Code Listing: layout

Max Mathys

February 4, 2016

1 tests/layout/index.html

```
<!DOCTYPE html>
<html lang="de-ch">
  <head>
      <meta charset="utf-8">
      <meta name="viewport" content=
      ↪  "width=device-width; initial-scale=1.0; maximum-scale=1.0; user-scalable=0;"
      ↪  >
      <meta name="description" content="Template">

      <title>Layout</title>

      <!-- Gemeinsames CSS, für alle Tests -->
      <link rel="stylesheet" href="/css/style.css" type="text/css">

      <!-- Testspezifisches CSS -->
      <link rel="stylesheet" href="style.css" type="text/css">
  </head>

  <body>
    <br>

    <!-- Titel, Auswahl Datensätze -->
    <div class="toggle container clearfix">
      <p class="caps toggle-title">Datensätze</p>
      <div id="select-row" class="mxn1">
      </div>
    </div>

    <!-- Visualisation -->
    <div id="visualization-container">
      <div id="visualization-wrap">
        <div id="display-overlay">
          <!-- Displays werden hier hineinkopiert -->
        </div>
        <svg id="visualization">
          <!-- Einfügen des SVG-Elements durch Javascript-Code -->
```

```
35          </svg>
36        </div>
37      </div>
38      <hr>
39
40      <!-- Auswahl der Interpolationen -->
41      <div class="container clearfix">
42        <h2>Interpolation</h2>
43        <select class="block mb1 field">
44          <option value="linear">Linear</option>
45          <option value="linear-closed">linear-closed (d3)</option>
46          <option value="step-before">step-before (d3)</option>
47          <option value="step-after">step-after (d3)</option>
48          <option value="basis">basis (d3)</option>
49          <option value="basis-open">basis-open (d3)</option>
50          <option value="basis-closed">basis-closed (d3)</option>
51          <option value="bundle">bundle (d3)</option>
52          <option value="cardinal">cardinal (d3)</option>
53          <option value="cardinal-open">cardinal-open (d3)</option>
54          <option value="cardinal-closed">cardinal-closed (d3)</option>
55          <option value="monotone">monotone (d3)</option>
56        </select>
57
58        <!-- Optionen für Anzeige -->
59        <label class="block col-12 mb2">
60          <input id="checkbox-points" type="checkbox">
61          Punkte anzeigen
62        </label>
63
64        <label class="block col-12 mb2">
65          <input id="checkbox-lines" type="checkbox" checked>
66          Linien anzeigen
67        </label>
68      </div>
69
70      <br><br>
71
72      <!-- JavaScript Bibliotheken -->
73      <script src="/js/d3.js"></script>
74      <script src="/js/jquery-2.1.4.min.js"></script>
75
76      <!-- JavaScript des Tests -->
77      <script src="script.js"></script>
78    </body>
79  </html>
```

2 tests/layout/meta.json

```
{
  "datasets": [
    {
      "url":"data/WWDI-AFG_EN_ATM_CO2E_KT.csv",
      "config":
      [
        {
          "row": "Date",
          "type": "index",
          "data_type": "Date",
          "date_format": "%Y-%m-%d",
          "name": "Datum"
        },
        {
          "row": "Value",
          "type": "value",
          "data_type": "Number",
          "name": "Afghanistan",
          "activated":true,
          "unit": "kt"
        }
      ]
    },
    {
      "url":"data/WWDI-AGO_EN_ATM_CO2E_KT.csv",
      "config":
      [
        {
          "row": "Date",
          "type": "index",
          "data_type": "Date",
          "date_format": "%Y-%m-%d",
          "name": "Datum"
        },
        {
          "row": "Value",
          "type": "value",
          "data_type": "Number",
          "name": "Angola",
          "activated":false,
          "unit":"kt"
        }
      ]
    },
    {
      "url":"data/WWDI-ARM_EN_ATM_CO2E_KT.csv",
      "config":
```

```
      [
        {
          "row": "Date",
          "type": "index",
          "data_type": "Date",
          "date_format": "%Y-%m-%d",
          "name": "Datum"
        },
        {
          "row": "Value",
          "type": "value",
          "data_type": "Number",
          "name": "Armenien",
          "activated":false,
          "unit":"kt"
        }
      ]
    },
    {
      "url":"data/WWDI-AUS_EN_ATM_CO2E_KT.csv",
      "config":
      [
        {
          "row": "Date",
          "type": "index",
          "data_type": "Date",
          "date_format": "%Y-%m-%d",
          "name": "Datum"
        },
        {
          "row": "Value",
          "type": "value",
          "data_type": "Number",
          "name": "Australien",
          "activated":false,
          "unit":"kt"
        }
      ]
    },
    {
      "url":"data/WWDI-BGD_EN_ATM_CO2E_KT.csv",
      "config":
      [
        {
          "row": "Date",
          "type": "index",
          "data_type": "Date",
          "date_format": "%Y-%m-%d",
          "name": "Datum"
```

```
        },
        {
          "row": "Value",
          "type": "value",
          "data_type": "Number",
          "name": "Bangladesh",
          "activated":false,
          "unit":"kt"
        }
      ]
    },
    {
      "url":"data/WWDI-BHR_EN_ATM_CO2E_KT.csv",
      "config":
      [
        {
          "row": "Date",
          "type": "index",
          "data_type": "Date",
          "date_format": "%Y-%m-%d",
          "name": "Datum"
        },
        {
          "row": "Value",
          "type": "value",
          "data_type": "Number",
          "name": "Bahrain",
          "activated":false,
          "unit":"kt"
        }
      ]
    },
    {
      "url":"data/WWDI-BLR_EN_ATM_CO2E_KT.csv",
      "config":
      [
        {
          "row": "Date",
          "type": "index",
          "data_type": "Date",
          "date_format": "%Y-%m-%d",
          "name": "Datum"
        },
        {
          "row": "Value",
          "type": "value",
          "data_type": "Number",
          "name": "Belarus",
          "activated":false,
```

```
            "unit":"kt"
          }
        ]
      },
      {
        "url":"data/WWDI-BLZ_EN_ATM_CO2E_KT.csv",
        "config":
        [
          {
            "row": "Date",
            "type": "index",
            "data_type": "Date",
            "date_format": "%Y-%m-%d",
            "name": "Datum"
          },
          {
            "row": "Value",
            "type": "value",
            "data_type": "Number",
            "name": "Belize",
            "activated":false,
            "unit":"kt"
          }
        ]
      },
      {
        "url":"data/WWDI-CHE_EN_ATM_CO2E_KT.csv",
        "config":
        [
          {
            "row": "Date",
            "type": "index",
            "data_type": "Date",
            "date_format": "%Y-%m-%d",
            "name": "Datum"
          },
          {
            "row": "Value",
            "type": "value",
            "data_type": "Number",
            "name": "Schweiz",
            "activated":false,
            "unit":"kt"
          }
        ]
      },
      {
        "url":"data/WWDI-CMR_EN_ATM_CO2E_KT.csv",
        "config":
```

```
      [
        {
          "row": "Date",
          "type": "index",
          "data_type": "Date",
          "date_format": "%Y-%m-%d",
          "name": "Datum"
        },
        {
          "row": "Value",
          "type": "value",
          "data_type": "Number",
          "name": "Kamerun",
          "activated":false,
          "unit":"kt"
        }
      ]
    },
    {
      "url":"data/WWDI-COL_EN_ATM_CO2E_KT.csv",
      "config":
      [
        {
          "row": "Date",
          "type": "index",
          "data_type": "Date",
          "date_format": "%Y-%m-%d",
          "name": "Datum"
        },
        {
          "row": "Value",
          "type": "value",
          "data_type": "Number",
          "name": "Kolumbien",
          "activated":false,
          "unit":"kt"
        }
      ]
    },
    {
      "url":"data/WWDI-COM_EN_ATM_CO2E_KT.csv",
      "config":
      [
        {
          "row": "Date",
          "type": "index",
          "data_type": "Date",
          "date_format": "%Y-%m-%d",
          "name": "Datum"
```

```
        },
        {
          "row": "Value",
          "type": "value",
          "data_type": "Number",
          "name": "Komoren",
          "activated":false,
          "unit":"kt"
        }
      ]
    },
    {
      "url":"data/WWDI-CYP_EN_ATM_CO2E_KT.csv",
      "config":
      [
        {
          "row": "Date",
          "type": "index",
          "data_type": "Date",
          "date_format": "%Y-%m-%d",
          "name": "Datum"
        },
        {
          "row": "Value",
          "type": "value",
          "data_type": "Number",
          "name": "Zypern",
          "activated":false,
          "unit":"kt"
        }
      ]
    },
    {
      "url":"data/WWDI-DEU_EN_ATM_CO2E_KT.csv",
      "config":
      [
        {
          "row": "Date",
          "type": "index",
          "data_type": "Date",
          "date_format": "%Y-%m-%d",
          "name": "Datum"
        },
        {
          "row": "Value",
          "type": "value",
          "data_type": "Number",
          "name": "Deutschland",
          "activated":false,
```

```
            "unit":"kt"
          }
        ]
      },
      {
        "url":"data/WWDI-DMA_EN_ATM_CO2E_KT.csv",
        "config":
        [
          {
            "row": "Date",
            "type": "index",
            "data_type": "Date",
            "date_format": "%Y-%m-%d",
            "name": "Datum"
          },
          {
            "row": "Value",
            "type": "value",
            "data_type": "Number",
            "name": "Dominica",
            "activated":false,
            "unit":"kt"
          }
        ]
      }
    ]
}
```

