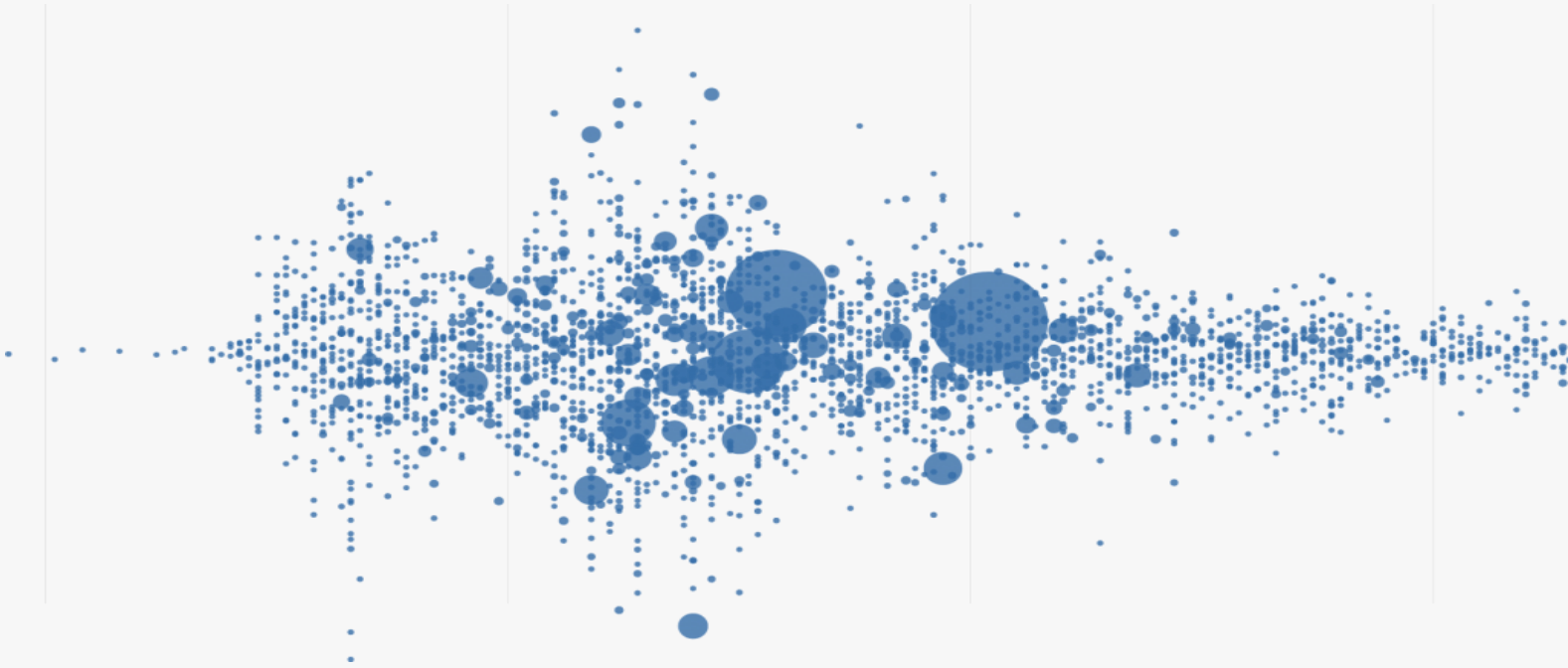
Updates, Motion and Transitions

Workshop

Melvin Weiershäuser
und Aileen Jurkosek



Einstieg

- Datensätze verändern sich stetig
  - → Veränderungen von Daten müssen in der Visualisierung dargestellt werden
- Updates ermöglichen Veränderungen
- Transitions passen die visuelle Darstellung an

