UNIVERSITÄT BERN

Übung Open Data:

Daten visualisieren & Layouts

Termin 6, 7. April 2016

Dr. Matthias Stürmer und Prof. Dr. Thomas Myrach

Forschungsstelle Digitale Nachhaltigkeit
Institut für Wirtschaftsinformatik
Universität Bern

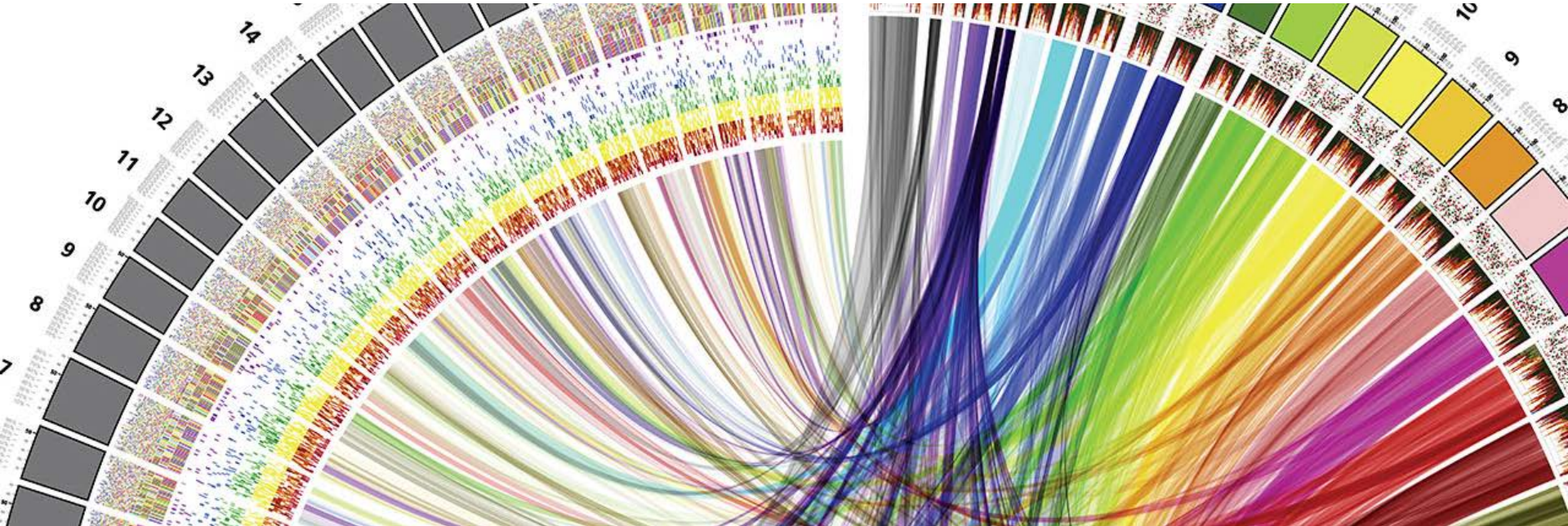
Terminübersicht Übungen

- 25.02.2016: Informationen zu den Übungen, App-Demos & Einführung in Tools
- 03.03.2016: Einführung Web-Programmierung
- 10.03.2016: Open Data Speed Dating
- 17.03.2016: Einführung D3.js & Daten einbinden in D3.js
- 24.03.2016: Anpassen von bestehenden Apps & Bibliotheken, die D3.js verwenden
- 31.03.2016: Osterferien
- **07.04.2016: Daten visualisieren & Layouts**
- 14.04.2016: Skalen und Achsen & Responsive Design
- 21.04.2016: User Experience, Usability Patterns
- 28.04.2016: Zwischenpräsentation & Datenaktualisierung und Transitionen
- 05.05.2016: Auffahrt
- 12.05.2016: Interactivity & Geomapping
- 19.05.2016: 3D Web-Programmierung mit Three.js & Programming Coaching
- 26.05.2016: Abschlusspräsentationen
- 02.06.2016: frei

UNIVERSITÄT BERN

Agenda

1. **Daten als Balken darstellen**
2. Daten als Kreise darstellen und positionieren
3. Layouts

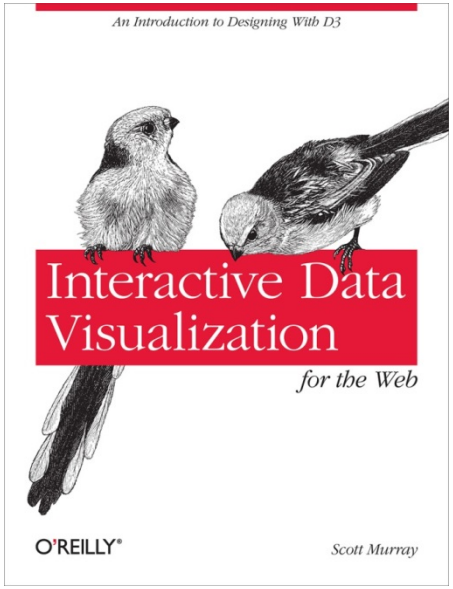


Interactive Data Visualization for the Web

Chapter 6. Drawing with Data:

> *It's time to start drawing with data.*

> http://chimera.labs.oreilly.com/books/1230000000345/ch06.html



Drawing divs

```
<div style="display: inline-block;
            width: 20px;
            height: 75px;
            background-color: teal;"></div>
```

zeichnet:

CSS class "bar"

In das CSS Stylesheet einfügen:

```
div.bar {
    display: inline-block;
    width: 20px;
    height: 75px;   /* We'll override height later */
    background-color: teal;
}
```

Von jetzt an **`div`** einfach formatieren:

```
<div class="bar"></div>
```

Setting Attributes

```
<p class="caption">
<select id="country">
<img src="logo.png" width="100px" alt="Logo" />
```

Diese HTML-Elemente enthalten 5 Attribute:

Attribute	Value
class	caption
id	country
src	logo.png
width	100px
alt	Logo

Dem Attribut "class" den Wert "bar" zuweisen:

```
.attr("class", "bar")
```

Using D3.js API Reference

Befehl (operator) **`selection.attr()`** in API Referenz nachschlagen:

selection.**attr**(*name*[, *value*])

If *value* is specified, sets the attribute with the specified name to the specified value on all selected elements. If *value* is a constant, then all elements are given the same attribute value; otherwise, if *value* is a function, then the function is evaluated for each selected element (in order), being passed the current datum `d` and the current index `i`, with the `this` context as the current DOM element. The function's return value is then used to set each element's attribute. A null value will remove the specified attribute.

If *value* is not specified, returns the value of the specified attribute for the first non-null element in the selection. This is generally useful only if you know that the selection contains exactly one element.

The specified *name* may have a namespace prefix, such as `xlink:href`, to specify an "href" attribute in the XLink namespace. By default, D3 supports svg, xhtml, xlink, xml, and xmlns namespaces. Additional namespaces can be registered by adding to d3.ns.prefix.