3 tests/layout/script.js

```
var tooltip = require('./modules/tooltip')
var line = require('./modules/line')
var sort = require('./modules/sort')
var points = require('./modules/points')
var id = require('./modules/id')
var format = require('./modules/format')
var filter = require('./modules/filter')
var domain = require('./modules/domain')
var toggle = require('./modules/toggle')

/* global d3, alert, £ */

/**
 *
 * Initialisierung Visualisation
 *
 *
```

```
 */

// Für die Visualisation benötigte Variablen

var config,           // Config-Array für _alle_ Elemente
  datasetsMeta,       // Das 'datasets'-Attribut von meta.json
  index,              // Config-Objekt für die Index-Spalte (X-Wert)
  values,             // Config-Array für Werte-Spalten (Y-Werte)
  v_accessor,         // Funktion, die den Werteaccessor zurückgibt
  v_accessor_cord,    // Funktion, die den Koordinatenaccessor zurückgibt
  v_accessor_scaled,  // Funktion, die den skalierten Wert zurückgibt.
  v_bundle,           // Objekt, das die drei v-Funktionen enthält.

  xScale,             // X-Skala
  yScale,             // Y-Skala
  xWertebereich,      // Bereich der X-Werte
  yWertebereich,      // Bereich der Y-Werte
  xAxis,              // X-Achse
  yAxis,              // Y-Achse

  w,                  // Breite der Visualisation
  h,                  // Höhe der Visualisation
  graphTransform,     // Verschiebung des Graphenbereichs

  showPoints,         // Gibt an, ob Punkte angezeigt werden sollen
  showLines           // Gibt an, ob die Linien angezeigt werden sollen

showPoints = false

/**
 * Laden der Konfigurationsdatei
 * @param  {[String]} "meta.json"             Der Dateiname für die
 *                                            Konfigurationsdatei
 * @param  {[Function]} function(err, config) Callback
 */
d3.json('meta.json', function (err, res) {
  if (err) {
    console.log(err)
    alert(err)
    return
  }

  config = []
  datasetsMeta = res.datasets

  index = {}
  values = []

  var colors = d3.scale.category20()
```

```
68
69    for (var i = 0; i < datasetsMeta.length; i++) {
70      var dataset = datasetsMeta[i]
71      var url = dataset.url
72
73      for (var j = 0; j < dataset.config.length; j++) {
74        var c = dataset.config[j]
75        c.url = url
76
77        // Generiere id
78        c.rowId = id.get(c)
79
80        config.push(c)
81
82        // Einfügen der Config in index oder values
83        if (c.type === 'index') {
84          index = c
85        } else if (c.type === 'value') {
86          // Spaltenspezifische Farbe generieren
87          c.color = colors(values.length + 1)
88
89          // Wenn das Attribut activated nicht gesetzt ist, setze es auch true.
90          if (typeof c.activated === 'undefined') {
91            c.activated = true
92          }
93          values.push(c)
94        }
95      }
96    // Bei unbekannten Typen: nicht in values oder index einfügen.
97    }
98
99    // Datentyp der Skalen festlegen
100   if (index.data_type === 'Number') {
101     xScale = d3.scale.linear()
102   } else if (index.data_type === 'Date') {
103     xScale = d3.time.scale()
104   }
105
106   if (values[0].data_type === 'Number') {
107     yScale = d3.scale.linear()
108   } else if (values[0].data_type === 'Date') {
109     yScale = d3.time.scale()
110   }
111
112   // Höhe und Breite des gesamten SVG-Elements definieren; Verschiebung des
113   // Graphs
114   w = 1100
115   h = 550
116
```

```
117     graphTransform = {xstart: 70, ytop: 0, xend: 0, ybottom: 50}
118
119     // Das Tooltip über die Transformation benachrichtigen
120     tooltip.opt.graphTransform = graphTransform
121
122
        ↪  // Wertebereich der Achsenskalierungen definieren. Hier ist die Anzahl der Pixel
123     // gemeint, über die sich die Achsen erstrecken. Die x-Achse und die y-Achse
124     // verschieben wir um 50 nach rechts, damit man die y-Achse beschriften kann.
125     xScale.range([0, w - graphTransform.xstart - graphTransform.xend])
126     yScale.range([h - graphTransform.ytop - graphTransform.ybottom, 0])
127
128     // Die Achsen werden von d3 generiert.
129     xAxis = d3.svg.axis().scale(xScale).orient('bottom')
130       .ticks(5)
131     yAxis = d3.svg.axis().scale(yScale).orient('left')
132       .ticks(5)
133       .innerTickSize(-w + graphTransform.xstart + graphTransform.xend)
134       .outerTickSize(2)
135
136     /**
137      *
138      *
139      * Accessors für die Daten
140      *
141      *
142      */
143
144     // Index-Accessor-Funktion: Gibt für eine bestimmte Datenreihe den Wert der
145       // Index-Spalte zurück.
146
147     index.accessor = function (d) {
148       return d[index.row]
149     }
150
151     // ..._scaled: Gibt den Skalierten Wert von accessor zurück.
152     index.accessor_scaled = function (d) {
153       return xScale(d[index.row])
154     }
155
156     // Funktion, die die Werte-Accessor-Funktion zurückgibt. Da sich die Werte-
157     // Accessor-Funktionen im Gegensatz zum statischen Index-Accessor unterschei-
158     // den, müssen sie für jede Spalte neu generiert werden. Diese Funktion ist
159     // dafür zuständig.
160
161     v_accessor = function (entry) {
162       return function (d) {
163         return d[entry.rowId]
164       }
```

```
165     }
166
167     v_accessor_scaled = function (entry) {
168       return function (d) {
169         return yScale(d[entry.rowId])
170       }
171     }
172
173     // Funktion, die den Koordinatenaccessor für die in entry angegebene Spalte
174     // zurückgibt.
175     v_accessor_cord = function (index, entry) {
176       return function (d) {
177         return [index.accessor_scaled(d), v_accessor_scaled(entry)(d)]
178       }
179     }
180
181     v_bundle = {
182       'raw': v_accessor,
183       'scaled': v_accessor_scaled,
184       'cord': v_accessor_cord
185     }
186
187     // Die Daten laden
188     loadFiles()
189   })
190
191   /**
192    *
193    *
194    * Laden der Daten
195    *
196    *
197    */
198
199   /**
200    * Die Funktion, die den Datensatz lädt und vorbereitet.
201    *
202    * Vorgehen:  1. Laden der Daten
203
       ↪    *                                                    2. Formatieren des Datensatzes (data_types
204
       ↪    *                                                    3. 'Mergen' mit den anderen Datensätzen, d
205    *                                                  4. Sortieren
206
       ↪    *                                                    5. Die gemergten Datensätze weitergeben
207    */
208   function loadFiles () {
209     // Anzahl von Dateien, die schon heruntergeladen wurden
210     var loaded = 0
```

```
211
212     // Die Variable für die gemergten Datensätze
213     var data = []
214
215     // Jedes einzelne File herunterladen (1)
216     for (var i = 0; i < datasetsMeta.length; i++) {
217       d3.csv(datasetsMeta[i].url, mkcb(i))
218     }
219
220     /**
221      * Funktion, die die Callback-Funktion für einen bestimmten Datensatz-Meta-
222      * daten-Objekt mit Index i zurückgibt. Siehe auch: MKCB-Problem.
223      * @param  {[Number]} i   Index des Datensatz-Metadaten-Objekts aus
224      *                        datasetsMeta.
225      * @return {[Function]}  Das generierte Callback, das nach dem Laden der
226      *                        Datei ausgeführt wird.
227      */
228     function mkcb (i) {
229       return function (err, resp) {
230         if (err) {
231           alert(err)
232           console.log(err)
233           return
234         }
235
236         // Formatieren (2)
237         resp = format.data_types(resp, datasetsMeta[i].config)
238         resp = format.ids(resp, datasetsMeta[i].config)
239
240         // Merge (3)
241         for (var j = 0; j < resp.length; j++) {
242           data.push(resp[j])
243         }
244
245         if (++loaded === datasetsMeta.length) {
246           // Alle Datein sind heruntergeladen worden und gemergt.
247
248           // Sortieren (4)
249           data = sort(data, index)
250
251           // Weitergeben (5)
252           loadVisualization(data)
253         }
254       }
255     }
256   }
257
258   /*
259    *
```

```
260   *
261   * Laden der Visualisation
262   *
263   *
264   */
265
266  /**
267   * Lädt die Visualisation
268   * @param  {[Array]} data Die gemergten Datensätze
269   */
270  function loadVisualization (data) {
271    /**
272     *
273     *  Achsen initalisieren (d3)
274     *
275     */
276
277    xWertebereich = domain.overflowX(data, index, 1.1)
278    yWertebereich = domain.overflowY(data, values, v_bundle, 1.1)
279    xScale.domain(xWertebereich)
280    yScale.domain(yWertebereich)
281
282    /**
283     *
284     * Zoom (d3)
285     *
286     */
287
288    // Zoom hinzufügen
289    var zoom = d3.behavior.zoom()
290      .x(xScale)
291      .y(yScale)
292      .scaleExtent([0.9, 50])
293      .on('zoom', draw)
294
295    // Die Variable graph initialiseren, damit sie in der Funktion zoomed() ver-
296    // wendet werden kann, obwohl sie erst später definiert wird.
297    var graph
298
299    /**
300     * Wird aufgerufen, sobald der Graph neu gezeichnet werden sollte.
301     */
302    function draw () {
303      // Achsen neu zeichnen
304      xAxisContainer.call(xAxis)
305      yAxisContainer.call(yAxis)
306
307      // Punkte neu berechnen.
308      for (var i = 0; i < values.length; i++) {
```

```
309         v.selectAll("circle.data-point[data-row='" + values[i].rowId + "']")
310           .attr('cx', index.accessor_scaled)
311           .attr('cy', v_accessor_scaled(values[i]))
312       }
313
314       // Tooltip und Linie aktualisieren
315       tooltip.updateAll(data, index, values, v_bundle, xScale, yScale)
316
317       line.updateAll(data, index, values, v_bundle)
318     }
319
320     /**
321      *
322      * Elemente einfügen
323      *
324      */
325
326     // SVG-Element mit id 'visualization' extrahieren aus html
327     var v = d3.select('#visualization')
328       .attr('width', w)
329       .attr('height', h)
330
331       // Unterstützung für Zoom hinzufügen
332       .call(zoom)
333
334     // SVG-Maske für den Graph: Wir wollen nicht, dass Punkte aus unserem
335     // definierten Feld auftauchen. Siehe Masken-Problem.
336     v.append('mask')
337       .attr('id', 'mask')
338       .append('rect')
339       .attr('x', 0)
340       .attr('y', 0)
341       .attr('width', w - graphTransform.xstart - graphTransform.xend)
342       .attr('height', h - graphTransform.ytop - graphTransform.ybottom)
343       .attr('fill', 'white')
344
345     // Container für die Visualisation hinzufügen und zu der Maske "linken"
346     // Transformation nach den definierten Angaben mit transform, translate
347     graph = v.append('g')
348       .attr('id', 'graph')
349       .attr('transform', 'translate(' + graphTransform.xstart +
350         ',' + graphTransform.ytop + ')')
351       .attr('mask', 'url(#mask)')
352
353     // Die Punkte zeichnen für jede Datenspalte
354     for (var i = 0; i < values.length; i++) {
355
       ↪  // Die Punkte einer Spalte haben für das Attribut data-row die generierte id
356       // (siehe Identifikations-Problem)
```

```
357       var circles = graph.selectAll("circle[data-row='" + values[i].rowId + "']")
358
359         // Aus dem gesamten gemergten Datensatz die Elemente extrahieren, die die
360         // entsprechende Reihe besitzen. Siehe Merge-Problem.
361         // Daten an Selektion binden: Alle Aktionen, die an diesem Element
362
            ↪  // ausgeführt werden, werden auch auf alle anderen Datenreihen ausgeführt.
363         .data(filter.row(data, values[i].rowId)).enter()
364
365       // Aktionen an datengebundener Selektion ausführen
366       circles.append('circle')
367         .attr('class', 'data-point')
368         .attr('data-row', values[i].rowId)
369         .attr('cx', index.accessor_scaled)
370         .attr('cy', v_accessor_scaled(values[i]))
371     }
372
373     // Sichtbarkeit der Punkte akualisieren
374     points.updateVisibility(values)
375
376     /**
377      *
378      * d3-Achsen einfügen
379      *
380      */
381
382     var xAxisContainer = v.append('g')
383       .attr('class', 'axis axis-x')
384       .attr('transform', 'translate(' +
385         graphTransform.xstart + ',' +
386         (h - graphTransform.ybottom) + ')')
387       .call(xAxis)
388
389     var yAxisContainer = v.append('g')
390       .attr('class', 'axis axis-y')
391       .attr('transform', 'translate(' + graphTransform.xstart + ',0)')
392       .call(yAxis)
393
394     /**
395      *
396      * Tooltip (nicht von d3, selber implementiert)
397      *
398      */
399
400
        ↪  // Maus-Koordinaten: Um auf die Maus-Koordinaten zugreifen zu können, muss man
401     // ein unsichtbares Element über den gesamten Graph legen, der alle
402     // 'Maus-Events' 'aufnimmt'. Ein leerer g-SVG-Container (wie 'graph') ist
403     // nicht fähig, Maus-Events aufzunehmen. Siehe Event-Problem.
```

```
404     v.append('rect')
405       .attr('id', 'overlay')
406       .attr('x', graphTransform.xstart)
407       .attr('y', graphTransform.ytop)
408       .attr('width', w - graphTransform.xstart - graphTransform.xend)
409       .attr('height', h - graphTransform.ytop - graphTransform.ybottom)
410       .on('mousemove', function () {
411         tooltip.mouse = d3.mouse(this)
412         tooltip.updateAll(data, index, values, v_bundle, xScale, yScale)
413       })
414
415     // Overlay für die Detailanzeige für Tooltip
416     d3.select('#display-overlay')
417       .attr('style', 'left: ' + graphTransform.xstart + 'px;' +
418         'top: ' + graphTransform.ytop + 'px;' +
419         'max-width: ' + (w - graphTransform.xstart - graphTransform.xend) + 'px;'
          ↪  +
420         'max-height: ' + (h - graphTransform.ytop - graphTransform.ybottom) +
          ↪  'px;')
421
422     /**
423      *
424      * Linien
425      *
426      */
427
428     // Für jede Datenspalte die Linie einfügen
429     for (var i = 0; i < values.length; i++) {
430       line.addLine(filter.row(data, values[i].rowId), index, values[i], v_bundle)
431     }
432
433     // Falls der Interpolationsmodus wechselt: Neuen Modus setzen und Linien
434     // aktualisieren.
435     $('select').on('change', function () {
436       line.mode = this.value
437       line.updateAll(data, index, values, v_bundle)
438     })
439
440     // Falls die Checkbox für die Sichtbarkeit der Punkte angeklickt wird:
441     // Sichtbarkeit akutalisieren.
442     $('#checkbox-points').on('change', function () {
443       if ($(this).is(':checked')) {
444         showPoints = true
445       } else {
446         showPoints = false
447       }
448       points.visible = showPoints
449       points.updateVisibility(values)
450     })
```

```
451
452   $('#checkbox-lines').on('change', function () {
453     if ($(this).is(':checked')) {
454       showLines = true
455     } else {
456       showLines = false
457     }
458     line.lineVisibility(showLines, values)
459   })
460
461   /**
462    *
463    * Toggles: Ein- und ausblenden von Datenreihen.
464    *
465    */
466
467   // Die Toggle-Elemente für jede Spalte generieren.
468   for (var i = 0; i < values.length; i++) {
469     toggle.add(data, index, values, values[i], v_bundle, zoom, yWertebereich,
       ↪  yScale, yAxis, draw)
470   }
471 }
```