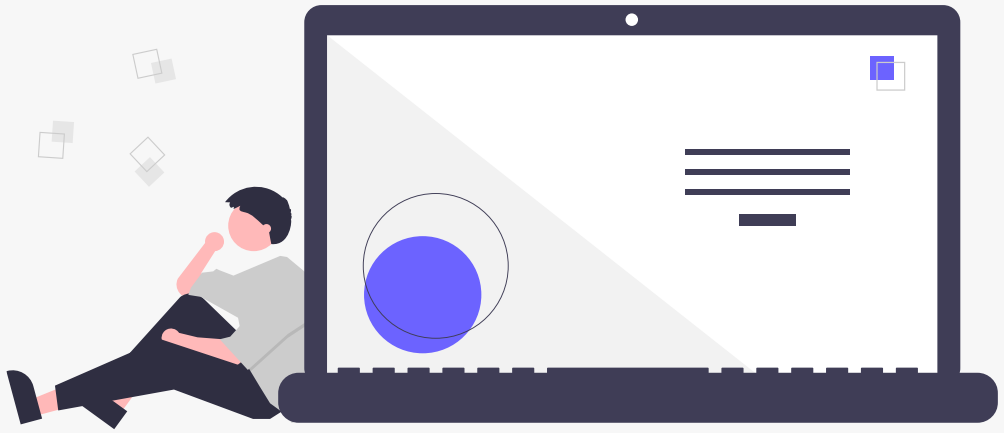


Table of contents

01

Updates

Updates

- Einfaches Update: alle Datenwerte werden zur gleichen Zeit aktualisiert, die Menge der Werte bleibt allerdings gleich

Vorgehen:

1. Daten im Datensatz werden angepasst
2. Neue Datenwerte werden an bereits existierende Elemente gebunden
3. Neue Attributwerte für die Visualisierung der Daten werden gesetzt

Event Listener

- Event Listener wird zu einem Element hinzugefügt
- Listener wartet auf ein Event – findet es statt wird ein nachstehendes Ereignis getriggert
- Mit D3 Methode .on() kann ein Event Listener an ein Element gebunden werden
  → Zwei Argumente: Typ und Listener

```
<p>Click on this text to update the chart with new data values (once).</p>
```

```
d3.select("p")
    .on("click", function() {
        //Do something on click
    });
```

Daten verändern

1. Dataset wird geupdatet, indem bisherige Werte überschrieben werden

```
dataset = [ 11, 12, 15, 20, 18, 17, 16, 18, 23, 25,
            5, 10, 13, 19, 21, 25, 22, 18, 15, 13 ];
```

2. Rects werden ausgewählt und data() mit dem neuen Datensatz aufgerufe

```
svg.selectAll("rect")
   .data(dataset);     //New data successfully bound!
```

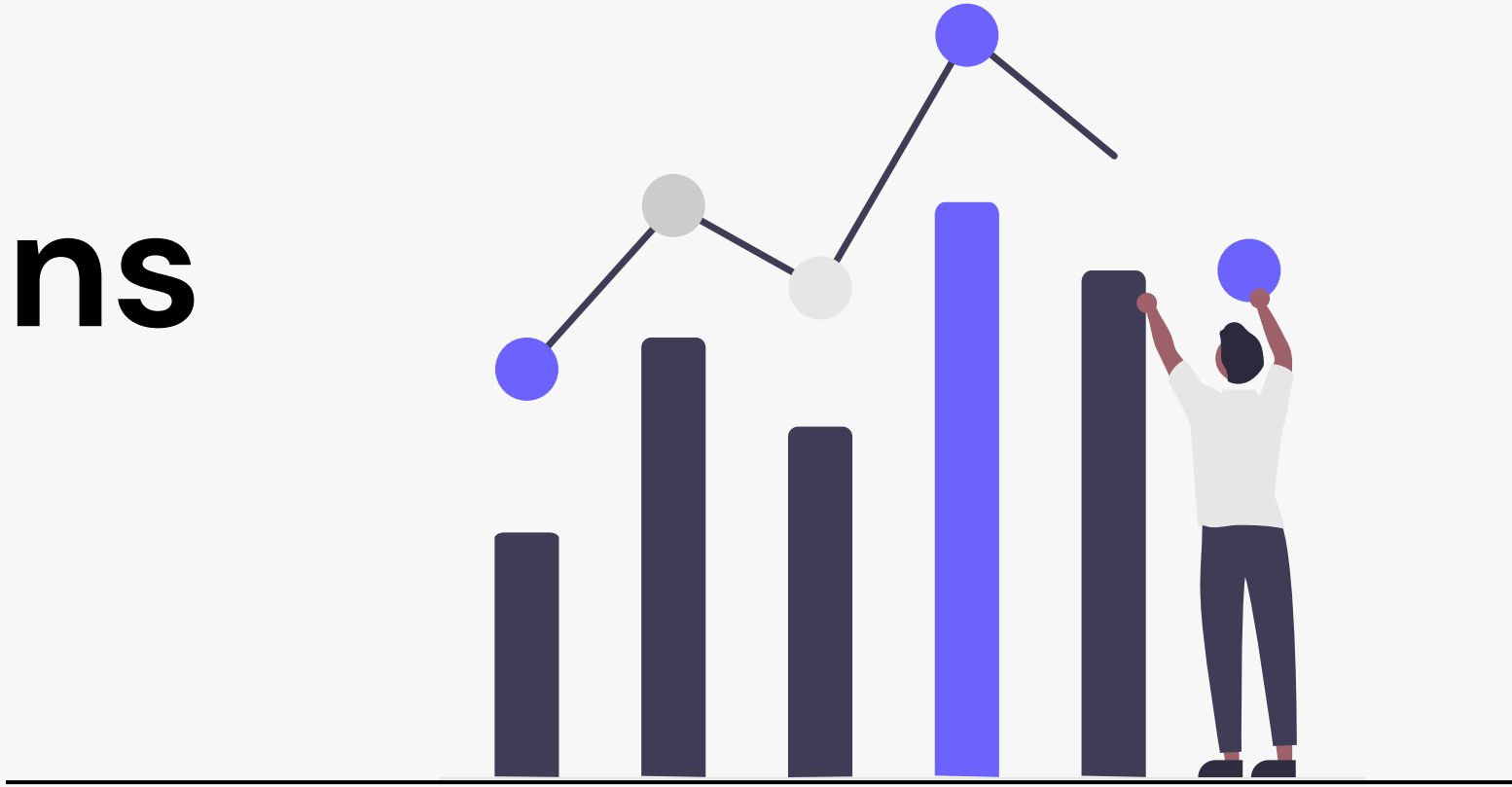
Visualisierung updaten

- Visuelle Attribute müssen geupdatet werden, damit die neuen Datenwerte referenziert werden
- Auch Farbwerte und Beschriftungen müssen angepasst werden, damit die Visualisierung sinnig ist

```
svg.selectAll("rect")
   .data(dataset)
   .attr("y", function(d) {
       return h - yScale(d);
   })
   .attr("height", function(d) {
       return yScale(d);
   });
```

02

Transitions



Transitions

- Können in D3 über die Funktion .transition() hinzugefügt werden
- NACH der Selektierung der Daten und VOR der Veränderung der Attribute
- Über die Transition wird eine Animation eingefügt, welche den Zustandsübergang visualisiert

```
d3.selectAll("circle")
  .attr("cx", 0)      // Initial value for 'cx' is set
  .transition()       // Transition is initiated
  .attr("cx", 100);  // 'cx' will be interpolated to 100
```

Scales Updaten

→ Mit geänderten Datensätzen muss ggf. auch der Scale angepasst werden, damit alle Werte dargestellt werden können und die Balken der Visualisierung nicht zu klein oder zu groß dargestellt werden

```
var yScale = d3.scaleLinear()
                .domain([0, d3.max(dataset)])
                .range([0, h]);
```

```
//Update scale domain
yScale.domain([0, d3.max(dataset)]);
```