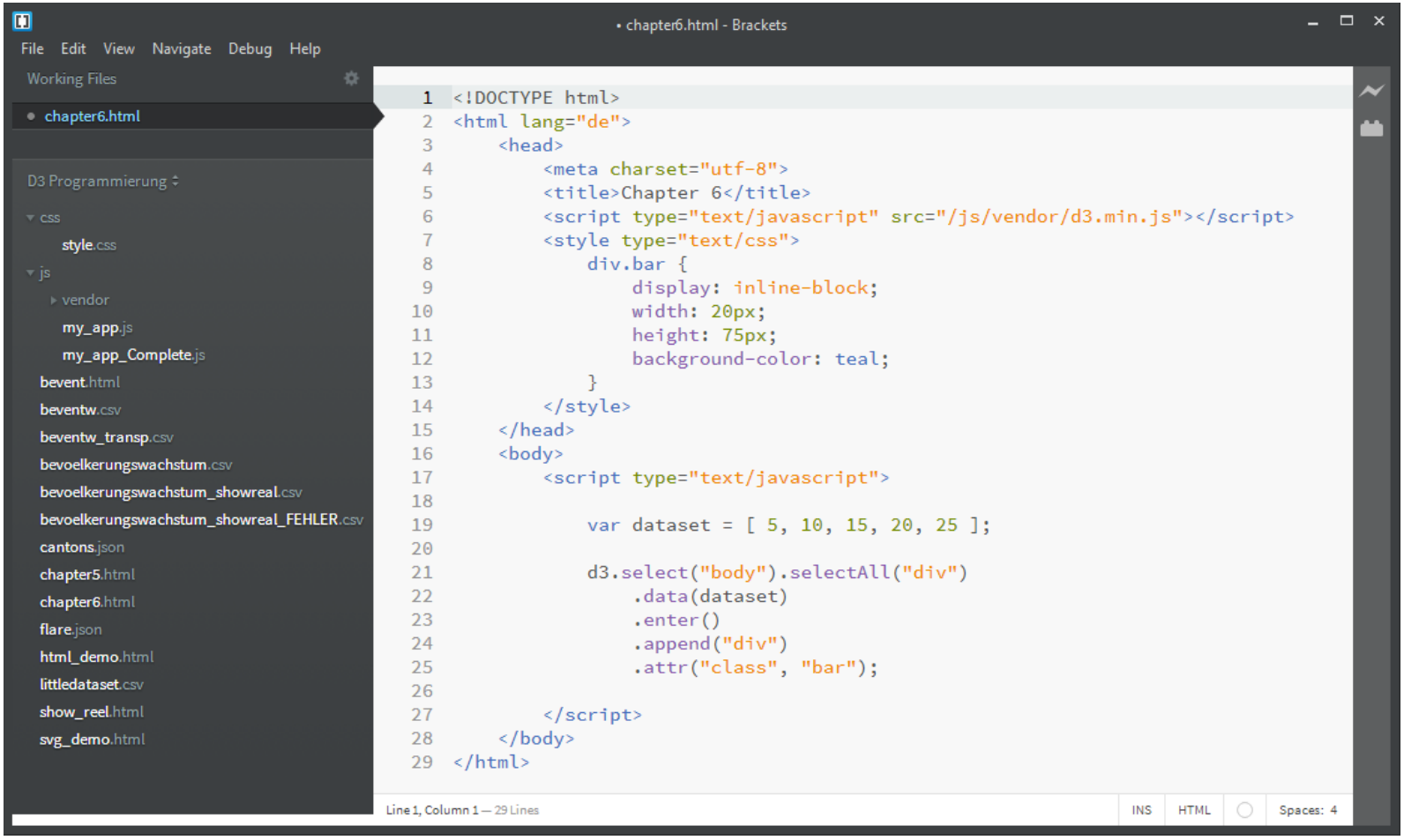
name can also be an Object of *name* and *value* attributes.

Link: https://github.com/mbostock/d3/wiki/Selections#attr

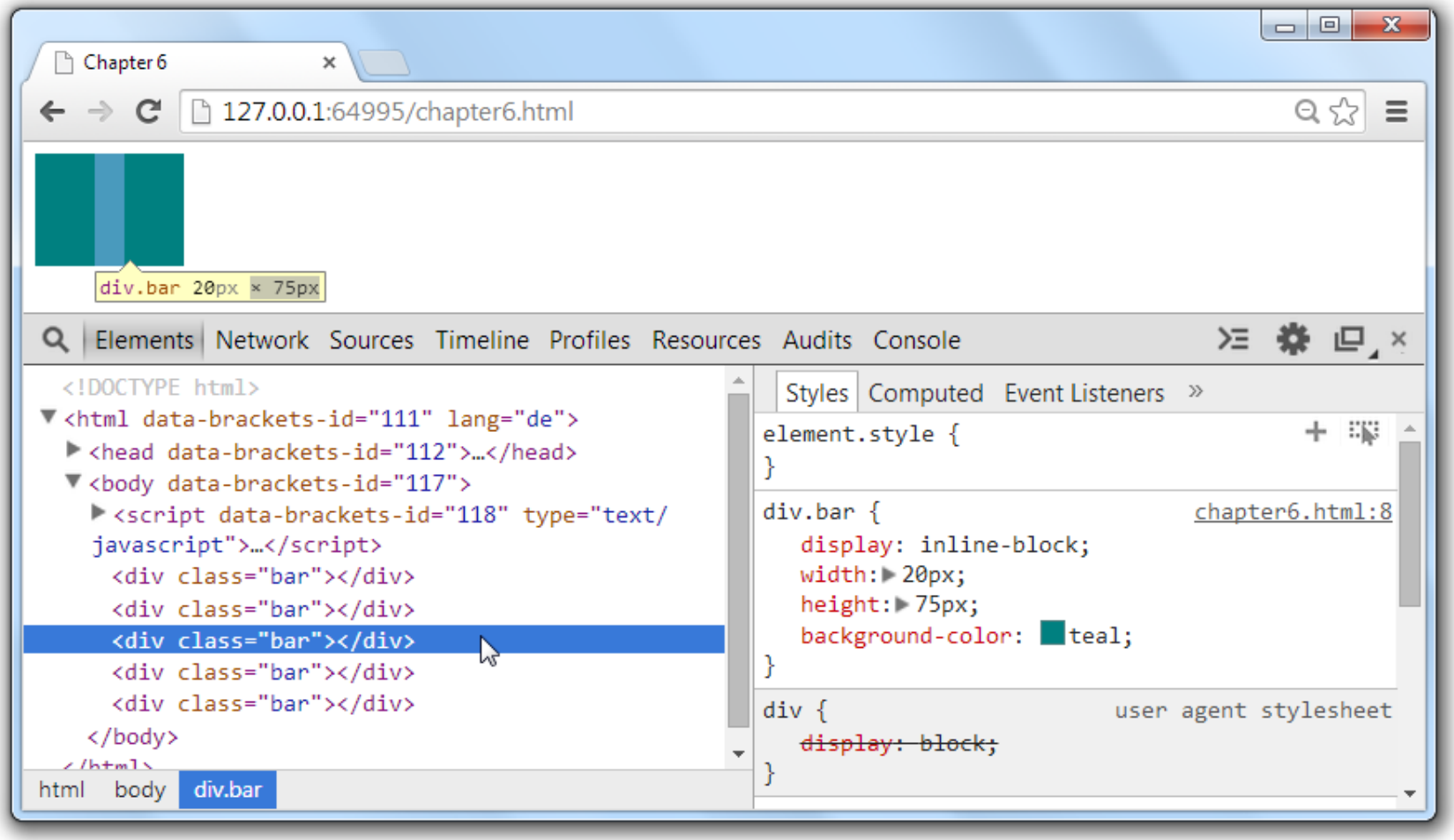
D3.js Bar Example

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Chapter 6</title>
        <script type="text/javascript" src="/js/vendor/d3.min.js"></script>
        <style type="text/css">
            div.bar {
                display: inline-block;
                width: 20px;
                height: 75px;
                background-color: teal;
            }
        </style>
    </head>
    <body>
        <script type="text/javascript">

            var dataset = [ 5, 10, 15, 20, 25 ];

            d3.select("body").selectAll("div")
                .data(dataset)
                .enter()
                .append("div")
                .attr("class", "bar");

        </script>
    </body>
</html>
```

D3.js Bar Example



D3.js Bar Example



D3.js Bar Example

Balken mit **`div`** aus dem Array generieren:

```
var dataset = [ 5, 10, 15, 20, 25 ];

d3.select("body").selectAll("div")
     .data(dataset)
     .enter()
     .append("div")
     .attr("class", "bar");
```

Setting Styles

Die Höhe eines Balkens kann mit **`height`** festgelegt werden:

```
<div style="height: 75px;"></div>
```

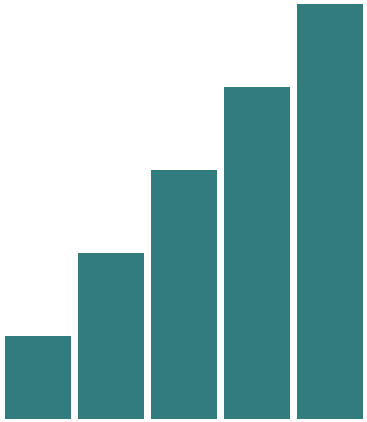
Im D3.js Code den Befehl **`.style("height"...)`** anfügen:

```
d3.select("body").selectAll("div")
      .data(dataset)
      .enter()
      .append("div")
      .attr("class", "bar")
      .style("height", function(d) {return d + "px";});
```

Adding Space