4 tests/layout/style.scss

```
1  $graph-borders-color: rgba(0,0,0,.2);
2  $graph-axis-text-color: rgba(0,0,0,.6);
3  $graph-axis-font-size: 14px;
4  $graph-axis-size: 1.5px;
5  $graph-axis-tick-size: 1px;
6  $data-title-font-size: 16px;
7
8  #visualization-container {
9    display: flex;
10   justify-content: center;
11   margin: 0;
12 }
13
14 #visualization-wrap {
15   position: relative;
16 }
17
18 #display-overlay {
19   position:absolute;
20   display: inline-flex;
21   flex-wrap: nowrap;
22   pointer-events: none;
23   align-content: space-between;
```

```
24  }
25
26  .tip-element {
27    border-left-width: 3px;
28    border-left-style: solid;
29    border-left-color: #000;
30    padding: 2px 15px 2px 5px;
31  }
32
33  .tip-title {
34    font-size: 12px;
35    margin-bottom: 2px;
36  }
37
38  .tip-attribute {
39    font-size: 12px;
40    margin: 0;
41  }
42
43  #overlay {
44    // Die Applikation soll die auf Bewegung der Maus zugreifen können. Dieses
45    // Attribut wird gebraucht.
46    pointer-events: all;
47  }
48
49  #graph {
50
51  }
52
53  .domain {
54    fill: none;
55    stroke: $graph-borders-color;
56    stroke-width: $graph-axis-size;
57  }
58
59  .tick > line {
60    fill: none;
61    stroke: $graph-borders-color;
62    stroke-width: $graph-axis-tick-size;
63
64  }
65
66  .tick > text {
67    fill: $graph-axis-text-color;
68    font-size: $graph-axis-font-size;
69  }
70
71  circle.data-point {
72    r: 2px;
```

```
}

circle#tooltip-circle {
  r: 7px;
  fill: rgba(0, 0, 0, .4);
}

#label > text {
  fill: black;
  font-weight: 400;
  font-size: $graph-axis-font-size;
}

#overlay {
  visibility: hidden;
}

.line {
  fill: none;
  stroke: rgba(0,0,0,.7);
  stroke-width: 2px;
}

.hidden {
  display: none;
}

#select-row {
  padding: 0 0 30px;
  margin: 0;
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
}

.select-row-item {
  color: #000;
  padding-bottom: 6px;
  margin: 12px 20px 0 0;
  cursor: pointer;
  border-bottom: 2px solid rgba(0, 0, 0, 1);
  border-color: #000;

  transition: all 0.2s cubic-bezier(.5,1,.8,1);

  user-select: none;
}

.select-row-item:hover {
```

```
    color: black;
    text-decoration: none;
}

.inactive {
    color: LightGray;
    border-color: #fff !important;
}

.toggle-title {
    margin-bottom: 8px;
}
```

