Handout - Updates, Transitions and Motion

Updates

1. Daten im Datensatz anpassen
2. neue Datenwerte an existierende Elemente binden
3. neue Attributwerte in Visualisierung setzen

Event Listener

- wartet auf ein Event und triggert ein darauf folgendes Ereignis

```
<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

- bisherige Werte überschreiben
- neue Werte werden an existierende Elemente angefügt

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

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

Visualisierung updaten

- visuelle Attribute, Farbwerte und Schriften müssen angepasst werden, um die neuen Datenwerte zu referenzieren

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

```
d3.select("p")
    .on("click", function() {

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

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

Scales updaten()

- Scale muss ggf. an geänderte Datensätze angepasst werden

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

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


von Melvin Weiershäuser und Aileen Jurkosek

Achsen updaten()

- entsprechende Achsen werden ausgewählt und 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);
```

Transitions

- visualisiert den Übergang zwischen verschiedenen Zuständen
- nach Selektierung der Daten und vor Veränderung der Attribute

```
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
```

Start und Ende einer Transition mit on()

- on() erhält als Argumente start oder end, sowie eine anonyme Funktion, die am Start oder Ende der Transition ausgeführt werden soll

```
.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);
});
```

Methoden

duration()

- Zeit, die während einer Transition vergeht (in Millisekunden)

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

delay()

- Verzögerung der Transition

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

ease()

- Bewegung der Transition (Beschleunigung/ bremsen)

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

→

`d3.easeCircleIn`
Gradual ease in and acceleration until elements snap into place.

`d3.easeElasticOut`
The best way to describe this one is "sproingy."

`d3.easeBounceOut`
Like a ball bouncing, then coming to rest.

---

von Melvin Weiershäuser und Aileen Jurkosek

Andere Arten von Data Updates

Werte hinzufügen

- Daten-Array wird verlängert → x-Achse muss in ihrer Länge angepasst werden
- enter() wird aufgerufen, wenn neue Daten gebindet werden
- durch merge() wird enter-selection mit update-selection kombiniert

```
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

- mit exit()-Selektion wird remove() aufgerufen, über die ein Element entfernt werden kann

```
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

- Jedem Wert muss ein Key hinzugefügt werden, über den er identifiziert werden kann
- anstatt eines Werte-Arrays wird ein Objekt-Array erstellt

```
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 kann aufgerufen werden, wenn Daten an ein Element gebunden werden
- mehr Flexibilität und Hinzufügen und Löschen von Daten

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

- Daten werden wie folgt referenziert:

```
.data(dataset, key)
```

Combo Platter

- “gleichzeitiges” hinzufügen oder entfernen von Daten

```
//See which p was clicked
var paragraphID = d3.select(this).attr("id");

//Decide what to do next
if (paragraphID == "add") {
    //Add a data value
    var minValue = 2;
    var maxValue = 25 - minValue;
    var newNumber = Math.floor(Math.random() * maxValue) + minValue;
    var lastKeyValue = dataset[dataset.length - 1].key;
    dataset.push({
        key: lastKeyValue + 1,
        value: newNumber

    });
} else {
    //Remove a value
    dataset.shift();
}
```


von Melvin Weiershäuser und Aileen Jurkosek