```
div.bar {
    display: inline-block;
    width: 20px;
    height: 75px;
    background-color: teal;
    margin-right: 2px;
}
```

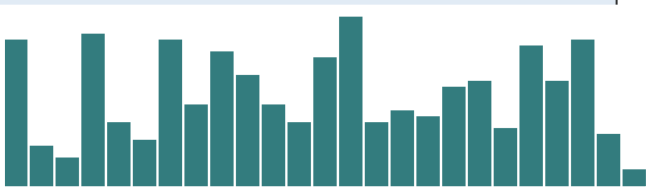
zeichnet:



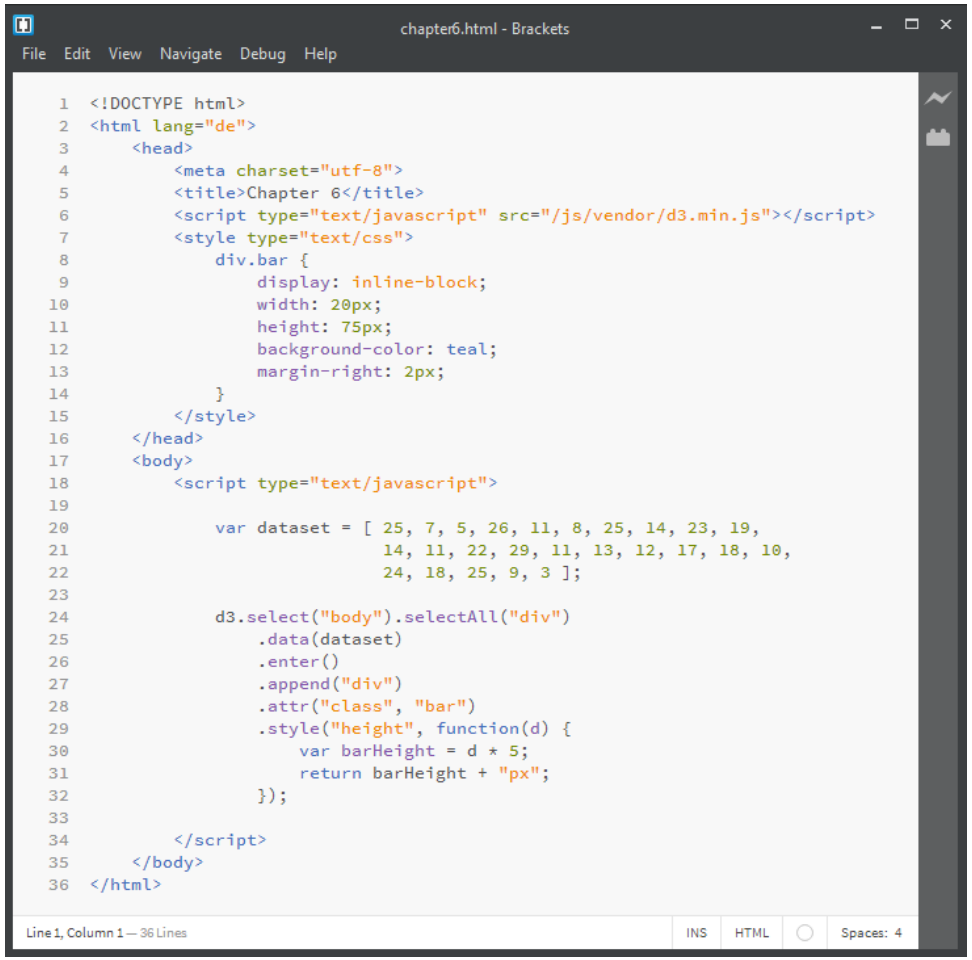
Flexibilität von .data()

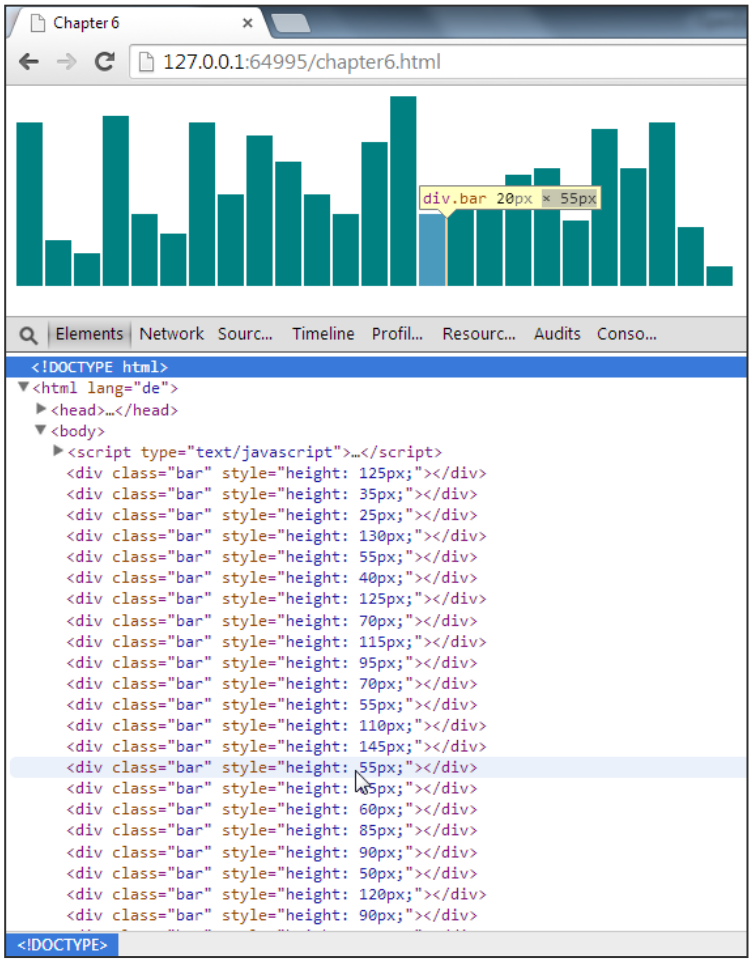
Mehr Daten hinzufügen und Höhe der Balken verstärken:

```
var dataset = [ 25, 7, 5, 26, 11, 8, 25, 14, 23, 19,
                14, 11, 22, 29, 11, 13, 12, 17, 18, 10,
                24, 18, 25, 9, 3 ];

d3.select("body").selectAll("div")
    .data(dataset)
    .enter()
    .append("div")
    .attr("class", "bar")
    .style("height", function(d) {
        var barHeight = d * 5;
        return barHeight + "px";
    });
```



Flexibilität von .data()





Zufallsgenerator Math.random()

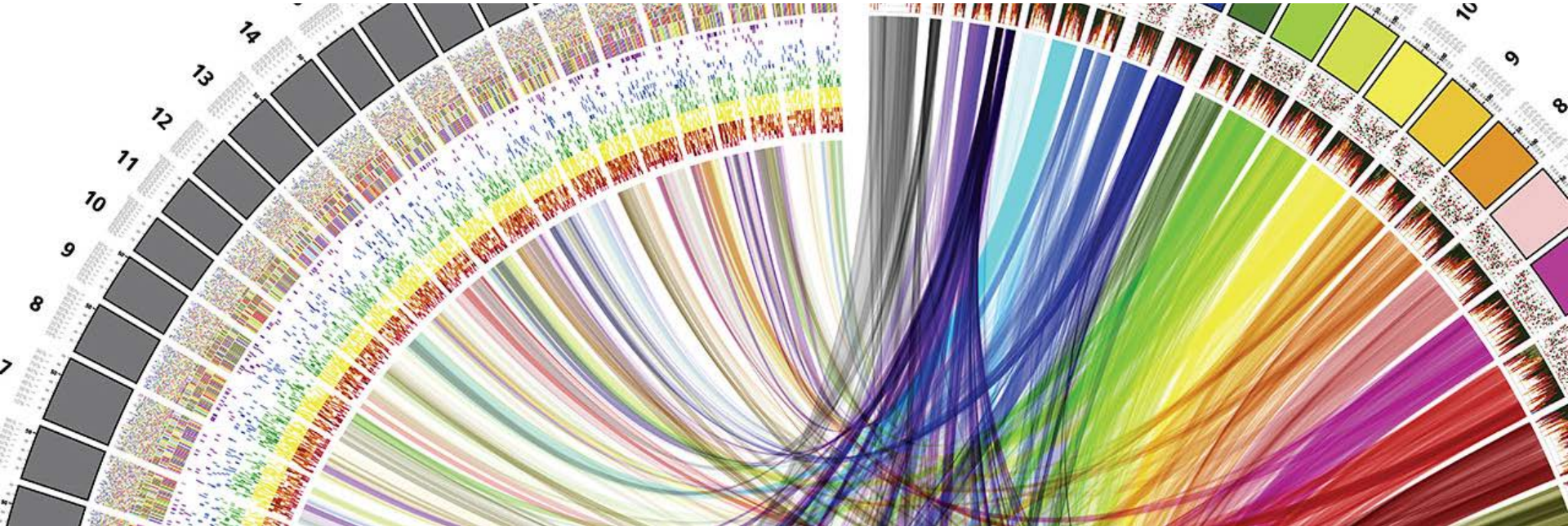
Zufallszahlen generieren:

```
var dataset = [];
for (var i = 0; i < 25; i++) {
    var newNumber = Math.random() * 30;
    dataset.push(newNumber);
}
```

UNIVERSITÄT BERN

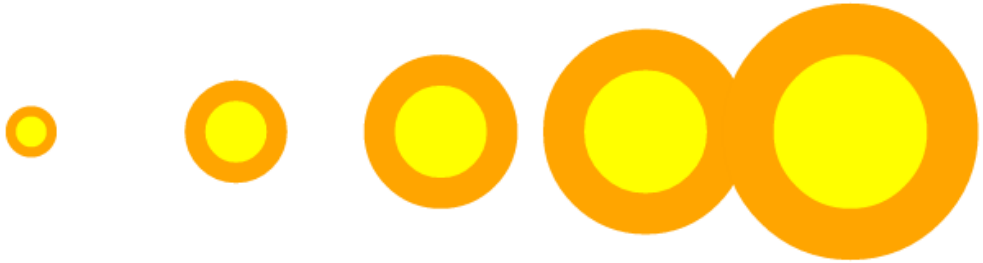
Agenda

1. Daten als Balken darstellen
2. **Daten als Kreise darstellen und positionieren**
3. Layouts

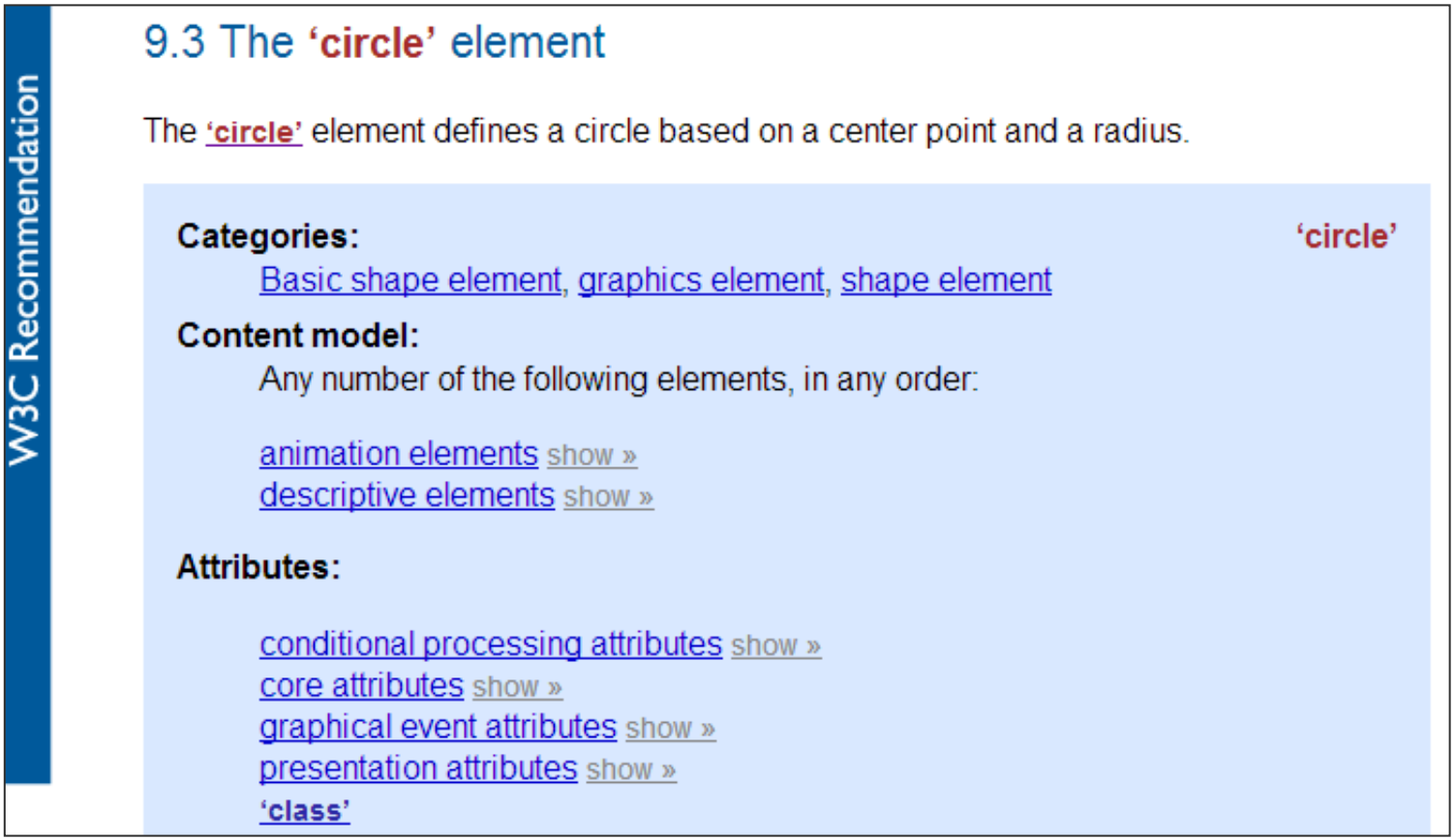


D3.js Circles Example

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Chapter 6 - Circles Example</title>
        <script type="text/javascript" src="/js/vendor/d3.min.js"></script>
    </head>
    <body>
        <script type="text/javascript">
   //Width and height
   var w = 500;
   var h = 100;

   //Data
   var dataset = [ 5, 10, 15, 20, 25 ];

   //Create SVG element
   var svg = d3.select("body")
      .append("svg")
      .attr("width", w)
      .attr("height", h);

   var circles = svg.selectAll("circle")
       .data(dataset)
       .enter()
       .append("circle");

   circles.attr("cx", function(d, i) {
          return (i * 50) + 25;
       })
       .attr("cy", h/2)
       .attr("r", function(d) {
          return d;
       })
       .attr("fill", "yellow")
       .attr("stroke", "orange")
       .attr("stroke-width", function(d) {
          return d/2;
       });
        </script>
    </body>
</html>
```



SVG Circle



W3C Recommendation

9.3 The 'circle' element

The 'circle' element defines a circle based on a center point and a radius.

'circle'

Categories:

Basic shape element, graphics element, shape element

Content model:

Any number of the following elements, in any order:

