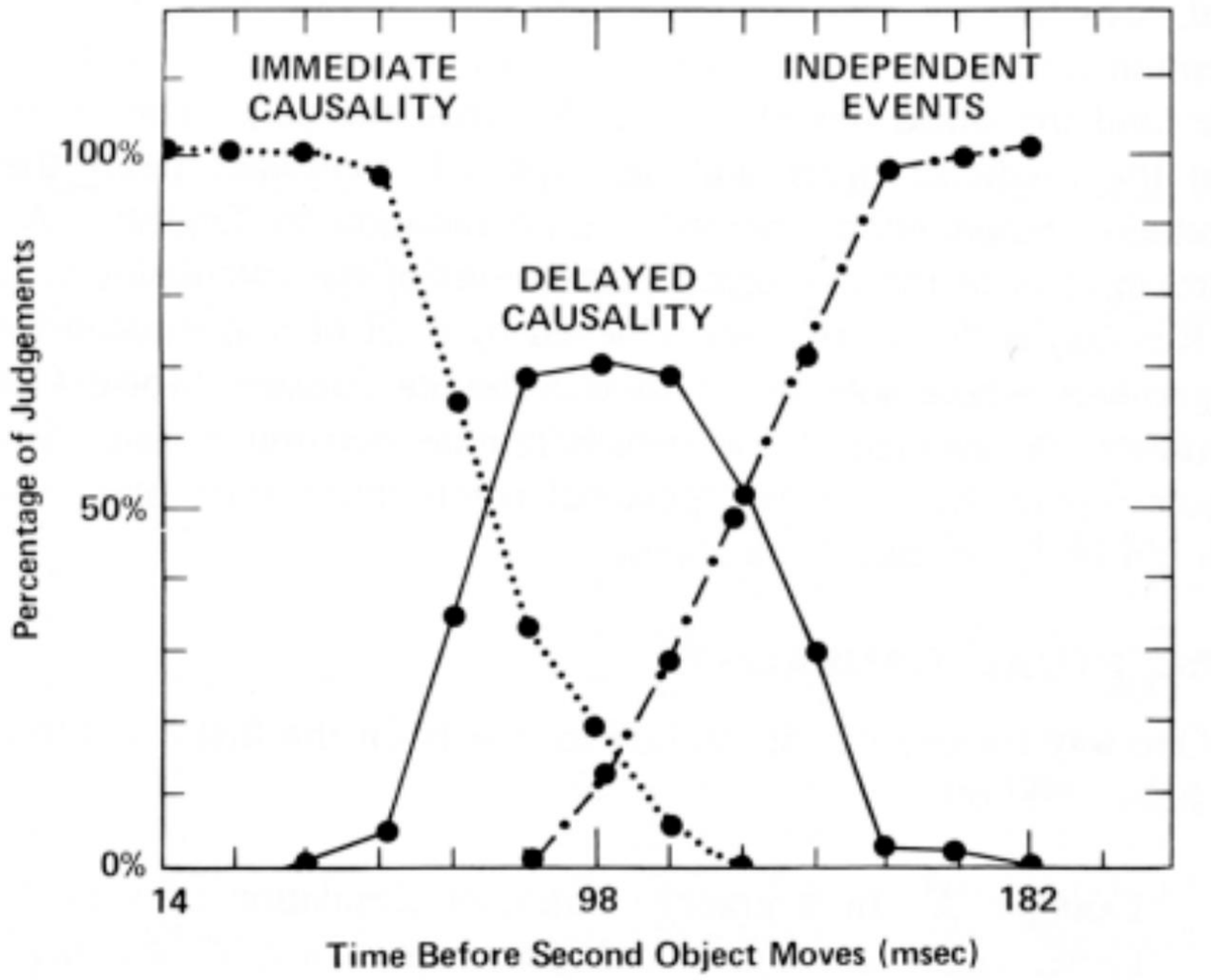


Causality

- Hard-wired to infer causality from motion



Causality


IMMEDIATE CAUSALITY
INDEPENDENT EVENTS
DELAYED CAUSALITY
100%
50%
0%
Percentage of Judgements
14
98
182
Time Before Second Object Moves (msec)

Storytelling

Animation	Helps?	Hurts?
Attention	direct attention	distraction
Constancy	change tracking	false relations
Causality	cause and effect	false agency
Engagement	increase interest	“chart junk”
Calibration		too slow: boring
		too fast: errors

Animation principles (Heer)

- Don’t change aesthetic mappings or scales if possible
- Respect correspondence, geometries should always represent the same observation
- Minimize occlusion
- Maintain vaild data graphics during transitions
- Use simple transitions
- Use staging for complex transitions
- Group similar transitions
- Different operators should have distinct animations