5 tests/layout/data/WWDI-AFG_EN_ATM_CO2E_KT.csv

```
Date,Value
2010-12-31,8236.082
2009-12-31,6523.593
2008-12-31,3927.357
2007-12-31,1991.181
2006-12-31,1338.455
2005-12-31,1015.759
2004-12-31,733.4
2003-12-31,583.053
2002-12-31,359.366
2001-12-31,645.392
2000-12-31,781.071
1999-12-31,832.409
1998-12-31,1056.096
1997-12-31,1114.768
1996-12-31,1199.109
1995-12-31,1268.782
1994-12-31,1320.12
1993-12-31,1375.125
1992-12-31,1426.463
1991-12-31,2493.56
1990-12-31,2676.91
1989-12-31,2775.919
1988-12-31,2867.594
1987-12-31,3124.284
1986-12-31,3142.619
1985-12-31,3509.319
1984-12-31,2830.924
1983-12-31,2522.896
1982-12-31,2101.191
1981-12-31,1983.847
1980-12-31,1760.16
1979-12-31,2240.537
```

```
1978-12-31,2159.863
1977-12-31,2390.884
1976-12-31,1987.514
1975-12-31,2126.86
1974-12-31,1917.841
1973-12-31,1639.149
1972-12-31,1532.806
1971-12-31,1895.839
1970-12-31,1672.152
1969-12-31,942.419
1968-12-31,1224.778
1967-12-31,1283.45
1966-12-31,1092.766
1965-12-31,1008.425
1964-12-31,839.743
1963-12-31,707.731
1962-12-31,689.396
1961-12-31,491.378
1960-12-31,414.371
```

6 tests/layout/data/WWDI-AGO_EN_ATM_CO2E_KT.csv

```
Date,Value
2010-12-31,30417.765
2009-12-31,27836.197
2008-12-31,26596.751
2007-12-31,25151.953
2006-12-31,22266.024
2005-12-31,19156.408
2004-12-31,18793.375
2003-12-31,9064.824
2002-12-31,12665.818
2001-12-31,9732.218
2000-12-31,9541.534
1999-12-31,9156.499
1998-12-31,7308.331
1997-12-31,7381.671
1996-12-31,10491.287
1995-12-31,11012.001
1994-12-31,3890.687
1993-12-31,5801.194
1992-12-31,4418.735
1991-12-31,4367.397
1990-12-31,4429.736
1989-12-31,5009.122
1988-12-31,5130.133
1987-12-31,5815.862
1986-12-31,4660.757
```

```
1985-12-31,4701.094
1984-12-31,5009.122
1983-12-31,5115.465
1982-12-31,4649.756
1981-12-31,5280.48
1980-12-31,5346.486
1979-12-31,5504.167
1978-12-31,5412.492
1977-12-31,3534.988
1976-12-31,3285.632
1975-12-31,4415.068
1974-12-31,4873.443
1973-12-31,4880.777
1972-12-31,4506.743
1971-12-31,3410.31
1970-12-31,3582.659
1969-12-31,2786.92
1968-12-31,1672.152
1967-12-31,993.757
1966-12-31,1554.808
1965-12-31,1188.108
1964-12-31,1224.778
1963-12-31,1151.438
1962-12-31,1180.774
1961-12-31,454.708
1960-12-31,550.05
```

7 tests/layout/data/WWDI-ARM_EN_ATM_CO2E_KT.csv

```
Date,Value
2010-12-31,4220.717
2009-12-31,4352.729
2008-12-31,5555.505
2007-12-31,5067.794
2006-12-31,4382.065
2005-12-31,4360.063
2004-12-31,3644.998
2003-12-31,3428.645
2002-12-31,3043.61
2001-12-31,3542.322
2000-12-31,3465.315
1999-12-31,3058.278
1998-12-31,3406.643
1997-12-31,3278.298
1996-12-31,2607.237
1995-12-31,3490.984
1994-12-31,2966.603
1993-12-31,2896.93
```

```
1992-12-31,4052.035
```

8 tests/layout/data/WWDI-AUS_EN_ATM_CO2E_KT.csv

```
Date,Value
2010-12-31,373080.58
2009-12-31,395093.581
2008-12-31,387634.903
2007-12-31,377235.291
2006-12-31,371214.077
2005-12-31,362684.635
2004-12-31,348757.369
2003-12-31,346476.495
2002-12-31,341001.664
2001-12-31,324859.53
2000-12-31,329604.628
1999-12-31,325523.257
1998-12-31,346912.868
1997-12-31,333623.66
1996-12-31,329259.93
1995-12-31,307433.946
1994-12-31,303957.63
1993-12-31,302116.796
1992-12-31,294456.433
1991-12-31,281530.258
1990-12-31,287331.452
1989-12-31,277771.583
1988-12-31,261145.405
1987-12-31,256106.947
1986-12-31,239964.813
1985-12-31,241229.928
1984-12-31,236594.84
1983-12-31,225003.453
1982-12-31,234119.615
1981-12-31,230360.94
1980-12-31,220746.066
1979-12-31,205069.641
1978-12-31,202015.03
1977-12-31,187787.07
1976-12-31,174244.839
1975-12-31,175883.988
1974-12-31,172356.334
1973-12-31,170992.21
1972-12-31,157486.649
1971-12-31,152774.554
1970-12-31,147618.752
1969-12-31,142257.598
1968-12-31,134622.904
```

```
1967-12-31,129265.417
1966-12-31,120332.605
1965-12-31,120966.996
1964-12-31,108979.573
1963-12-31,101029.517
1962-12-31,94912.961
1961-12-31,90589.568
1960-12-31,88202.351
```

9 tests/layout/data/WWDI-BGD_EN_ATM_CO2E_KT.csv

```
Date,Value
2010-12-31,56152.771
2009-12-31,52328.09
2008-12-31,46435.221
2007-12-31,48466.739
2006-12-31,48136.709
2005-12-31,37553.747
2004-12-31,39750.28
2003-12-31,33883.08
2002-12-31,33707.064
2001-12-31,32456.617
2000-12-31,27869.2
1999-12-31,25236.294
1998-12-31,24048.186
1997-12-31,25063.945
1996-12-31,24029.851
1995-12-31,22816.074
1994-12-31,18969.391
1993-12-31,17407.249
1992-12-31,17748.28
1991-12-31,15940.449
1990-12-31,15533.412
1989-12-31,13454.223
1988-12-31,13545.898
1987-12-31,11862.745
1986-12-31,11463.042
1985-12-31,10234.597
1984-12-31,9123.496
1983-12-31,8236.082
1982-12-31,8599.115
1981-12-31,7931.721
1980-12-31,7638.361
1979-12-31,6648.271
1978-12-31,6017.547
1977-12-31,5812.195
1976-12-31,5570.173
1975-12-31,4869.776
```

```
1974-12-31,4660.757
1973-12-31,4554.414
1972-12-31,3509.319
```

10 tests/layout/data/WWDI-BHR_EN_ATM_CO2E_KT.csv

```
Date,Value
2010-12-31,24202.2
2009-12-31,24169.197
2008-12-31,24301.209
2007-12-31,22398.036
2006-12-31,19497.439
2005-12-31,19207.746
2004-12-31,17509.925
2003-12-31,16468.497
2002-12-31,15698.427
2001-12-31,13927.266
2000-12-31,18643.028
1999-12-31,18019.638
1998-12-31,18404.673
1997-12-31,17319.241
1996-12-31,15621.42
1995-12-31,16021.123
1994-12-31,15254.72
1993-12-31,15141.043
1992-12-31,10590.296
1991-12-31,11503.379
1990-12-31,11884.747
1989-12-31,11738.067
1988-12-31,12163.439
1987-12-31,11430.039
1986-12-31,11012.001
1985-12-31,10194.26
1984-12-31,9207.837
1983-12-31,8228.748
1982-12-31,9838.561
1981-12-31,8525.775
1980-12-31,7887.717
1979-12-31,8133.406
1978-12-31,7766.706
1977-12-31,7469.679
1976-12-31,6574.931
1975-12-31,5753.523
1974-12-31,5405.158
1973-12-31,5522.502
1972-12-31,3681.668
1971-12-31,3039.943
1970-12-31,2592.569
```

```
1969-12-31,1272.449
1968-12-31,1103.767
1967-12-31,1004.758
1966-12-31,649.059
1965-12-31,1228.445
1964-12-31,1598.812
1963-12-31,1195.442
1962-12-31,1591.478
1961-12-31,1771.161
1960-12-31,575.719
```

11 tests/layout/data/WWDI-BLR_EN_ATM_CO2E_KT.csv

```
Date,Value
2010-12-31,62221.656
2009-12-31,60292.814
2008-12-31,62815.71
2007-12-31,60281.813
2006-12-31,61829.287
2005-12-31,59064.369
2004-12-31,58037.609
2003-12-31,53721.55
2002-12-31,52390.429
2001-12-31,52595.781
2000-12-31,53468.527
1999-12-31,55819.074
1998-12-31,57597.569
1997-12-31,59519.077
1996-12-31,60157.135
1995-12-31,62118.98
1994-12-31,65914.325
1993-12-31,74451.101
1992-12-31,87487.286
```

12 tests/layout/data/WWDI-BLZ_EN_ATM_CO2E_KT.csv

```
Date,Value
2010-12-31,421.705
2009-12-31,414.371
2008-12-31,407.037
2007-12-31,425.372
2006-12-31,407.037
2005-12-31,396.036
2004-12-31,381.368
2003-12-31,374.034
2002-12-31,359.366
2001-12-31,711.398
2000-12-31,689.396
```

```
1999-12-31,601.388
1998-12-31,370.367
1997-12-31,388.702
1996-12-31,308.028
1995-12-31,377.701
1994-12-31,374.034
1993-12-31,377.701
1992-12-31,355.699
1991-12-31,359.366
1990-12-31,311.695
1989-12-31,300.694
1988-12-31,249.356
1987-12-31,227.354
1986-12-31,205.352
1985-12-31,190.684
1984-12-31,172.349
1983-12-31,172.349
1982-12-31,172.349
1981-12-31,183.35
1980-12-31,190.684
1979-12-31,209.019
1978-12-31,216.353
1977-12-31,198.018
1976-12-31,176.016
1975-12-31,176.016
1974-12-31,154.014
1973-12-31,146.68
1972-12-31,157.681
1971-12-31,143.013
1970-12-31,121.011
1969-12-31,135.679
1968-12-31,102.676
1967-12-31,121.011
1966-12-31,80.674
1965-12-31,84.341
1964-12-31,84.341
1963-12-31,62.339
1962-12-31,69.673
1961-12-31,36.67
1960-12-31,44.004
```

13 tests/layout/data/WWDI-CHE_EN_ATM_CO2E_KT.csv

```
Date,Value
2010-12-31,38756.523
2009-12-31,41598.448
2008-12-31,40392.005
2007-12-31,38019.456
```

```
2006-12-31,41877.14
2005-12-31,41374.761
2004-12-31,40392.005
2003-12-31,40204.988
2002-12-31,40718.368
2001-12-31,42962.572
2000-12-31,39049.883
1999-12-31,40700.033
1998-12-31,41829.469
1997-12-31,41466.436
1996-12-31,39929.963
1995-12-31,39233.233
1994-12-31,41319.756
1993-12-31,40762.372
1992-12-31,43065.248
1991-12-31,41998.151
1990-12-31,42863.563
1989-12-31,39467.921
1988-12-31,40711.034
1987-12-31,40252.659
1986-12-31,42295.178
1985-12-31,39827.287
1984-12-31,39174.561
1983-12-31,40069.309
1982-12-31,36629.663
1981-12-31,38859.199
1980-12-31,40538.685
1979-12-31,39911.628
1978-12-31,42218.171
1977-12-31,41085.068
1976-12-31,40483.68
1975-12-31,39097.554
1974-12-31,41481.104
1973-12-31,46262.872
1972-12-31,42973.573
1971-12-31,41943.146
1970-12-31,40296.663
1969-12-31,38096.463
1968-12-31,36028.275
1967-12-31,32559.293
1966-12-31,31525.199
1965-12-31,30370.094
1964-12-31,28166.227
1963-12-31,29101.312
1962-12-31,24150.862
1961-12-31,20388.52
1960-12-31,19523.108
```

14 tests/layout/data/WWDI-CMR_EN_ATM_CO2E_KT.csv

```
Date,Value
2010-12-31,7234.991
2009-12-31,6673.94
2008-12-31,5544.504
2007-12-31,5834.197
2006-12-31,3828.348
2005-12-31,3696.336
2004-12-31,3956.693
2003-12-31,3795.345
2002-12-31,3417.644
2001-12-31,3421.311
2000-12-31,3432.312
1999-12-31,3080.28
1998-12-31,3208.625
1997-12-31,3215.959
1996-12-31,4602.085
1995-12-31,4363.73
1994-12-31,3817.347
1993-12-31,3898.021
1992-12-31,3795.345
1991-12-31,1111.101
1990-12-31,1738.158
1989-12-31,7638.361
1988-12-31,2211.201
1987-12-31,1855.502
1986-12-31,2005.849
1985-12-31,6475.922
1984-12-31,6057.884
1983-12-31,6589.599
1982-12-31,6343.91
1981-12-31,5342.819
1980-12-31,3905.355
1979-12-31,1829.833
1978-12-31,2002.182
1977-12-31,1565.809
1976-12-31,1092.766
1975-12-31,1162.439
1974-12-31,975.422
1973-12-31,898.415
1972-12-31,861.745
1971-12-31,810.407
1970-12-31,638.058
1969-12-31,572.052
1968-12-31,506.046
1967-12-31,458.375
1966-12-31,344.698
1965-12-31,311.695
```

```
1964-12-31,337.364
1963-12-31,300.694
1962-12-31,289.693
1961-12-31,282.359
1960-12-31,271.358
```

15 tests/layout/data/WWDI-COL_EN_ATM_CO2E_KT.csv

```
Date,Value
2010-12-31,75679.546
2009-12-31,70850.107
2008-12-31,66438.706
2007-12-31,63439.1
2006-12-31,62940.388
2005-12-31,60945.54
2004-12-31,55071.006
2003-12-31,57421.553
2002-12-31,55661.393
2001-12-31,56273.782
2000-12-31,57923.932
1999-12-31,56512.137
1998-12-31,65976.664
1997-12-31,64909.567
1996-12-31,60527.502
1995-12-31,59614.419
1994-12-31,67571.809
1993-12-31,64022.153
1992-12-31,62049.307
1991-12-31,57120.859
1990-12-31,57337.212
1989-12-31,53233.839
1988-12-31,52445.434
1987-12-31,50487.256
1986-12-31,49101.13
1985-12-31,48378.731
1984-12-31,48980.119
1983-12-31,49449.495
1982-12-31,45914.507
1981-12-31,44458.708
1980-12-31,44356.032
1979-12-31,44433.039
1978-12-31,41682.789
1977-12-31,39354.244
1976-12-31,38089.129
1975-12-31,35896.263
1974-12-31,36479.316
1973-12-31,33699.73
1972-12-31,31481.195
```

```
1971-12-31,30326.09
1970-12-31,28404.582
1969-12-31,28048.883
1968-12-31,26604.085
1967-12-31,24715.58
1966-12-31,23501.803
1965-12-31,22885.747
1964-12-31,21708.64
1963-12-31,21257.599
1962-12-31,19442.434
1961-12-31,18217.656
1960-12-31,16409.825
```

16 tests/layout/data/WWDI-COM_EN_ATM_CO2E_KT.csv

```
Date,Value
2010-12-31,139.346
2009-12-31,124.678
2008-12-31,124.678
2007-12-31,121.011
2006-12-31,121.011
2005-12-31,110.01
2004-12-31,102.676
2003-12-31,99.009
2002-12-31,91.675
2001-12-31,88.008
2000-12-31,84.341
1999-12-31,80.674
1998-12-31,73.34
1997-12-31,66.006
1996-12-31,66.006
1995-12-31,80.674
1994-12-31,77.007
1993-12-31,77.007
1992-12-31,77.007
1991-12-31,77.007
1990-12-31,77.007
1989-12-31,62.339
1988-12-31,62.339
1987-12-31,58.672
1986-12-31,51.338
1985-12-31,55.005
1984-12-31,47.671
1983-12-31,47.671
1982-12-31,47.671
1981-12-31,47.671
1980-12-31,47.671
1979-12-31,22.002
```

```
1978-12-31,29.336
1977-12-31,40.337
1976-12-31,40.337
1975-12-31,33.003
1974-12-31,29.336
1973-12-31,29.336
1972-12-31,29.336
1971-12-31,29.336
1970-12-31,29.336
1969-12-31,18.335
1968-12-31,18.335
1967-12-31,18.335
1966-12-31,18.335
1965-12-31,14.668
1964-12-31,11.001
1963-12-31,11.001
1962-12-31,11.001
1961-12-31,11.001
1960-12-31,11.001
```

17 tests/layout/data/WWDI-CYP_EN_ATM_CO2E_KT.csv

```
Date,Value
2010-12-31,7708.034
2009-12-31,8140.74
2008-12-31,8555.111
2007-12-31,8195.745
2006-12-31,7788.708
2005-12-31,7502.682
2004-12-31,7334.0
2003-12-31,7748.371
2002-12-31,7022.305
2001-12-31,6846.289
2000-12-31,6849.956
1999-12-31,6475.922
1998-12-31,6391.581
1997-12-31,5885.535
1996-12-31,5764.524
1995-12-31,5551.838
1994-12-31,5636.179
1993-12-31,5485.832
1992-12-31,5276.813
1991-12-31,4741.431
1990-12-31,4653.423
1989-12-31,4341.728
1988-12-31,4129.042
1987-12-31,4121.708
1986-12-31,3545.989
```

```
1985-12-31,3102.282
1984-12-31,3182.956
1983-12-31,3098.615
1982-12-31,3102.282
1981-12-31,3047.277
1980-12-31,3208.625
1979-12-31,3003.273
1978-12-31,2808.922
1977-12-31,2669.576
1976-12-31,2464.224
1975-12-31,1980.18
1974-12-31,1925.175
1973-12-31,2500.894
1972-12-31,2405.552
1971-12-31,1884.838
1970-12-31,1705.155
1969-12-31,1672.152
1968-12-31,1624.481
1967-12-31,1364.124
1966-12-31,1243.113
1965-12-31,1151.438
1964-12-31,1001.091
1963-12-31,986.423
1962-12-31,905.749
1961-12-31,865.412
1960-12-31,887.414
```

18 tests/layout/data/WWDI-DEU_EN_ATM_CO2E_KT.csv

```
Date,Value
2010-12-31,745383.756
2009-12-31,732248.562
2008-12-31,783359.208
2007-12-31,784015.601
2006-12-31,808859.526
2005-12-31,806703.33
2004-12-31,825896.408
2003-12-31,833406.424
2002-12-31,828771.336
2001-12-31,853662.932
2000-12-31,829977.779
1999-12-31,822460.429
1998-12-31,855364.42
1997-12-31,862276.715
1996-12-31,889614.2
1995-12-31,864110.215
1994-12-31,865558.68
1993-12-31,877645.112
```

```
1992-12-31,891975.748
1991-12-31,929973.202
```

19 tests/layout/data/WWDI-DMA_EN_ATM_CO2E_KT.csv

```
Date,Value
2010-12-31,135.679
2009-12-31,128.345
2008-12-31,128.345
2007-12-31,150.347
2006-12-31,110.01
2005-12-31,113.677
2004-12-31,110.01
2003-12-31,113.677
2002-12-31,102.676
2001-12-31,113.677
2000-12-31,102.676
1999-12-31,80.674
1998-12-31,77.007
1997-12-31,80.674
1996-12-31,73.34
1995-12-31,80.674
1994-12-31,69.673
1993-12-31,62.339
1992-12-31,58.672
1991-12-31,58.672
1990-12-31,58.672
1989-12-31,58.672
1988-12-31,55.005
1987-12-31,47.671
1986-12-31,47.671
1985-12-31,47.671
1984-12-31,44.004
1983-12-31,40.337
1982-12-31,40.337
1981-12-31,36.67
1980-12-31,36.67
1979-12-31,33.003
1978-12-31,25.669
1977-12-31,25.669
1976-12-31,29.336
1975-12-31,29.336
1974-12-31,29.336
1973-12-31,25.669
1972-12-31,25.669
1971-12-31,25.669
1970-12-31,25.669
1969-12-31,18.335
```

```
1968-12-31,22.002
1967-12-31,22.002
1966-12-31,14.668
1965-12-31,14.668
1964-12-31,14.668
1963-12-31,14.668
1962-12-31,11.001
1961-12-31,11.001
1960-12-31,11.001
```

20 tests/layout/modules/domain.js

```
var range = require('./range')

/**
 * Modul: Domain
 * -------------
 * Gibt einen überhöhten Wertbereich zurück für x und x.
 * Überhöhte Wertebereiche werden hier benutzt, damit ein wenig Platz links und
 * oberhalb der Linie entsteht.
 */

/**
 * Gibt überhöhten Wertebereich für x zurück.
 * @param  {[Array]} data             Gemergter Datensatz, ungefiltert
 * @param  {[Object]} index           Config-Objekt für die Index-Spalte
 * @param  {[Number]} overflowFactor  Überhöhungsfaktor
 * @return {[Array]}                  Das Minimum und Maximum in einem Array.
 */
module.exports.overflowX = function (data, index, overflowFactor) {
  var xWertebereich = []
  xWertebereich[0] = range.min(data, index.accessor)
  xWertebereich[1] = range.max(data, index.accessor)
  xWertebereich[1] = range.applyOverflow(xWertebereich[0], xWertebereich[1],
    overflowFactor, index.data_type)
  return xWertebereich
}
/**
 * Gibt überhöhten Wertebereich für y zurück.
 * @param  {[Array]} data             Gemergter Datensatz, ungefiltert

 ↪   * @param  {[Array]} values           Array von Config-Objekten der Wertespalten
 * @param  {[Object]} v_bundle        Accessor-Bundle
 * @param  {[Number]} overflowFactor  Überhöhungsfaktor
 * @return {[Array]}                  Das Minimum und Maximum in einem Array.
 */
module.exports.overflowY = function (data, values, v_bundle, overflowFactor) {
  var yWertebereich = []
```

```
36    yWertebereich[0] = range.minMultipleSets(data, values, v_bundle)
37    yWertebereich[1] = range.maxMultipleSets(data, values, v_bundle)
38    yWertebereich[1] = range.applyOverflow(yWertebereich[0], yWertebereich[1],
39      overflowFactor, values[0].data_type)
40    return yWertebereich
41  }
```

21 tests/layout/modules/filter.js

```
1   /**
2    * Modul: filter
3    * -------------
4
     ↪    * Filtert den gemergten Datensatz. Gibt die Zeilen zurück, bei denen die Spalte
5    * 'row' gesetzt ist.
6    */
7
8   /**
9    * Gibt den gefilterten Datensatz zurück.
10   * @param  {[Array]} data Ungefilterter, gemergter Datensatz.
11   * @param  {[String]} row Name der Spalte, nach der gefiltert werden soll
12   * @return {[Array]}      Gefilterter, gemergter Datensatz.
13   */
14
15  module.exports.row = function (data, row) {
16    var ret = []
17    for (var i = 0; i < data.length; i++) {
18      if (typeof data[i][row] !== 'undefined') {
19        ret.push(data[i])
20      }
21    }
22    return ret
23  }
```

22 tests/layout/modules/format.js

```
1   var sort = require('./sort')
2   var id = require('./id')
3
4   /**
5    * Modul: Format
6    * -------------
7    * Formatiert den Datensatz
8    */
9
10  /**
11   * Konvertiert die Zeichenketten (Strings) in dem Datensatz in Javascript-
12   * Objekte, wie zum Beispiel Zahlen (Float) oder Daten (Date).
```

```
 * @param  {[Array]} data   Unformatierter Datensatz
 * @param  {{Array}} config Array von Config-Objekten
 * @return {[Array]}        Gefilterter Datensatz
 */
module.exports.data_types = function (data, config) {
  // index suchen
  for (var i = 0; i < data.length; i++) {
    for (var j = 0; j < config.length; j++) {
      if (config[j].data_type === 'Number') {
        data[i][config[j].row] = parseFloat(data[i][config[j].row])
      } else if (config[j].data_type === 'Date') {
        data[i][config[j].row] =  d3.time.format(config[j].date_format)
                                    .parse(data[i][config[j].row])
      }
    }
  }
  return data
}

/**
 * Fügt das Attribut 'rowId' für jedes Objekt hinzu. 'rowId' ist eine aus dem
 * Reihennamen und dem Pfad des Datensatzes generierte einzigartige ID.
 * @param  {[Array]} data   Datensatz ohne rowIds
 * @param  {{Array}} config Array von Config-Objekten
 * @return {[Array]}        Datensatz mit rowIds
 */
module.exports.ids = function (data, config) {
  for (var i = 0; i < data.length; i++) {
    for (var j = 0; j < config.length; j++) {
      if (config[j].type === 'index') {
        continue
      }
      data[i][id.get(config[j])] = data[i][config[j].row]
      delete data[i][config[j].row]
    }
  }

  return data
}
```

23 tests/layout/modules/id.js

```
/**
 * Modul: Id
 * ---------
 * Generiert einzigartige ID für eine Spalte.
 */

```

```
/**
 * Gibt generierte ID zurück.
 * @param  {[Object]} config Config-Objekt
 * @return {[String]}        ID
 */
module.exports.get = function (config) {
  return config.row + '#' + config.url
}

/**
 * Gibt Config-Objekt für eine Spalte zurück
 * @param  {[String]} id     ID der Spalte
 * @param  {[Array]} values Array von Config-Objekten aller Datenspalten
 * @return {[Object]}       Config-Objekt der Spalte
 */
module.exports.invert = function (id, values) {
  for (var i = 0; i < values.length; i++) {
    if (id === values[i].rowId) {
      return values[i]
    }
  }
}

/**
 * Gibt ID für benutzerdefinierte Attribute zurück.
 * @param  {[String]} attr Benutzerdefiniertes Attribut
 * @param  {[String]} url  URL des Datensatzes
 * @return {[String]}      ID
 */
module.exports.raw = function (attr, url) {
  return attr + '#' + url
}
```

24 tests/layout/modules/line.js

```
var filter = require('./filter')

/**
 * Modul: Line
 * -----------
 * Helfer-Funktionen für die Generierung von Linien
 */

/**
 * Modus für die Interpolation der Linie:
 *         - "undefined" oder "linear": Lineare Interpolation
 *         - Restliche: Modi, die von d3 unterstützt werden.
 * @type {String}
```

```
14   */
15  module.exports.mode = 'undefined'
16
17  /**
18   * Fügt eine Linie für die angegebnen Datenspalte hinzu.
19   * @param  {[Array]} data       Datensatz
20   * @param  {[Object]} index     Index-Config-Objekt
21   * @param  {[Array]} config     Array von Config-Objekten der Datenspalten
22   * @param  {[Object]} v_bundle  Accessors
23   */
24  module.exports.addLine = function (data, index, config, v_bundle) {
25    var path = d3.select('#graph')
26      .append('path')
27      .attr('class', 'line')
28      .attr('style', 'stroke:' + config.color)
29      .attr('data-row', config.rowId)
30
31    if (module.exports.mode === 'linear' || module.exports.mode === 'undefined')
      ↪  {
32      path.attr('d', linear(data, v_bundle.cord(index, config)))
33    }
34  }
35
36  /**
37   * Gibt die lineare Interpolation als SVG-Path-String zurück
38   * @param  {[Array]} data        Das Datenarray
39   * @param  {{Function}} accessor Die Funktion, welche die Koordinaten zurück-
40   *                               gibt des entsprechenden Punktes
41   * @return {[String]}            String, das in das Attribut 'd' im path-
42   *                               Element eingesetzt werden muss.
43   */
44  function linear (data, accessor) {
45    var path = ''
46
47    // figure out
48    var temp = 'Weighted Price'
49
50    for (var i = 0; i < data.length; i++) {
51      var coordinates = accessor(data[i], temp)
52
53      if (i !== 0) {
54        // L-Befehl für eine Linie
55        path += 'L' + coordinates[0] + ',' + coordinates[1]
56      } else {
57        // Erster Punkt: M-Befehl für Anfangspunkt.
58        path += 'M' + coordinates[0] + ',' + coordinates[1]
59      }
60
61      if (i !== data.length - 1) {
```

```
62          path += ' '
63        }
64      }
65      return path
66    }
67
68    /**
69     * Aktualisiert eine Linie.
70     * @param  {[Array]} data       Datensatz (gefiltert)
71     * @param  {[Object]} index     Index-Config-Objekt
72     * @param  {[Object]} config    Config-Objekt der Spalte
73     * @param  {[Object]} v_bundle  Accessor-Funktionen
74     */
75    module.exports.update = function (data, index, config, v_bundle) {
76      if (module.exports.mode === 'linear' || module.exports.mode === 'undefined')
        ↪ {
77        d3.select('.line[data-row="' + config.rowId + '"]')
78          .attr('d', linear(data, v_bundle.cord(index, config)))
79      } else {
80        var line = d3.svg.line()
81          .x(index.accessor_scaled)
82          .y(v_bundle.scaled(config))
83          .interpolate(module.exports.mode)
84
85        d3.select('.line[data-row="' + config.rowId + '"]')
86          .attr('d', line(data))
87      }
88    }
89
90    /**
91     * Ruft die Funktion update für alle Config-Objekte in values auf.
92     * @param  {[Array]} data       Datensatz (ungefiltert)
93     * @param  {[Object]} index     Index-Config-Objekt
94     * @param  {[Array]} values     Config-Array der Datenspalten
95     * @param  {[Object]} v_bundle  Accessor-Funktionen
96     */
97    module.exports.updateAll = function (data, index, values, v_bundle) {
98      for (var i = 0; i < values.length; i++) {
99        module.exports.update(filter.row(data, values[i].rowId), index, values[i],
          ↪  v_bundle)
100     }
101   }
102
103   /**
104    * Ein- oder Ausblenden einer Linie.
105    * @param  {[Boolean]} activated  true: Linie aktivieren;
106    *                                false: Linie ausblenden
107    * @param  {[Object]} config      Config-Objekt der Datenspalte
108    */
```

```
109 module.exports.setActivated = function (activated, config) {
110   var points_s = d3.selectAll(".data-point[data-row='" + config.rowId + "']")
111   var line = d3.selectAll(".line[data-row='" + config.rowId + "']")
112 
113   line.classed('hidden', !activated)
114 
115   config.activated = activated
116 
117   if (!activated) {
118     // Nicht aktiviert: Override
119     points_s.classed('hidden', !activated)
120   } else {
121     // Aktiviert: Zeigen, danach das Modul points entscheiden lassen.
122     points_s.classed('hidden', !activated)
123   }
124 }
125 
126 module.exports.lineVisibility = function (visible, values) {
127   // Verstecke alle Linien, falls visible. Sonst wende config.activated an.
128   for (var i = 0; i < values.length; i++) {
129     console.log('line with visibility = ', values[i].activated, '@', values[i].
      ↪  rowId)
130     var line = d3.selectAll('.line[data-row="' + values[i].rowId + '"]')
131     line.classed('hidden', visible ? !values[i].activated : true)
132   }
133   console.log('.')
134 }
```

25 tests/layout/modules/points.js

```
1  /**
2   * Modul: Points
3   * -------------
4   * Aus- und Einblenden von Punkten
5   */
6 
7  /**
8   * Sichtbarkeit der Punkte
9   * @type {Boolean}
10  */
11 module.exports.visible = false
12 
13 /**
14  * Sichtbarkeit der Punkte aktualisieren
15  * @param  {[Array]} values Array von Config-Objekten der Werte-Spalten
16  */
17 module.exports.updateVisibility = function (values) {
18   for (var i = 0; i < values.length; i++) {
```

```
19      var points = d3.selectAll('.data-point[data-row="' + values[i].rowId + '"]'
        ↪ )
20      if (module.exports.visible && values[i].activated) {
21        points.classed('hidden', false)
22      } else {
23        points.classed('hidden', true)
24      }
25    }
26  }
```

26 tests/layout/modules/range.js

```
1   /**
2    * Modul: Range
3    * ------------
4    * Wertebereich von Datenspalten bestimmen
5    */
6
7   /**
8    * Gibt das Minimum einer einzelnen Datenspalte zurück
9    * @param  {[Array]} data         Der Datensatz
10
    ↪   * @param  {{Function}} accessor  Der Accessor für die zu untersuchende Datenreihe
11   * @return {[Number]}             Das Minimum
12   */
13  module.exports.min = function (data, accessor) {
14    return d3.min(data, accessor)
15  }
16
17  /**
18   * Gibt das Maximum einer einzelnen Datenspalte zurück
19   * @param  {[Array]} data     Der Datensatz
20   * @param  {{Function}} index Der Accessor für die zu untersuchende Datenreihe
21   * @return {[Object]}         Das Maximum
22   */
23  module.exports.max = function (data, accessor) {
24    return d3.max(data, accessor)
25  }
26
27  /**
28   * Gibt das Minimum für mehrere Datenspalten zurück.
29   * @param  {[Array]} data           Der Datensatz
30   * @param  {[Array]} values         Der Config-Array für die zu untersuchenden
31   *                                                    Datenreihen.
32   * @param  {{Function}} v_accessor  Die Funktion, die für eine bestimmte value-
33   *                                  Reihe den Accessor zurückgibt.
34   * @return {[Object]}              Das Minimum
35   */
```

```
36  module.exports.minMultipleSets = function (data, values, v_bundle) {
37    var min
38    for (var i = 0; i < values.length; i++) {
39      if (!values[i].activated) {
40        continue
41      }
42      var lmin = d3.min(data, v_bundle.raw(values[i]))
43      if (typeof lmin === 'undefined') {
44        continue
45      }
46      if (typeof min === 'undefined' || lmin < min) {
47        min = lmin
48      }
49    }
50    return min
51  }
52
53  /**
54   * Gibt das Maximum für mehrere Datenspalten zurück.
55   * @param  {[Array]} data           Der Datensatz
56
     ↪    * @param  {[Array]} values         Der Config-Array für die zu untersuchenden Datenrei-
57   *                                 hen.
58   * @param  {{Function}} v_accessor  Die Funktion, die für eine bestimmte value-
59   *                                 Reihe den Accessor zurückgibt.
60   * @return {[Object]}              Das Maximum
61   */
62  module.exports.maxMultipleSets = function (data, values, v_bundle) {
63    var max
64    for (var i = 0; i < values.length; i++) {
65      if (!values[i].activated) {
66        continue
67      }
68      var lmax = d3.max(data, v_bundle.raw(values[i]))
69
70      if (typeof max === 'undefined' || lmax > max) {
71        max = lmax
72      }
73    }
74    return max
75  }
76
77  //  Wertebereich der Daten bestimmen mit d3: Um einen kleinen Abstand zwischen
78  //  den maximalen Punkten und den Rändern des Diagrammes zu bewahren,
79  //  wird der Unterschied (Δ) des Minimums und dem untersuchten Wert mit 1.1
80  //  mulitpliziert. Anschliessend wird die Summe des Minimums und des
81  //  multiplizierten Wertes an d3 zurückgegeben.
82
83  /**
```

```
84   * Gibt die Summe der Minimums und des mit dem Faktor factor multiplizierten
85   * Unterschied von min und max zurück.
86   * Wird verwendet, damit oben und rechts von Graphen Platz ausgelassen wird.
87   * @param  {[Number]} min       Minimum ohne Overflow
88   * @param  {[Number]} max       Maximum ohne Overflow
89   * @param  {[Number]} factor    Overflow-Faktor
90   * @param  {{String}} data_type Der Datentyp von min und max
91   * @return {[Number]}           Das Maximum mit Overflow.
92   */
93  module.exports.applyOverflow = function (min, max, factor, data_type) {
94    if (data_type === 'Date') {
95      return new Date(min.getTime() + (max.getTime() - min.getTime()) * factor)
96    } else if (data_type === 'Number') {
97      return min + (max - min) * factor
98    }
99  }
```

27 tests/layout/modules/sort.js

```
1   /**
2    * Modul: Sort
3    * -----------
4    * Sortiert einen Datensatz nach der Index-Spalte
5    */
6
7   /**
8    * Array sortieren, aufsteigend
9
     ↪    * https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sor
10   * @return {[Array]} Sortierter Datensatz
11   */
12  module.exports = function (data, index) {
13    data.sort(function (a, b) {
14      if (index.accessor(a) < index.accessor(b)) {
15        return -1
16      }
17      if (index.accessor(a) > index.accessor(b)) {
18        return 1
19      }
20      return 0
21    })
22
23    return data
24  }
```

28 tests/layout/modules/toggle.js

```
var line = require('./line')
var id = require('./id')
var points = require('./points')
var domain = require('./domain')

/**
 * Modul: Toggle
 * -------------
 * Steuerung der Sichtbarkeit von einzelnen Datenspalten
 */

/**
 * Toggle-Button hinzufügen
 * @param  {[Array]} data           Datensatz
 * @param  {[Object]} index         Index-Config-Objekt
 * @param  {[Array]} values         Config-Objekte der Wertespalten
 * @param  {[Object]} config        Config-Objekt der zu untersuchenden Spalte
 * @param  {[Object]} v_bundle      Accessors
 * @param  {[Object]} zoom          D3-Zoomobjekt
 * @param  {[Object]} yWertebereich D3-Wertebereich
 * @param  {[Object]} yScale        D3-Skalierung
 * @param  {[Object]} yAxis         D3-Achse
 * @param  {[Function]} draw        Funktion, die aufgerufen wird, wenn der
 *                                  Graph neu gezeichnet werden soll.
 */
module.exports.add = function (data, index, values, config, v_bundle, zoom,
 ↪  yWertebereich, yScale, yAxis, draw) {
  // Der Container für die Toggles hat die id select-row
  d3.select('#select-row')
    .append('p')
    .attr('class', 'select-row-item')
    .classed('inactive', !config.activated)
    // Spaltenspezifische Farbe hinzufügen
    .attr('data-row', config.rowId)
    // Falls der Name der Spalte in meta.json gesetzt ist, füge ihn ein.
    .text(config.name ? config.name : config.row)

  // Detail hinzufügen
  if (config.activated){
    addTooltipDetail(index, config)
  }

  line.setActivated(config.activated, config)
  if (config.activated) {
    points.updateVisibility(values)
  }

```

```
47    // Wenn die Toggle-Fläche angeklickt wird, aktualisiere die Sichtbarkeit
48    // der Linie.
49    $('.select-row-item[data-row="' + config.rowId + '"]').on('click', function
      ↪  () {
50      var row = $(this).attr('data-row')
51      var config = id.invert(row, values)
52
53      if ($(this).hasClass('inactive')) {
54        // Linie wird aktiviert.
55        $(this).toggleClass('inactive', false)
56        line.setActivated(true, config)
57        points.updateVisibility(values)
58        addTooltipDetail(index, config)
59      } else {
60        // Linie wird deaktiviert.
61        $(this).toggleClass('inactive', true)
62        line.setActivated(false, config)
63        removeTooltipDetail(config)
64      }
65
66      // und aktualisiere die y-Achse und Skalierung.
67      module.exports.updateYDomain(data, values, v_bundle, zoom, yWertebereich,
        ↪  yScale, yAxis, function () {
68        draw()
69      })
70    })
71  }
72
73  /**
74   * Funktion, um den Wertebereich und Skalierung bei einem Toggle zu
75   * aktualisieren
76   * @param  {[Array]} data           Datensatz
77   * @param  {[Array]} values         Config-Objekte der Wertespalten
78   * @param  {[Object]} v_bundle      Accessors
79   * @param  {[Object]} zoom          D3-Zoomobjekt
80   * @param  {[Object]} yWertebereich D3-Wertebereich
81   * @param  {[Object]} yScale        D3-Skalierung
82   * @param  {[Object]} yAxis         D3-Achse
83   * @param  {Function} callback      Funktion, die am Schluss aufgerufen wird.
84   */
85  module.exports.updateYDomain = function (data, values, v_bundle, zoom,
    ↪  yWertebereich, yScale, yAxis, callback) {
86    // Zoom zurücksetzen
87    zoom.scale(1)
88    zoom.translate([0, 0])
89
90    // y-Wertebereich und y-Skalierung aktualisieren.
91    yWertebereich = domain.overflowY(data, values, v_bundle, 1.1)
92    yScale.domain(yWertebereich)
```

```
93    zoom.y(yScale)
94    yAxis.scale(yScale)
95    callback()
96  }
97
98  /**
99   * Tooltip-Werteanzeige hinzufügen
100  * @param {[Object]} index  Index-Config-Objekt
101  * @param {[Object]} config Config-Objekt der Spalte
102  */
103 function addTooltipDetail(index, config) {
104   var container = d3.select('#display-overlay')
105     .append('div')
106     .attr('class', 'tip-element')
107     .attr('style', 'border-color:' + config.color)
108     .attr('data-row', config.rowId)
109
110   container.append('p')
111     .attr('class', 'tip-title caps')
112     .text(config.name ? config.name : config.row)
113
114   container.append('p')
115     .attr('class', 'tip-attribute')
116     .attr('data-attribute', index.name ? index.name : index.row)
117     .text((index.name ? index.name : index.row) + ': ')
118     .append('span')
119
120   container.append('p')
121     .attr('class', 'tip-attribute')
122     .attr('data-attribute', config.name ? config.name : config.row)
123     .text('Wert: ')
124     .append('span')
125 }
126
127 /**
128  * Tooltip-Werteanzeige entfernen
129  * @param  {[Object]} config Config-Objekt der Spalte
130  */
131 function removeTooltipDetail (config) {
132   d3.select('.tip-element[data-row="' + config.rowId + '"]')
133     .remove()
134 }
```

29 tests/layout/modules/tooltip.js

```
1  var filter = require('./filter')
2
3  /**
```

```
4   * Modul: Tooltip
5   * --------------
6   * Funktionen für den Tooltip
7   */
8
9   /**
10  * Einstellungen für dieses Modul.
11  *         - opt.graphTransform: Die Verschiebung des Graphen, wie in script.js
12  *                 definiert.
13  * @type {Object}
14  */
15  module.exports.opt = {}
16
17  /**
18
    ↪   * Findet den zu einem gegebenen Wert den nächsten in einem Array vorhandenem Wert.
19  * @param  {[Array]} data         Datenset
20  * @param  {[Function]} accessor  Funktion, das den zu vergleichenden Wert
21  *                                zurückgibt, wenn das Objekt gegeben wird.
22  * @param  {[type]} item          Der zu vergleichende Wert
23  * @return {[Number]}             Der Index (0 < @return < data.length-1)
24  */
25  function nextIndex (data, index, item) {
26    var pos = -1
27    for (var i = 0; i < data.length - 1; i++) {
28      // Liegt der Punkt zwischen zwei gegebenen Punkten?
29      var this_d = index.accessor(data[i])
30      var next = index.accessor(data[i + 1])
31      var afterThis = item >= this_d
32      var beforeNext = item <= next
33      var Δ1, Δ2
34
35      if (afterThis && beforeNext) {
36        // Falls ja, setze 'index' auf den index des näheren Punktes.
37        Δ1 = Math.abs(index.accessor(data[i]) - item)
38        Δ2 = Math.abs(index.accessor(data[i + 1]) - item)
39        pos = Δ1 < Δ2 ? i : i + 1
40      }
41    }
42    return pos
43  }
44
45  /**
46  * Funktion für den Tooltip-Kreis und die Werteanzeige
47  * @param  {[Array]} data           Datenarray
48  * @param  {{String}} rowId         Name der Reihe
49
    ↪   * @param  {[Function]} accessor    Funktion, die das Koordinatenpaar den Punktes
50  *                                    zurückgibt.
```

```
51
    ↪   * @param  {[Number]} index         Index des Datenarray, die den zu "tooltippenden"
52   *                                       Wert entspricht.
53
    ↪   * @param {{d3 View}} parent        d3-View, in das das Tooltip eingesetzt werden
54
    ↪   *                                                                          sollte.
55   * @param {{Function}} textAccessor Funktion, die den Text für das Tooltip zu-
56   *                                  rückgibt.
57   */
58  module.exports.tooltip = function (data, index, config, v_bundle, pos,
    ↪  indexTextAccessor, valueTextAccessor, activated) {
59    // tooltip-Variablen
60    var tip = d3.select('#tooltip[data-row="' + config.rowId + '"]')
61    tip.classed('hidden', !activated)
62
63    if (tip.empty()) {
64      tip = d3.select('#graph').append('g')
65        .attr('id', 'tooltip')
66        .attr('class', 'tooltip')
67        .attr('data-row', config.rowId)
68
69      tip.append('circle')
70        .attr('id', 'tooltip-circle')
71    }
72
73    var indexText
74    var valueText
75
76    if (pos === -1) {
77      tip.attr('visibility', 'hidden')
78      indexText = ''
79      valueText = ''
80    } else {
81      indexText = indexTextAccessor(data[pos])
82      valueText = valueTextAccessor(data[pos]) + (config.unit ? ' ' + config.unit
        ↪  : '')
83      tip.attr('visibility', 'visible')
84      var cord = v_bundle.cord(index, config)(data[pos])
85      tip.attr('transform', 'translate(' + cord[0] + ',' + cord[1] + ')')
86    }
87
88    d3.select('.tip-element[data-row="' + config.rowId + '"]' +
89      '> .tip-attribute[data-attribute="' + (index.name ? index.name : index.row)
        ↪  + '"]' +
90      '> span')
91      .text(indexText)
92
93    d3.select('.tip-element[data-row="' + config.rowId + '"]' +
```

```
94      '> .tip-attribute[data-attribute="' + (config.name ? config.name : config.
        ↪  row) + '"]' +
95      '> span')
96      .text(valueText)
97  }
98
99  /**
100  * Funktion, um den Ort des Tooltips neu zu berechnen (zum Beispiel wenn sich
101  * die Maus bewegt oder gezoomt wird).
102  * @param  {[Array]} data       Datensatz (Gefiltert)
103  * @param  {[Object]} index     Config-Objekt für den Index
104  * @param  {[Object]} config    Die Config für das Value-Objekt
105  * @param  {[Object]} v_bundle  Accessor-Bundle
106  * @param  {[Object]} xScale    x-Skalierung (d3)
107  * @param  {[Object]} yScale    y-Skalierung (d3)
108  */
109 module.exports.updateTooltip = function (data, index, config, v_bundle, xScale,
    ↪  yScale) {
110   var x = module.exports.mouse[0] - module.exports.opt.graphTransform.xstart
111
112   var x_date = xScale.invert(x)
113
114   var pos = nextIndex(data, index, x_date)
115
116   //  tooltip
117   module.exports.tooltip(data, index, config, v_bundle, pos, function (d) {
118     d = index.accessor(d)
119     if (d instanceof Date) {
120       var s = d.getDate().toString() + '/'
121       s += (d.getMonth() + 1).toString() + '/'
122       s += d.getFullYear().toString()
123       return s
124     }
125     return d.toString()
126   }, function (d) {
127     // Zahl runden
128
        ↪  // http://stackoverflow.com/questions/11832914/round-to-at-most-2-decimal-places-in-javas
129     var s = (Math.round(v_bundle.raw(config)(d) * 1000) / 1000).toString()
130     return s
131   }, config.activated)
132 }
133
134 /**
135  * - Ruft die Funktion 'updateTooltip' für alle Values auf.
136  * @param  {[Array]} data       Datensatz (ungefiltert)
137  * @param  {[Object]} index     Config-Objekt für den Index
138  * @param  {[Array]} values     Die Config für die Values
139  * @param  {[Object]} v_bundle  Accessor-Bundle
```

```
140   * @param  {[Object]} xScale    x-Skalierung (d3)
141   * @param  {[Object]} yScale    y-Skalierung (d3)
142   */
143 module.exports.updateAll = function (data, index, values, v_bundle, xScale,
    ↪  yScale) {
144   if (!module.exports.mouse) {
145     return
146   }
147   // updateTooltip aufrufen, Datensatz filtern.
148   for (var i = 0; i < values.length; i++) {
149     module.exports.updateTooltip(filter.row(data, values[i].rowId), index,
      ↪  values[i], v_bundle, xScale, yScale)
150   }
151 }
```