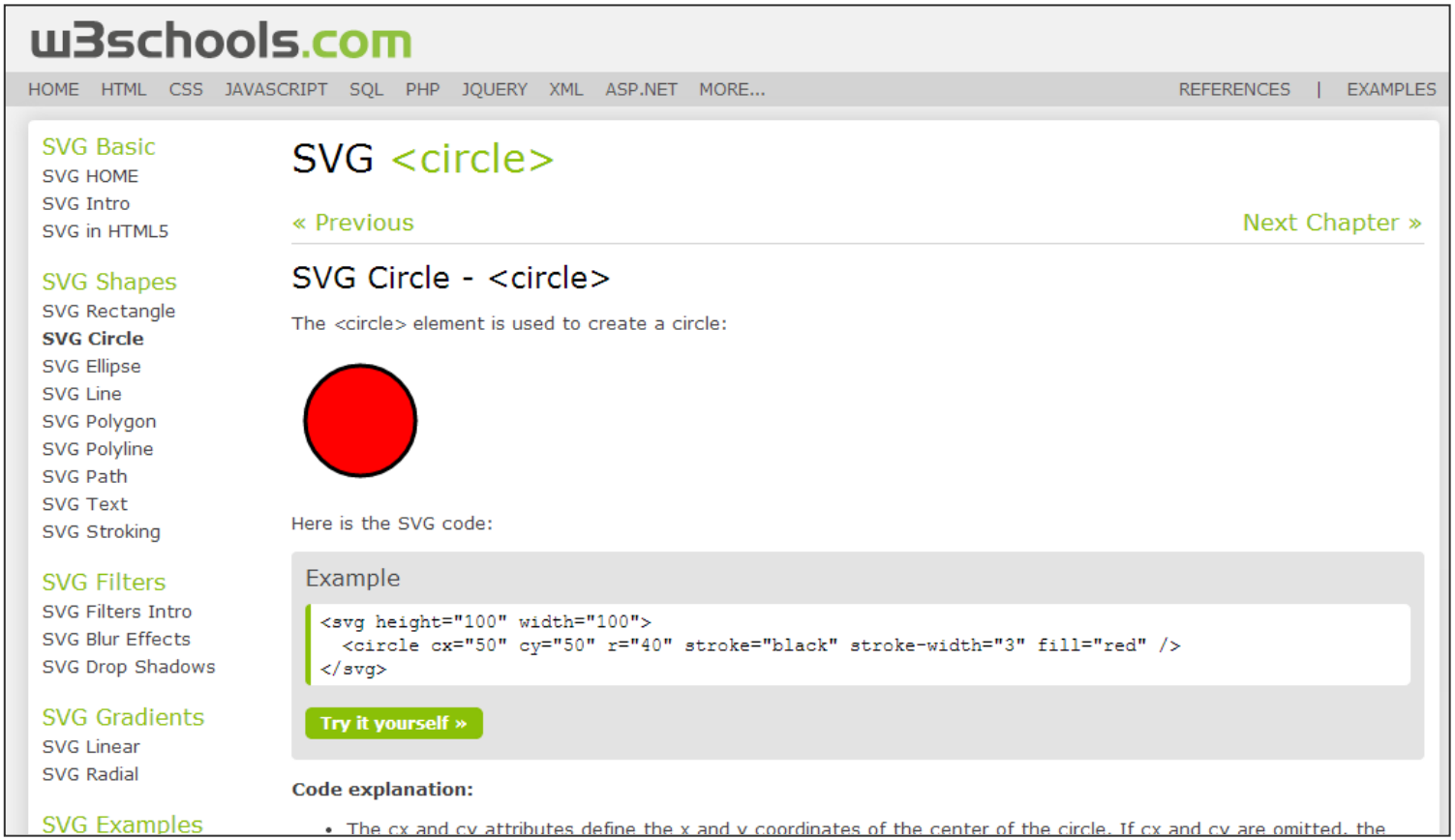
animation elements show »
descriptive elements show »

Attributes:

conditional processing attributes show »
core attributes show »
graphical event attributes show »
presentation attributes show »
'class'

Link: http://www.w3.org/TR/SVG11/shapes.html#CircleElement

SVG Circle



Link: http://www.w3schools.com/svg/svg_circle.asp

D3.js Circles Example

Grösse der SVG Fläche festlegen, Daten festlegen, SVG Fläche generieren:

```
//Width and height
var w = 500;
var h = 100;

//Data
var dataset = [ 5, 10, 15, 20, 25 ];

//Create SVG element
var svg = d3.select("body")
    .append("svg")
    .attr("width", w)
    .attr("height", h);
```

D3.js Circles Example

Kreise der SVG Fläche hinzufügen:

```
var circles = svg.selectAll("circle")
      .data(dataset)
      .enter()
      .append("circle");
```

Die Kreise werden generiert, aber noch ohne Attribute:

```
▼<svg width="500" height="100">
   <circle></circle>
   <circle></circle>
   <circle></circle>
   <circle></circle>
   <circle></circle>
 </svg>
```

D3.js Circles Example

Kreise immer 50 Pixel nach rechts verschoben mit **`cx`** positionieren:

```
circles.attr("cx", function(d, i) {
            return (i * 50) + 25;
        })
```

`d` sind die Daten (5, 10, 15, 20, 25)

`i` ist der Index des Datenpunkts (1, 2, 3, 4, 5)

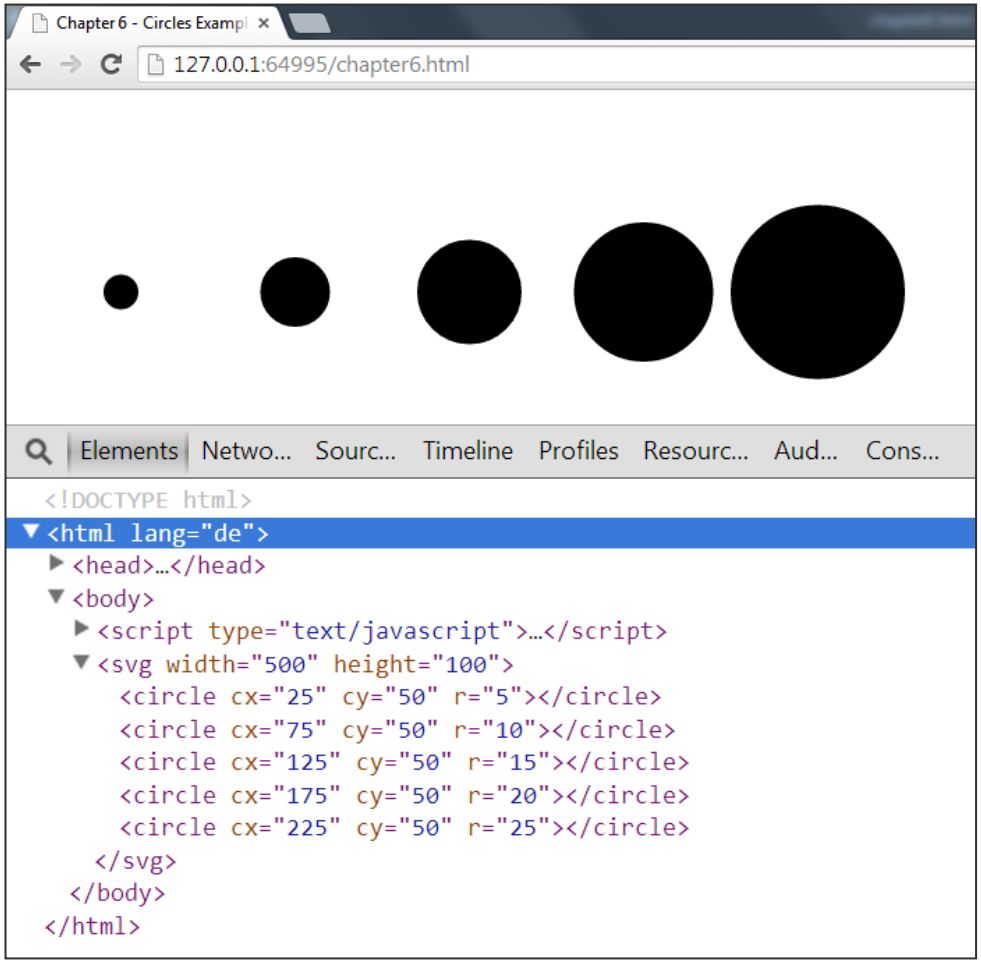
```
▼ <svg width="500" height="100">
    <circle cx="25"></circle>
    <circle cx="75"></circle>
    <circle cx="125"></circle>
    <circle cx="175"></circle>
    <circle cx="225"></circle>
  </svg>
```

D3.js Circles Example

Kreise vertikal in der Mitte der SVG-Fläche **h/2** positionieren und den Radius so gross wie den Datenwert **d** festlegen:

```
circles.attr("cx", function(d, i) {
            return (i * 50) + 25;
        })
        .attr("cy", h/2)
        .attr("r", function(d) {
            return d;
        });
```

D3.js Circles Example



D3.js Circles Example

Füllfarbe gelb, Randfarbe orange und Randdicke proportional zum Kreis:

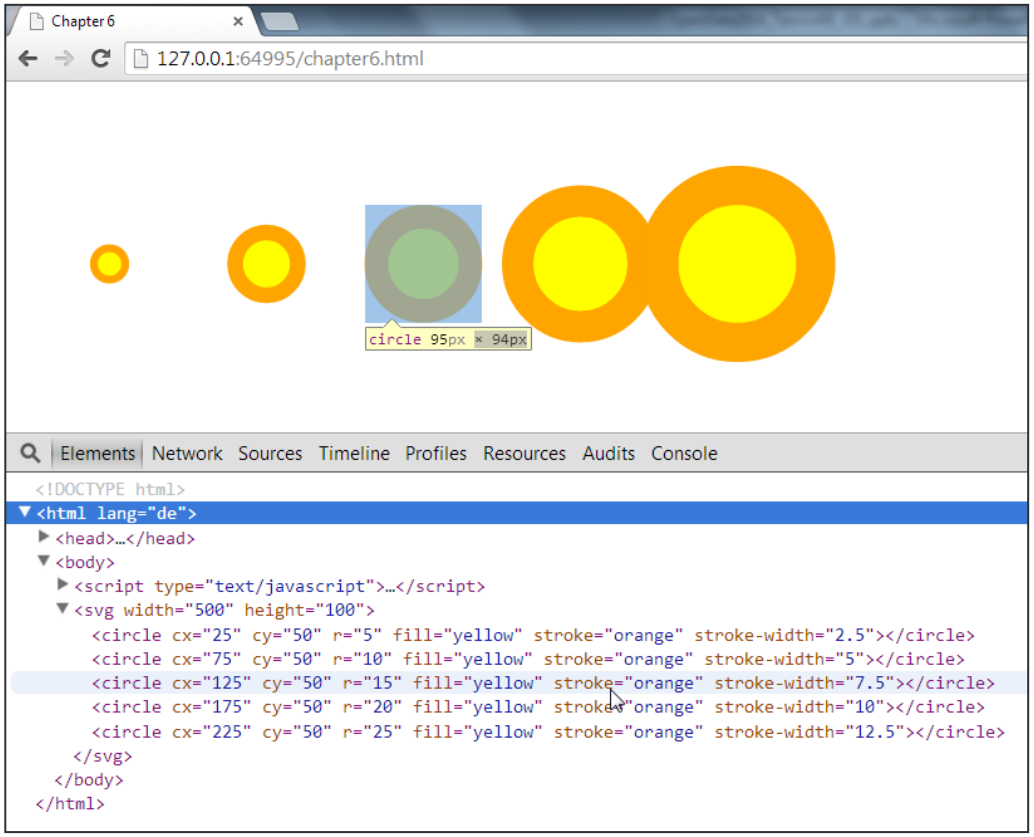
```
circles.attr("cx", function(d, i) {
            return (i * 50) + 25;
        })
        .attr("cy", h/2)
        .attr("r", function(d) {
            return d;
        })
        .attr("fill", "yellow")
        .attr("stroke", "orange")
        .attr("stroke-width", function(d) {
            return d/2;
        });
```

D3.js Circles Example

File Edit View Navigate Debug Help

```
<!DOCTYPE html>
<html lang="de">
    <head>
        <meta charset="utf-8">
        <title>Chapter 6 - Circles Example</title>
        <script type="text/javascript" src="/js/vendor/d3.min.js"></script>
    </head>
    <body>
        <script type="text/javascript">

            //Width and height
            var w = 500;
            var h = 100;

            //Data
            var dataset = [ 5, 10, 15, 20, 25 ];

            //Create SVG element
            var svg = d3.select("body")
                        .append("svg")
                        .attr("width", w)
                        .attr("height", h);

            var circles = svg.selectAll("circle")
                .data(dataset)
                .enter()
                .append("circle");

            circles.attr("cx", function(d, i) {
                        return (i * 50) + 25;
                    })
                    .attr("cy", h/2)
                    .attr("r", function(d) {
                        return d;
                    })
                    .attr("fill", "yellow")
                    .attr("stroke", "orange")
                    .attr("stroke-width", function(d) {
                        return d/2;
                    });

        </script>
    </body>
</html>
```

Line 1, Column 1 — 44 Lines



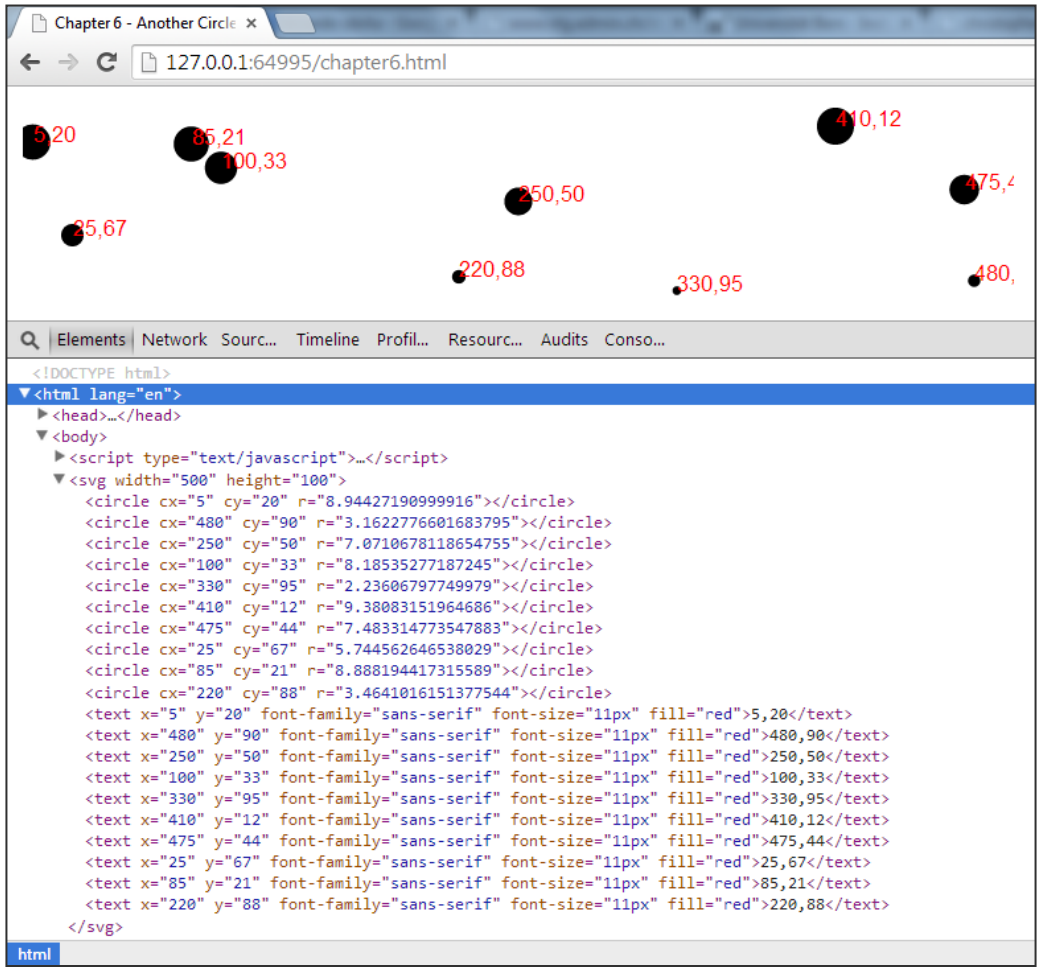
D3.js Circles Example

3 Aufgaben zum Üben:

1. Kreise horizontal proportional zu ihrer Grösse verteilen
2. Zufällige Grösse der Kreise erstellen
3. Datenwert pro Kreis anzeigen