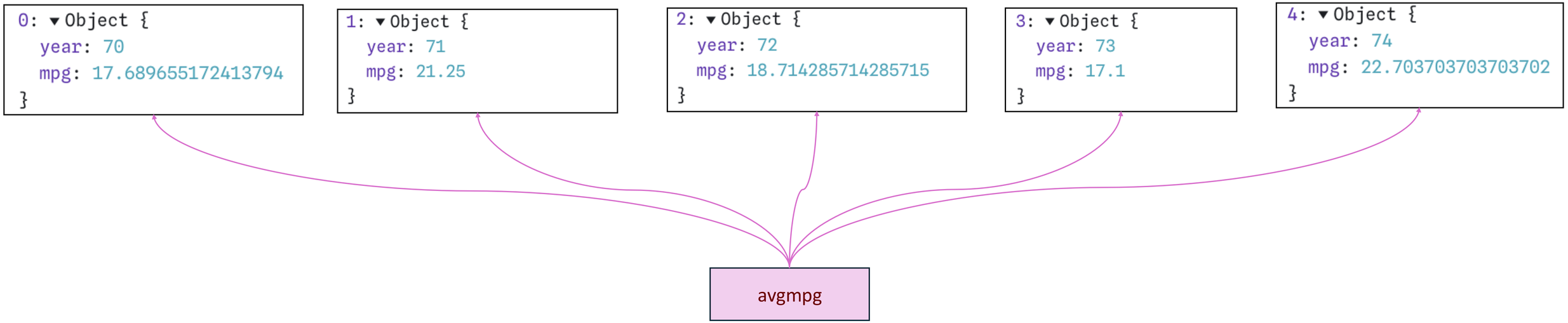
Line & area plots revisited

Access just mpg variable

Group MPG data by year

```
avgmpg = {
  const mpgbyyear = d3.rollup(cars, v => d3.mean(v, d => d["economy (mpg)"]), d => d.year)
  return [...d3.sort(mpgbyyear)].map(d => ({year: d[0], mpg: d[1]}))
}
```

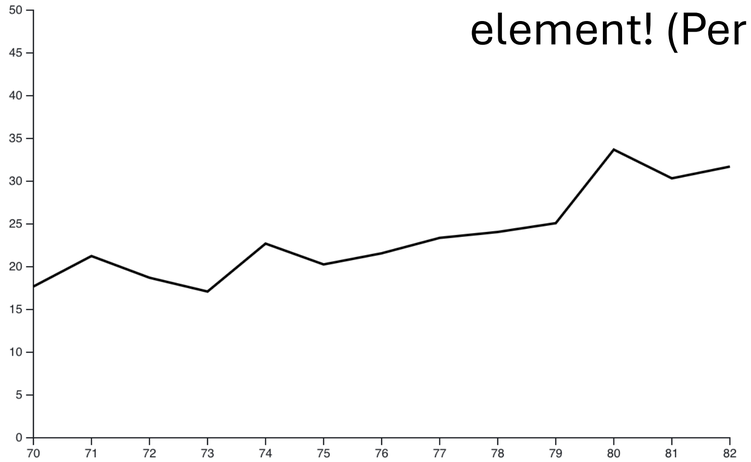
Sort & convert to objects



```
const line = d3.line()
  .x(d => x(d.year))
  .y(d => y(d.mpg));

svg.append('path')
  .attr("d", line(avgmpg))
  .attr("stroke-width", 2)
  .attr("stroke", "black")
  .attr("fill", "none")
```

D3 line generator creates “d” attribute from data

Lineplot only has 1 element! (Per line)



<svg width=”640" height=”400” >

<path d=” M40,246.172L87.5,…,134.124L562.5,157.65L610,148.032 " stroke=”black" fill=”none” stroke-width=”2" >


```
▼Object {
  name: "AMC Ambassador Brougham"
  economy (mpg): 13
  cylinders: 8
  displacement (cc): 360
  power (hp): 175
  weight (lb): 3821
  0-60 mph (s): 11
  year: 73
}
```

```
▼Object {
  name: "AMC Ambassador DPL"
  economy (mpg): 15
  cylinders: 8
  displacement (cc): 390
  power (hp): 190
  weight (lb): 3850
  0-60 mph (s): 8.5
  year: 70
}
```

```
▼Object {
  name: "AMC Ambassador SST"
  economy (mpg): 17
  cylinders: 8
  displacement (cc): 304
  power (hp): 150
  weight (lb): 3672
  0-60 mph (s): 11.5
  year: 72
}
```

```
▼Object {
  name: "AMC Concord DL 6"
  economy (mpg): 20.2
  cylinders: 6
  displacement (cc): 232
  power (hp): 90
  weight (lb): 3265
  0-60 mph (s): 18.2
  year: 79
}
```

```
▼Object {
  name: "AMC Concord DL"
  economy (mpg): 18.1
  cylinders: 6
  displacement (cc): 258
  power (hp): 120
  weight (lb): 3410
  0-60 mph (s): 15.1
  year: 78
}
```