Achsen Updaten

Vorgehen:

- Achse wird ausgewählt
- Transition wird initiiert

```
//Create x-axis
svg.append("g")
    .attr("class", "x axis")    // <-- Note x added here
    .attr("transform", "translate(0," + (h - padding) + ")")
    .call(xAxis);

//Create y-axis
svg.append("g")
    .attr("class", "y axis")    // <-- Note y added here
    .attr("transform", "translate(" + padding + ",0)")
    .call(yAxis);
```

```
//Update x-axis
svg.select(".x.axis")
    .transition()
    .duration(1000)
    .call(xAxis);

//Update y-axis
svg.select(".y.axis")
    .transition()
    .duration(1000)
    .call(yAxis);
```

Start und Ende einer Transition mit on()

- On() kann eingesetzt werden, um ein Ereignis zu definieren, das am Anfang oder Ende einer Transition passieren soll
- Zwei Argumente:
  - Entweder start oder end
  - Anonyme Funktion, die ausgeführt werden soll

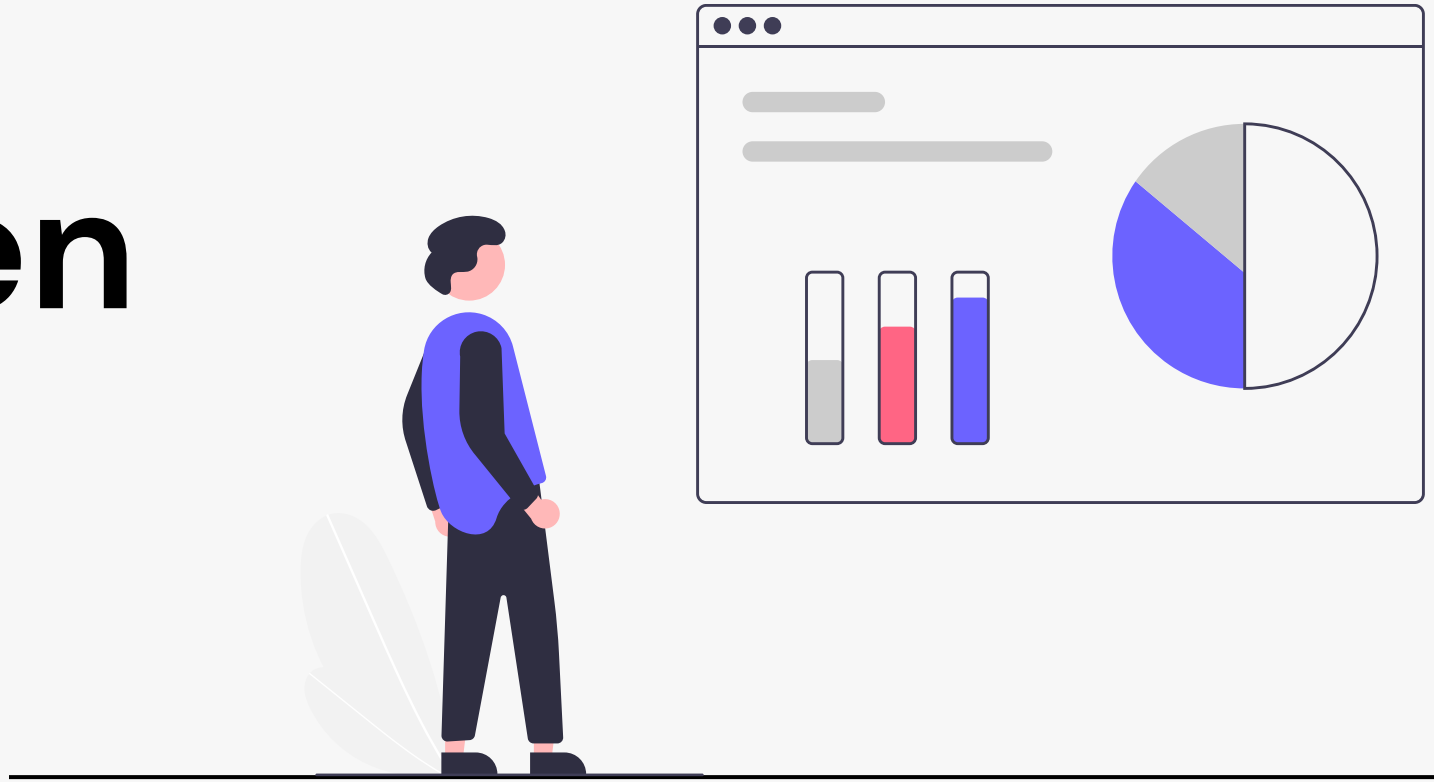
Beispiele:

```
.on("start", function() {
    d3.select(this)
      .attr("fill", "magenta")
      .attr("r", 3);
})
```

```
.on("end", function() {
    d3.select(this)
      .transition()
      .duration(1000)
      .attr("fill", "black")
      .attr("r", 2);
});
```

03

Methoden



duration()

- Beschreibt die Zeit, die während einer Transition vergeht
- Standardmäßig 250 Millisekunden

```
d3.selectAll("circle")
  .attr("cx", 0)
  .transition()
  .duration(2000)  // Transition will occur over 2 seconds
  .attr("cx", 100);
```

ease()

- Beschreibt die Bewegung innerhalb einer Transition
- Standardmäßig ist die Bewegung variabel (D3.easeCubicInOut)
- Kann linear eingestellt werden – konstante Geschwindigkeit der Bewegung und abruptes Ende

```
d3.selectAll("circle")
  .attr("cx", 0)
  .transition()
  .ease(d3.easeLinear)  // Transition will be linear
  .attr("cx", 100);
```

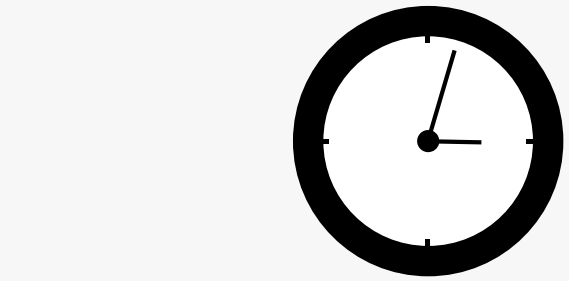
delay()

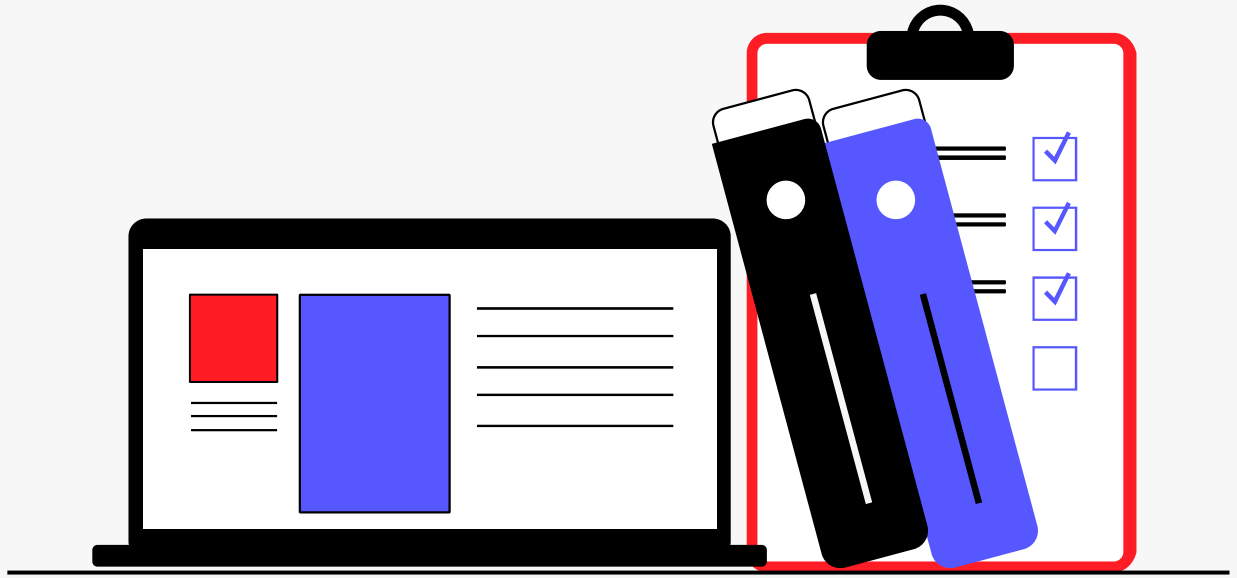
- Gibt an, ob die Transition verzögert beginnen soll
- Reihenfolge von ease() und duration() ist flexibel
- Kann für die einzelnen Elemente dynamisch eingestellt werden

```
d3.selectAll("circle")
  .attr("cx", 0)
  .transition()
  .delay(1000)  // Wait 1 second before starting
  .attr("cx", 100);
```

Aufgabe!

- Datensätze tauschen/updaten
- Mit Transition und ohne
- Ggf. Delay oder ease





04

Andere Arten von Data Updates

Werte hinzufügen

- Durch Hinzufügen eines Wertes wird das Dataset Array verlängert
- Achsen müssen zusätzlich in ihrer Länge angepasst werden
- Durch enter() können neue Daten gebindet werden
- Durch merge() kann die enter selection mit der update selection kombiniert werden

```
d3.selectAll("circle")
  .data(dataset)
  .enter()  // Returns the placeholders for circles to-be-created
  .append("circle");  // Creates a circle for each placeholder
```

```
bars.enter()  // Get the enter subselection
    .append("rect")
    …  // Set attributes for new elements…
    .merge(bars)  // Merge enter subselection with existing bars selection
    …  // Set attributes for all elements…
```

Werte entfernen

- In exit selection wird remove() aufgerufen
- Mit einer Transition wird das Element aus der Visualisierung entfernt
- Bars müssen vorher selected und mit der neuen Data rebinded werden

```
bars.exit()     // Get the exit subselection
    .transition()
    …  // Set attributes for exiting elements, e.g. dial down opacity…
```

```
bars.exit()     // Get the exit subselection
    .remove();  // Delete exiting elements immediately
```

Data Joins mit Keys

- Mit einer Key Function kann festgelegt werden, dass bestimmte Werte an bestimmte Elemente gebunden werden sollen
- Jedem Wert muss ein Key hinzugefügt werden, welcher zur Identifizierung dient
- Anstatt eines Werte-Arrays wird ein Array von Objekten erstellt, indem jedes Objekt einen Wert und einen Key erhält

```
var dataset = [ { key: 0, value: 5 },
                { key: 1, value: 10 },
                { key: 2, value: 13 },
                ... ]
```

```
var yScale = d3.scaleLinear()
                .domain([0, d3.max(dataset, function(d) { return d.value; })])
                .range([0, h]);
```

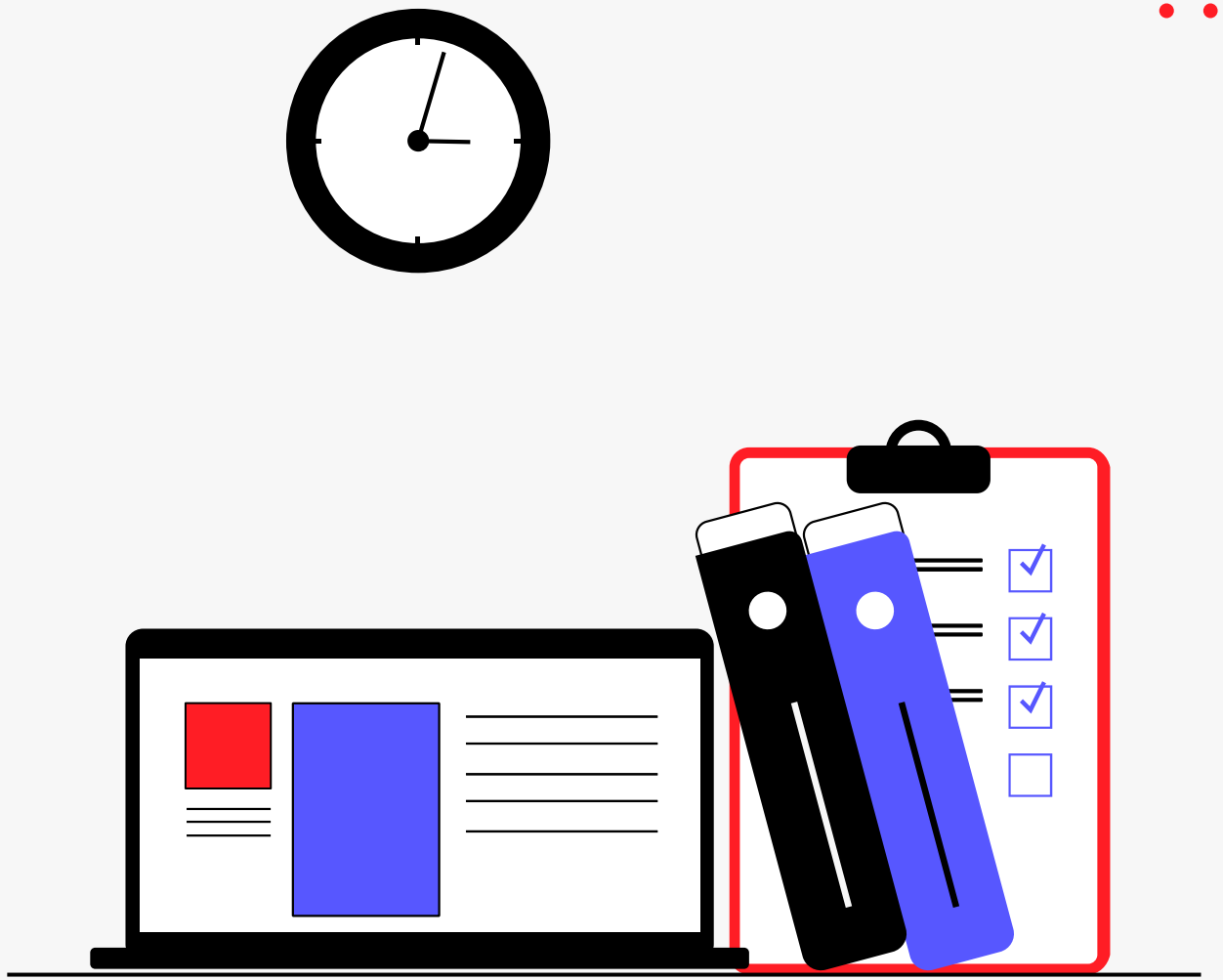
Key Funktionen

- Key Funktion wird für alle Fälle definiert, in denen Daten an ein Element gebunden werden
- Funktion erhält Objekte als Input und liefert den jeweiligen Key zurück →
- Key Values bieten mehr Flexibilität beim Hinzufügen und Löschen von Daten

```
var key = function(d) {
    return d.key;
};
```

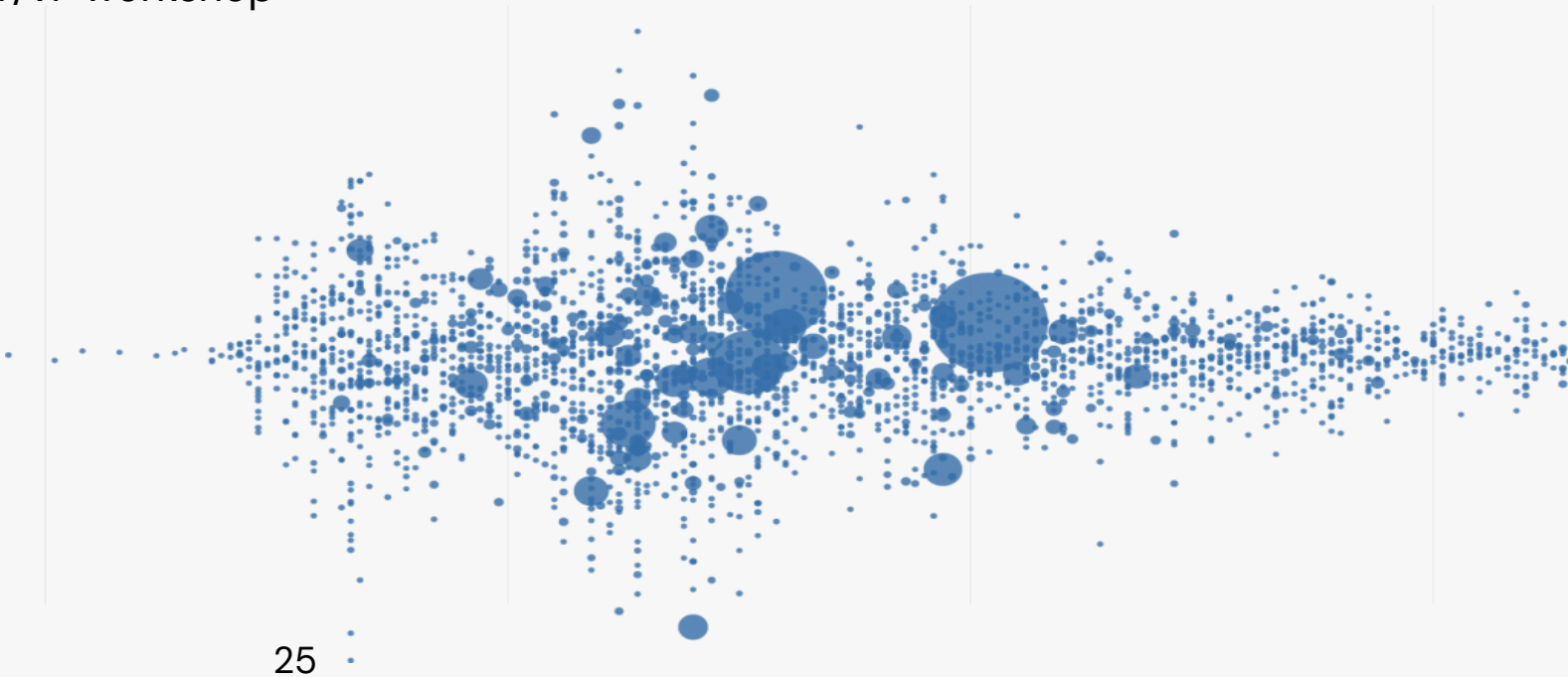
Aufgabe!

- add und remove umsetzen
- Data Joins



Resources

- https://clusterdesign.io/data-visualization-science-art-or-both/
- https://slidesgo.com/theme/administrative-process-review-meeting#position-1&rs=home-latest
- https://undraw.co/illustrations
- MURRAY, Scott. Interactive data visualization for the web: an introduction to designing with. " O'Reilly Media, Inc.", 2017.
- https://github.com/mweiershaeuser/vi-workshop



Danke für eure Aufmerksamkeit!