Another D3.js Circles Example

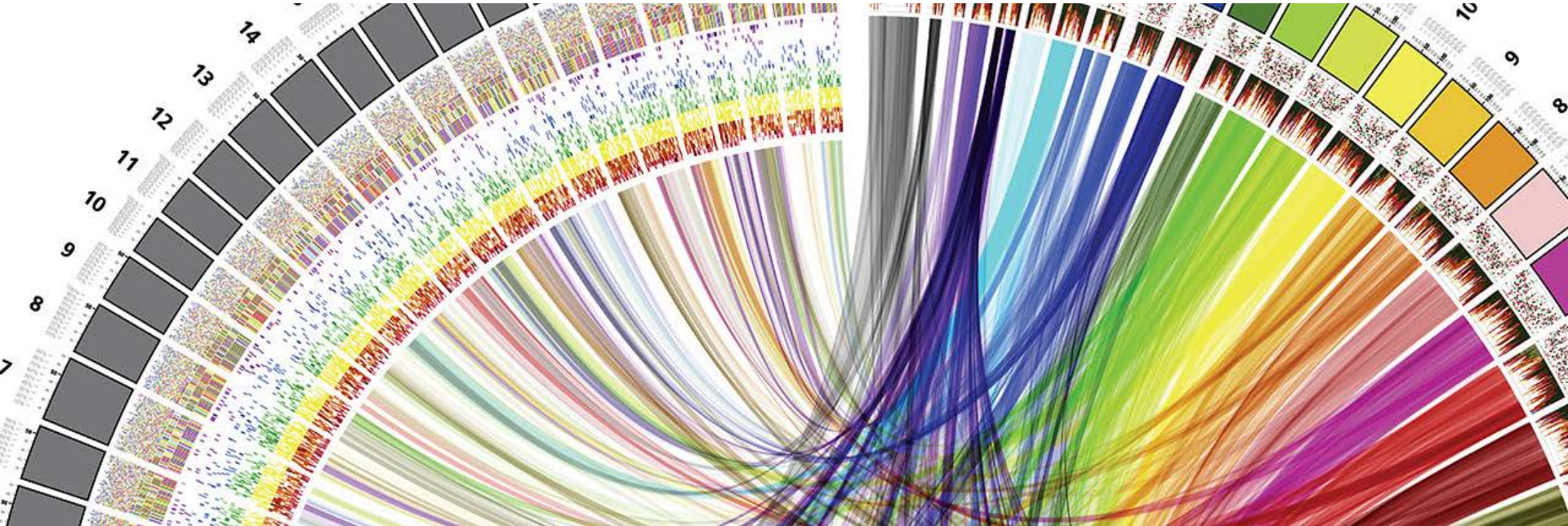
```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Chapter 6 - Another Circles Example</title>
        <script type="text/javascript" src="/js/vendor/d3.min.js"></script>
    </head>
    <body>
        <script type="text/javascript">

            var w = 500;
            var h = 100;
            var dataset = [
                          [5, 20], [480, 90], [250, 50], [100, 33], [330, 95],
                          [410, 12], [475, 44], [25, 67], [85, 21], [220, 88]
                        ];

            var svg = d3.select("body")
                        .append("svg")
                        .attr("width", w)
                        .attr("height", h);

            svg.selectAll("circle")
               .data(dataset)
               .enter()
               .append("circle")
               .attr("cx", function(d) {
                       return d[0];
               })
               .attr("cy", function(d) {
                       return d[1];
               })
               .attr("r", function(d) {
                       return Math.sqrt(h - d[1]);
               });

            svg.selectAll("text")
               .data(dataset)
               .enter()
               .append("text")
               .text(function(d) {
                       return d[0] + "," + d[1];
               })
               .attr("x", function(d) {
                       return d[0];
               })
               .attr("y", function(d) {
                       return d[1];
               })
               .attr("font-family", "sans-serif")
               .attr("font-size", "11px")
               .attr("fill", "red");

        </script>
    </body>
</html>
```



UNIVERSITÄT BERN

Agenda

1. Daten als Balken darstellen
2. Daten als Kreise darstellen und positionieren
3. **Layouts**

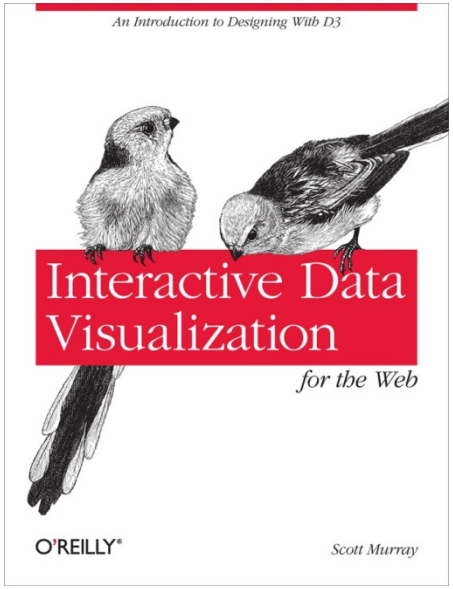


Interactive Data Visualization for the Web

Chapter 11. Layouts:

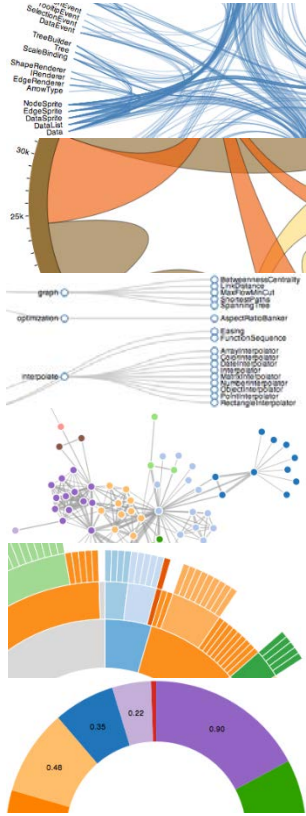
> *D3 layouts take data that you provide and remap or otherwise transform it, thereby generating new data that is more convenient for a specific visual task*

> http://chimera.labs.oreilly.com/books/1230000000345/ch11.html



D3.js Layouts

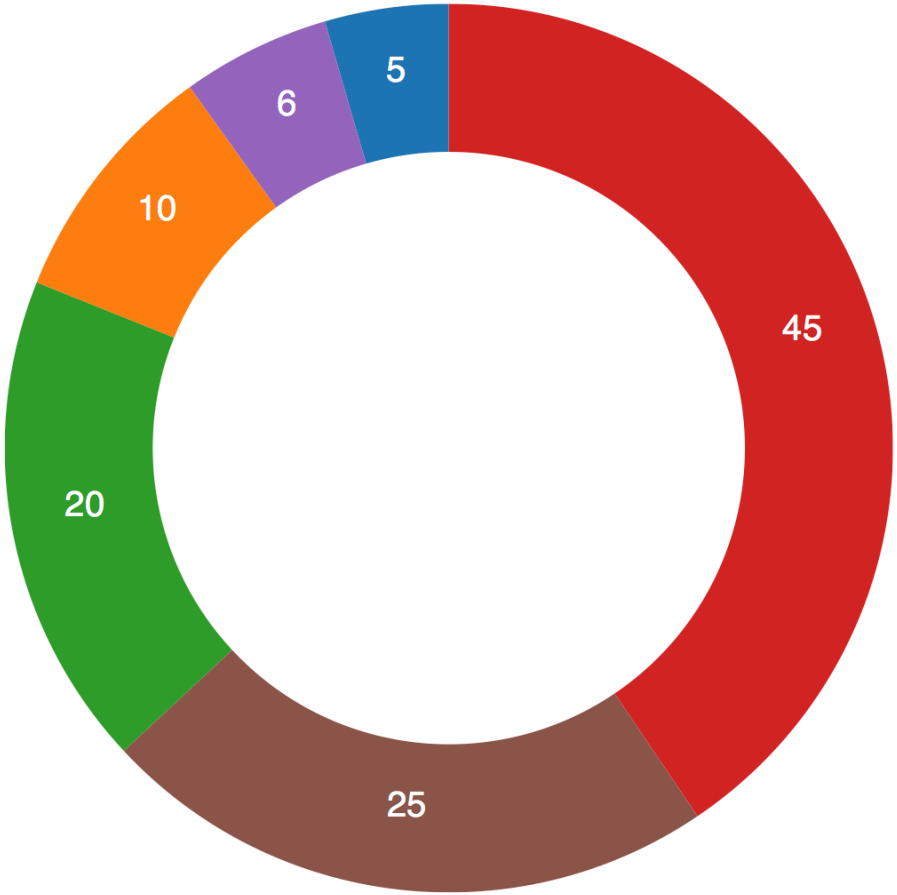
- > Bundle - apply Holten's hierarchical bundling algorithm to edges.
- > Chord - produce a chord diagram from a matrix of relationships.
- > Cluster - cluster entities into a dendrogram.
- > Force - position linked nodes using physical simulation.
- > Hierarchy - derive a custom hierarchical layout implementation.
- > Histogram - compute the distribution of data using quantized bins.
- > Pack - produce a hierarchical layout using recursive circle-packing.
- > Partition - recursively partition a node tree into a sunburst or icicle.
- > Pie - compute the start and end angles for arcs in a pie or donut chart.
- > Stack - compute the baseline for each series in a stacked bar or area chart.
- > Tree - position a tree of nodes tidily.
- > Treemap - use recursive spatial subdivision to display a tree of nodes.