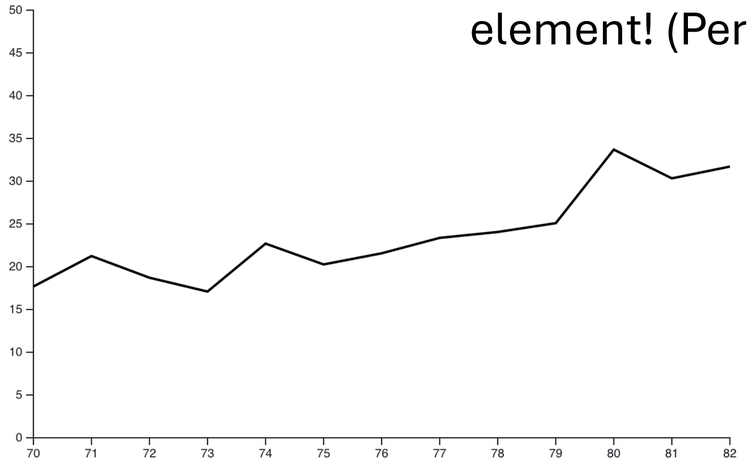
```
▼Object {
  name: "AMC Concord DL"
  economy (mpg): 23
  cylinders: 4
  displacement (cc): 151
  power (hp): null
  weight (lb): 3035
  0-60 mph (s): 20.5
  year: 82
}
```


cars

```
const line = d3.line()
  .x(d => x(d.year))
  .y(d => y(d.mpg));

svg.append('path')
  .attr("d", line(avgmpg))
  .attr("stroke-width", 2)
  .attr("stroke", "black")
  .attr("fill", "none")
```

No data joining! Just adding a single (path) element and setting its properties!

Lineplot only has 1 element! (Per line)



<svg width=”640" height=”400” >

<path d=” M40,246.172L87.5,...,134.124L562.5,157.65L610,148.032 " stroke=”black" fill=”none” stroke-width=”2" >


```
▼Object {
  name: "AMC Ambassador Brougham"
  economy (mpg): 13
  cylinders: 8
  displacement (cc): 360
  power (hp): 175
  weight (lb): 3821
  0-60 mph (s): 11
  year: 73
}
```

```
▼Object {
  name: "AMC Ambassador DPL"
  economy (mpg): 15
  cylinders: 8
  displacement (cc): 390
  power (hp): 190
  weight (lb): 3850
  0-60 mph (s): 8.5
  year: 70
}
```

```
▼Object {
  name: "AMC Ambassador SST"
  economy (mpg): 17
  cylinders: 8
  displacement (cc): 304
  power (hp): 150
  weight (lb): 3672
  0-60 mph (s): 11.5
  year: 72
}
```

```
▼Object {
  name: "AMC Concord DL 6"
  economy (mpg): 20.2
  cylinders: 6
  displacement (cc): 232
  power (hp): 90
  weight (lb): 3265
  0-60 mph (s): 18.2
  year: 79
}
```

```
▼Object {
  name: "AMC Concord DL"
  economy (mpg): 18.1
  cylinders: 6
  displacement (cc): 258
  power (hp): 120
  weight (lb): 3410
  0-60 mph (s): 15.1
  year: 78
}
```

```
▼Object {
  name: "AMC Concord DL"
  economy (mpg): 23
  cylinders: 4
  displacement (cc): 151
  power (hp): null
  weight (lb): 3035
  0-60 mph (s): 20.5
  year: 82
}
```


cars

Can conceptualize <path> elements as a pen drawing!

Pen movements

```
<path d=” M40,246.172L87.5,…,134.124L562.5,157.65L610,148.032 " stroke=”black" stroke-width=”2”
fill=”none” >
```

Pen properties

Prevents filling in shapes!

The above is rendered using a single `<path>` element with the following `d` attribute:

```
M 125 100 L 150 100 m 60 0 l 25 0 m –100 100 C 150 250, 200 250, 225 200 M 175 175 c 70 –15, 2 –20, 4.5 –70
```

Move pen to this location

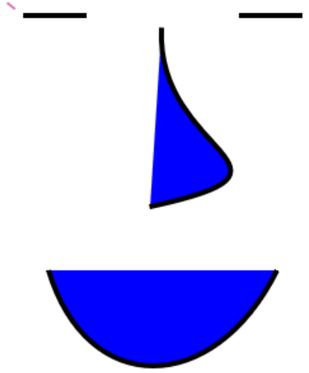
Draw line from current location this location

Lower case: coordinates relative to current location

C/c: Curves

command	description	x	y	x1	x2
M	Move to (absolute)	125.0	100.0	NA	NA
L	Line to (absolute)	150.0	100.0	NA	NA
m	Move to (relative)	60.0	0.0	NA	NA
l	Line to (relative)	25.0	0.0	NA	NA
m	Move to (relative)	–100.0	100.0	NA	NA
C	Curve to (absolute)	225.0	200.0	150.0	250.0
M	Move to (absolute)	175.0	175.0	NA	NA
c	Curve to (relative)	4.5	–70.0	70.0	–15.0

Can conceptualize <path> elements as a pen drawing!



Pen movements

```
<path d=” M40,246.172L87.5,…,134.124L562.5,157.65L610,148.032 " stroke=”black" stroke-width=”2”
fill=”blue” >
```

Pen properties

Path tries to draw closed shapes, unless Fill=“none”

The above is rendered using a single `<path>` element with the following `d` attribute:

```
M 125 100 L 150 100 m 60 0 l 25 0 m –100 100 C 150 250, 200 250, 225 200 M 175 175 c 70 –15, 2 –20, 4.5 –70
```

Move pen to this location

Draw line from current location this location

Lower case: coordinates relative to current location

C/c: Curves

command	description	x	y	x1	x2
M	Move to (absolute)	125.0	100.0	NA	NA
L	Line to (absolute)	150.0	100.0	NA	NA
m	Move to (relative)	60.0	0.0	NA	NA
l	Line to (relative)	25.0	0.0	NA	NA
m	Move to (relative)	–100.0	100.0	NA	NA
C	Curve to (absolute)	225.0	200.0	150.0	250.0
M	Move to (absolute)	175.0	175.0	NA	NA
c	Curve to (relative)	4.5	–70.0	70.0	–15.0

Can conceptualize <path> elements as a pen drawing!

Pen movements

```
<path d=” M40,246.172L87.5,…,134.124L562.5,157.65L610,148.032 " stroke=”black" stroke-width=”2”
fill=”blue” >
```

Pen properties

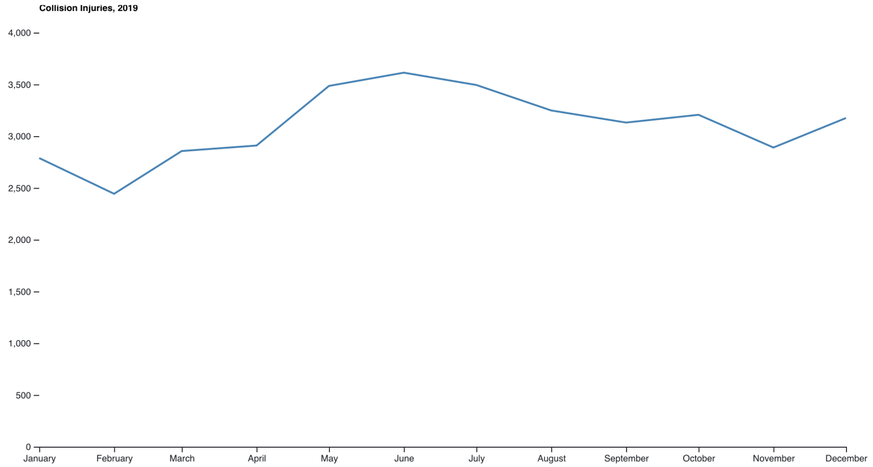
Path tries to draw closed shapes, unless Fill=“none”

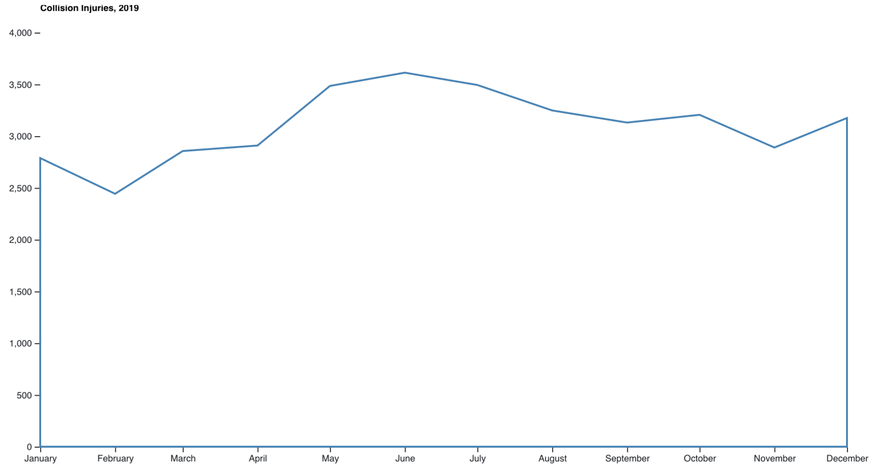
The above is rendered using a single `<path>` element with the following `d` attribute:

```
M 125 100 L 150 100 m 60 0 l 25 0 m -100 100 C 150 250, 200 250, 225 200 M 175 175 c 70 -15, 2 -20, 4.5 -70
```

Let D3 figure out the path for you!

```
const line = d3.line()
  .x(d => x(d.year))
  .y(d => y(d.mpg));

svg.append('path')
  .attr("d", line(avgmpg))
  .attr("stroke-width", 2)
  .attr("stroke", "black")
  .attr("fill", "none")
```






```
line = d3.line()
    .x(d => x(d.date))
    .y(d => y(d.injuries))
```

```
svg.append('path')
    .attr('stroke', 'steelblue')
    .attr('stroke-width', 2)
    .attr('fill', 'none')
    .attr('d', line(injuriesByMonth));
```

```
area = d3.area()
    .x(d => x(d.date))
    .y1(d => y(d.injuries))
    .y0(d => y(0))
```

```
svg.append('path')
    .attr('stroke', 'steelblue')
    .attr('stroke-width', 2)
    .attr('fill', 'none')
    .attr('d', area(injuriesByMonth));
```

```
0: ▼Object {
  date: 2019-01-01T08:00Z
  injuries: 2788
}
```

```
1: ▼Object {
  date: 2019-02-01T08:00Z
  injuries: 2443
}
```

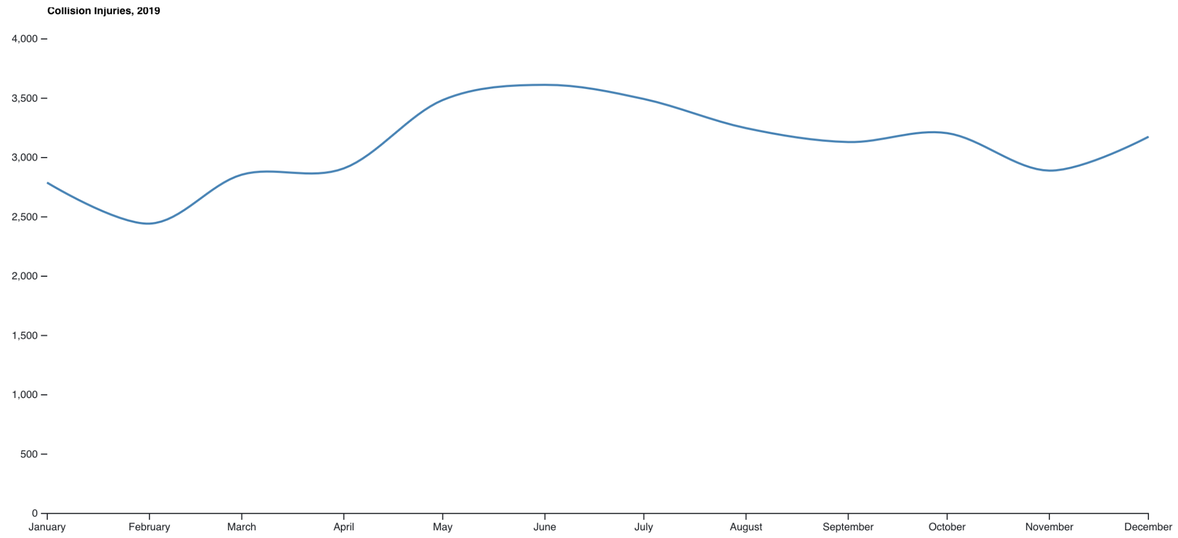
```
2: ▼Object {
  date: 2019-03-01T08:00Z
  injuries: 2856
}
```

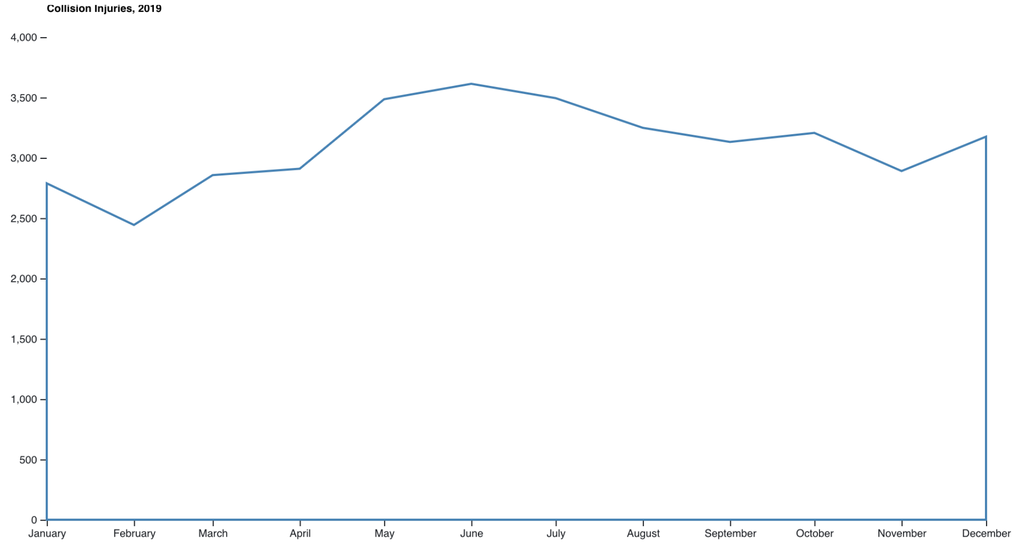
```
3: ▼Object {
  date: 2019-04-01T07:00Z
  injuries: 2909
}
```

```
4: ▼Object {
  date: 2019-05-01T07:00Z
  injuries: 3485
}
```

```
5: ▼Object {
  date: 2019-06-01T07:00Z
  injuries: 3613
}
```

injuries






```
line = d3.line()
    .x(d => x(d.date))
    .y(d => y(d.injuries))
    .curve(d3.curveCardinal)
```

```
svg.append('path')
    .attr('stroke', 'steelblue')
    .attr('stroke-width', 2)
    .attr('fill', 'none')
    .attr('d', line(injuriesByMonth));
```

```
area = d3.area()
    .x(d => x(d.date))
    .y1(d => y(d.injuries))
    .y0(d => y(0))
```

```
svg.append('path')
    .attr('stroke', 'steelblue')
    .attr('stroke-width', 2)
    .attr('fill', 'none')
    .attr('d', area(injuriesByMonth));
```

```
0: ▼Object {
  date: 2019-01-01T08:00Z
  injuries: 2788
}
```

```
1: ▼Object {
  date: 2019-02-01T08:00Z
  injuries: 2443
}
```

```
2: ▼Object {
  date: 2019-03-01T08:00Z
  injuries: 2856
}
```

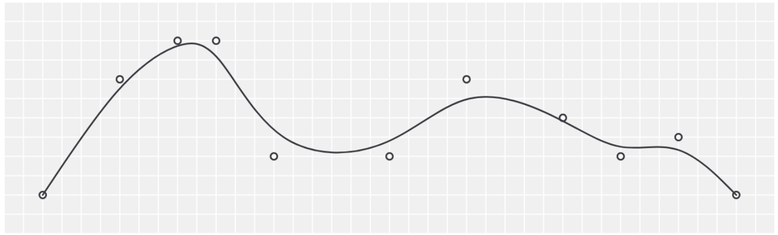
```
3: ▼Object {
  date: 2019-04-01T07:00Z
  injuries: 2909
}
```

```
4: ▼Object {
  date: 2019-05-01T07:00Z
  injuries: 3485
}
```

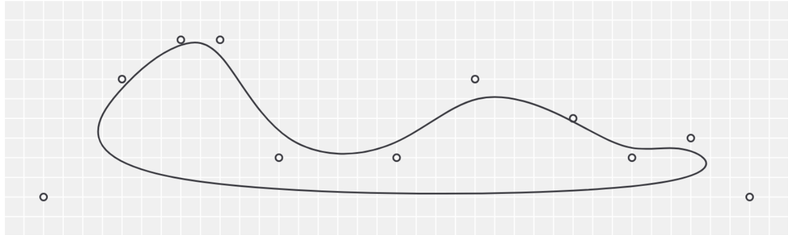
```
5: ▼Object {
  date: 2019-06-01T07:00Z
  injuries: 3613
}
```

injuries

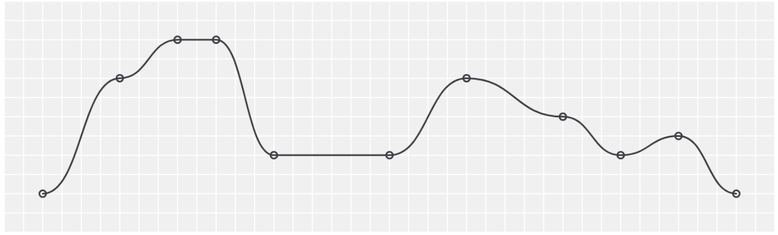
curveBasis(*context*)



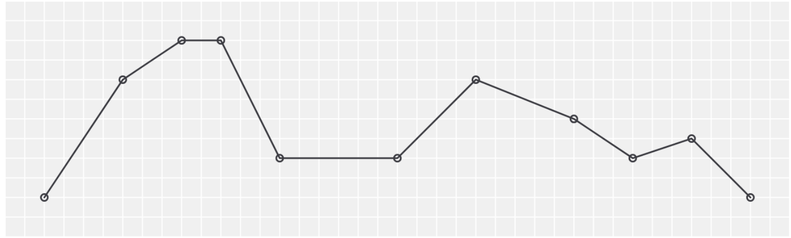
curveBasisClosed(*context*)



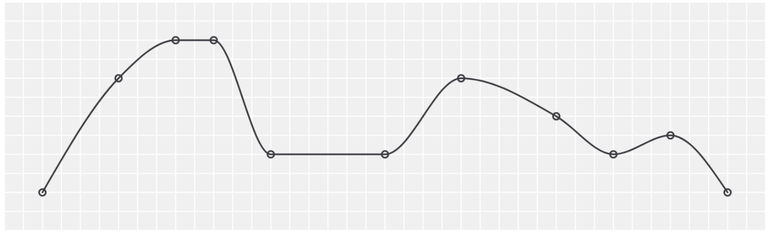
curveBumpX(*context*)



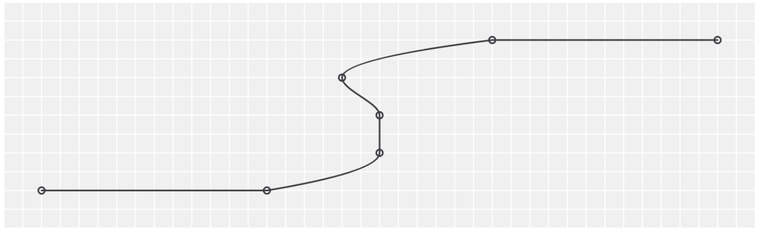
curveLinear(*context*)



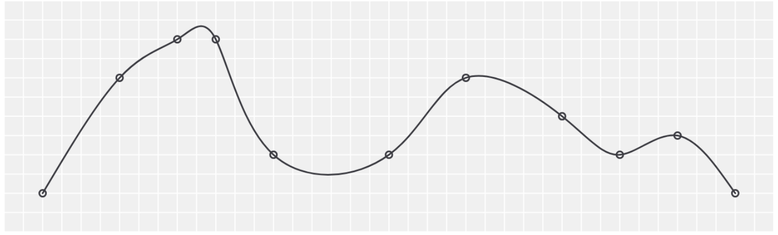
curveMonotoneX(*context*)



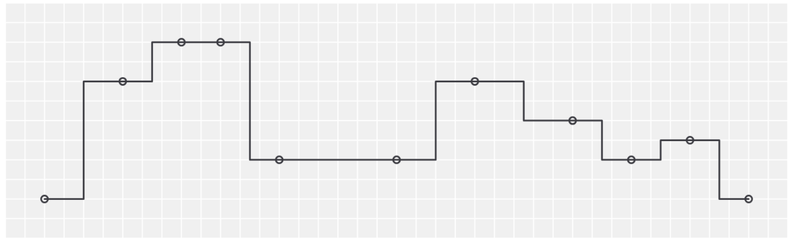
curveMonotoneY(*context*)

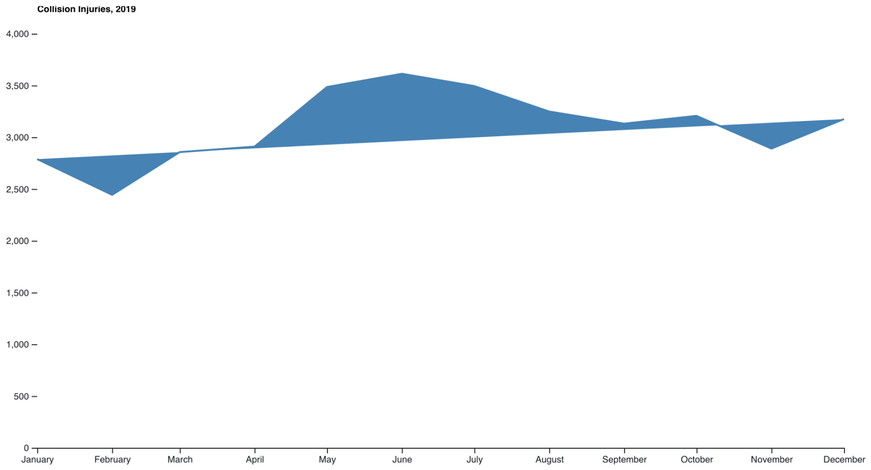


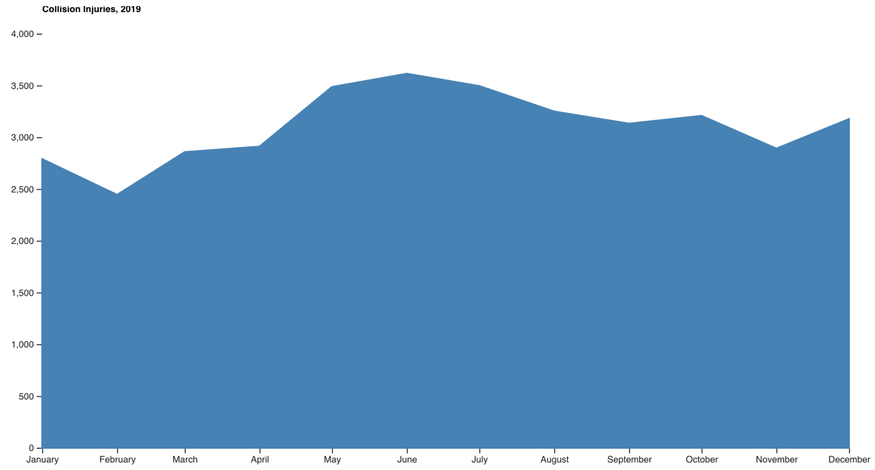
curveNatural(*context*)



curveStep(*context*)








```
line = d3.line()
     .x(d => x(d.date))
     .y(d => y(d.injuries))
```

```
svg.append('path')
     .attr('stroke', 'steelblue')
     .attr('stroke-width', 2)
     .attr('fill', 'none')
     .attr('d', line(injuriesByMonth));
```

```
area = d3.area()
     .x(d => x(d.date))
     .y1(d => y(d.injuries))
     .y0(d => y(0))
```

```
svg.append('path')
     .attr('stroke', 'steelblue')
     .attr('stroke-width', 2)
     .attr('fill', 'none')
     .attr('d', area(injuriesByMonth));
```

```
0: ▼Object {
  date: 2019-01-01T08:00Z
  injuries: 2788
}
```

```
1: ▼Object {
  date: 2019-02-01T08:00Z
  injuries: 2443
}
```

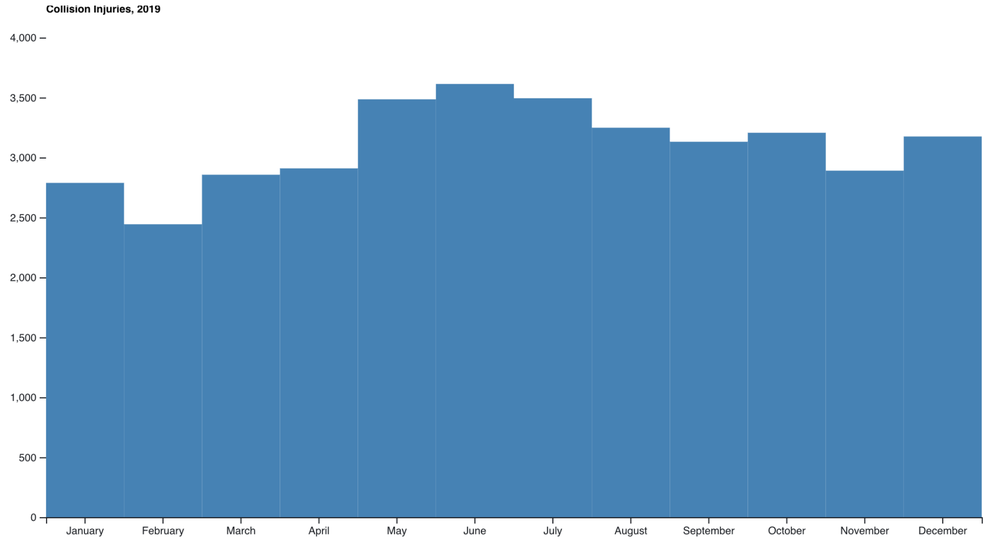
```
2: ▼Object {
  date: 2019-03-01T08:00Z
  injuries: 2856
}
```

```
3: ▼Object {
  date: 2019-04-01T07:00Z
  injuries: 2909
}
```

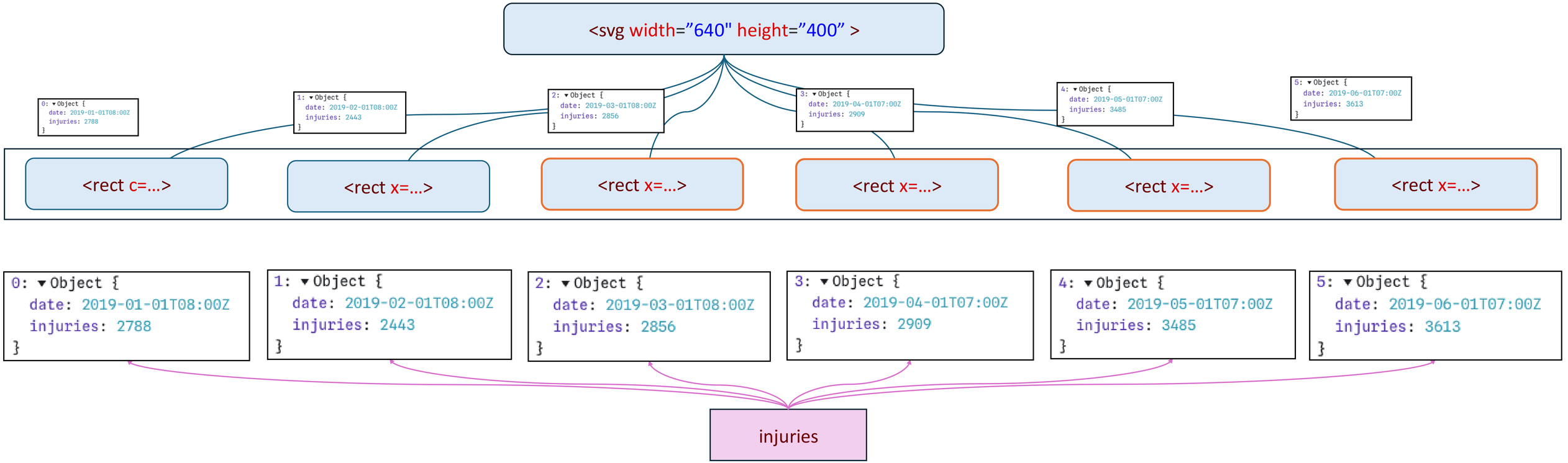
```
4: ▼Object {
  date: 2019-05-01T07:00Z
  injuries: 3485
}
```

```
5: ▼Object {
  date: 2019-06-01T07:00Z
  injuries: 3613
}
```

injuries




```
svg.selectAll('rect')
   .data(injuriesByMonth)
   .enter().append('rect')
   .attr('x', d => xband(d.date))
   .attr('width', d => xband.bandwidth())
   .attr('y', d => y(d.injuries))
   .attr('height', d => y(0) - y(d.injuries))
   .attr('fill', 'steelblue')
```



Case study

```
chart = {
  const width = 1152;
  const height = 400;
  const margin = {top: 20, right: 30, bottom: 30, left: 40};

  const x = d3.scaleBand()
      .domain(d3.reverse([...Array(110).keys()]))
      .range([margin.left, width - margin.right]);

  const y = d3.scaleLinear()
      .domain([0, 3500])
      .range([height - margin.bottom, margin.top]);

  const svg = d3.create("svg")
      .attr("width", width)
      .attr("height", height);

  svg.append("g")
      .attr("transform", `translate(0,${height - margin.bottom})`)
      .call(d3.axisBottom(x).tickValues([...Array(22).keys()].map((d) => d * 5)));

  svg.append("g")
      .attr("transform", `translate(${width - margin.right},0)`)
      .call(d3.axisRight(y).tickFormat(d => Math.round(d / 500) / 2 + 'k'));

  const colors = ["#4e79a7", "#e15759"];
  const data = population.filter((d) => d.year === icelandYear);
  svg.selectAll("rect")
      .data(data)
      .join("rect")
      .style("mix-blend-mode", "darken")
      .attr("fill", d => d.sex == 'M' ? colors[0] : colors[1])
      .attr("x", d => x(d.age))
      .attr("y", d => y(d.value))
      .attr("width", x.bandwidth())
      .attr("height", d => y(0) - y(d.value))

  return svg.node();
}
```

Year 1993

```
viewof icelandYear = Inputs.range([1841, 2019], {label: 'Year', step: 1})
```