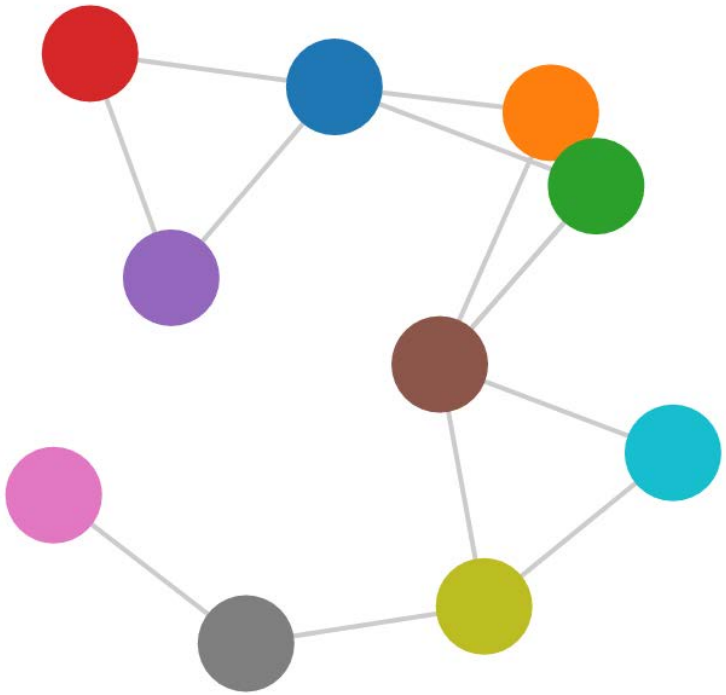
Link: https://github.com/mbostock/d3/wiki/Layouts

A simple ring chart

```
<!DOCTYPE html>
<html lang="en">
<head>
      <meta charset="utf-8">
      <title>D3: Pie layout ring chart</title>
      <script type="text/javascript" src="http://d3js.org/d3.v3.min.js"></script>
      <style type="text/css">
         text {
            font-family: sans-serif;
            font-size: 12px;
            fill: white;
         }
      </style>
   </head>
   <body>
      <script type="text/javascript">
         //Width and height
         var w = 300;
         var h = 300;
         var dataset = [ 5, 10, 20, 45, 6, 25 ];
         var outerRadius = w / 2;
         var innerRadius = w / 3;
         var arc = d3.svg.arc()
                     .innerRadius(innerRadius)
                     .outerRadius(outerRadius);
         var pie = d3.layout.pie();

         //Easy colors accessible via a 10-step ordinal scale
         var color = d3.scale.category10();
         //Create SVG element
         var svg = d3.select("body")
                  .append("svg")
                  .attr("width", w)
                  .attr("height", h);
         //Set up groups
         var arcs = svg.selectAll("g.arc")
                     .data(pie(dataset))
                     .enter()
                     .append("g")
                     .attr("class", "arc")
                  .attr("transform", "translate(" + outerRadius + "," + outerRadius + ")");
         //Draw arc paths
         arcs.append("path")
             .attr("fill", function(d, i) {
                return color(i);
             })
             .attr("d", arc);

         //Labels
         arcs.append("text")
             .attr("transform", function(d) {
                return "translate(" + arc.centroid(d) + ")";
             })
             .attr("text-anchor", "middle")
             .text(function(d) {
                return d.value;
             });

      </script>
   </body>
</html>
```



Force Layout



```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="utf-8">
      <title>D3: Force layout</title>
        <script type="text/javascript" src="http://d3js.org/d3.v3.min.js"></script>
      <style type="text/css">
         /* No style rules here yet */
      </style>
   </head>
   <body>
      <script type="text/javascript">

         //Width and height
         var w = 500;
         var h = 300;

         //Original data
         var dataset = {
            nodes: [
               { name: "Adam" },
               { name: "Bob" },
               { name: "Carrie" },
               { name: "Donovan" },
               { name: "Edward" },
               { name: "Felicity" },
               { name: "George" },
               { name: "Hannah" },
               { name: "Iris" },
               { name: "Jerry" }
            ],
            edges: [
               { source: 0, target: 1 },
               { source: 0, target: 2 },
               { source: 0, target: 3 },
               { source: 0, target: 4 },
               { source: 1, target: 5 },
               { source: 2, target: 5 },
               { source: 2, target: 5 },
               { source: 3, target: 4 },
               { source: 5, target: 8 },
               { source: 5, target: 9 },
               { source: 6, target: 7 },
               { source: 7, target: 8 },
               { source: 8, target: 9 }
            ]
         };

         //Initialize a default force layout, using the nodes and edges in dataset
         var force = d3.layout.force()
                           .nodes(dataset.nodes)
                           .links(dataset.edges)
                           .size([w, h])
                           .linkDistance([50])
                           .charge([-100])
                           .start();

         var colors = d3.scale.category10();

         //Create SVG element
         var svg = d3.select("body")
                  .append("svg")
                  .attr("width", w)
                  .attr("height", h);

         //Create edges as lines
         var edges = svg.selectAll("line")
            .data(dataset.edges)
            .enter()
            .append("line")
            .style("stroke", "#ccc")
            .style("stroke-width", 1);

         //Create nodes as circles
         var nodes = svg.selectAll("circle")
            .data(dataset.nodes)
            .enter()
            .append("circle")
            .attr("r", 10)
            .style("fill", function(d, i) {
               return colors(i);
            })
            .call(force.drag);

         //Every time the simulation "ticks", this will be called
         force.on("tick", function() {

            edges.attr("x1", function(d) { return d.source.x; })
                 .attr("y1", function(d) { return d.source.y; })
                 .attr("x2", function(d) { return d.target.x; })
                 .attr("y2", function(d) { return d.target.y; });

            nodes.attr("cx", function(d) { return d.x; })
                 .attr("cy", function(d) { return d.y; });

         });


      </script>
   </body>
</html>
```

Datensatz

Daten als **nodes** (Knoten) und **edges** (Kanten):

```
var dataset = {
        nodes: [
                { name: "Adam" },
                { name: "Bob" },
                { name: "Carrie" },
                { name: "Donovan" },
                { name: "Edward" },
                { name: "Felicity" },
                { name: "George" },
                { name: "Hannah" },
                { name: "Iris" },
                { name: "Jerry" }
        ],
        edges: [
                { source: 0, target: 1 },
                { source: 0, target: 2 },
                { source: 0, target: 3 },
                { source: 0, target: 4 },
                { source: 1, target: 5 },
                { source: 2, target: 5 },
                { source: 2, target: 5 },
                { source: 3, target: 4 },
                { source: 5, target: 8 },
                { source: 5, target: 9 },
                { source: 6, target: 7 },
                { source: 7, target: 8 },
                { source: 8, target: 9 }
        ]
};
```

Force Layout erstellen

Initialisieren des Force Layout:

```
var force = d3.layout.force()
                     .nodes(dataset.nodes)
                     .links(dataset.edges)
                     .size([w, h])
                     .linkDistance([50])      // Abstand zueinander
                     .charge([-100])          // Abstossende Kreise
                     .start();
```

Linien und Kreise zeichnen

SVG Linie für jede Kante zeichnen:

```
var edges = svg.selectAll("line")
        .data(dataset.edges)
        .enter()
        .append("line")
        .style("stroke", "#ccc")
        .style("stroke-width", 1);
```

SVG Kreis für jeden Knoten zeichnen:

```
var nodes = svg.selectAll("circle")
        .data(dataset.nodes)
        .enter()
        .append("circle")
        .attr("r", 10)
        .style("fill", function(d, i) {
                return colors(i);
        })
        .call(force.drag);
```

«Physikalischer» Effekt

Position von allen Elementen wird laufend aktualisiert, weil Netz zusammenhängt:

```
force.on("tick", function() {

    edges.attr("x1", function(d) { return d.source.x; })
         .attr("y1", function(d) { return d.source.y; })
         .attr("x2", function(d) { return d.target.x; })
         .attr("y2", function(d) { return d.target.y; });

    nodes.attr("cx", function(d) { return d.x; })
         .attr("cy", function(d) { return d.y; });

});
```

Force Layout in Bewegung

Koordinaten der Knoten und Kanten werden laufend aktualisiert: