Diagramme - next generation: D3.js im Unternehmen nutzen

Autor: Carsten Czarski, ORACLE Deutschland B.V. & Co KG

Das freie Visualisierungs-Framework d3js (D3 = DDD = Data Driven Documents) gewinnt mehr und mehr Popularität.

Die Fülle an Diagrammen und Visualisierungen, die sich mit D3 realisieren lassen, grenzt wirklich ans Unglaubliche. Nach eine kurzen Einführung in die Technologie zeigt der Vortrag anhand der Oracle-Datenbank mit Oracle Application Express (APEX), wie man ein D3-Diagramm integrieren und Tabellendaten damit auf moderne Art und Weise visualisieren kann - die Nutzung in anderen Umgebungen ist von dort aus nur ein kleiner Schritt.

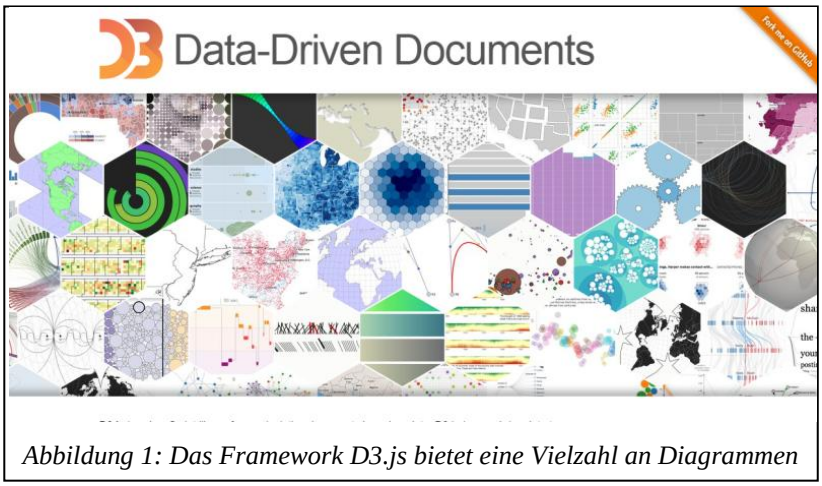

Abbildung 1: Das Framework D3.js bietet eine Vielzahl an Diagrammen

D3.js: Was ist das?

D3.js ist (ganz im Gegensatz zum in APEX integrierten AnyChart) keine Diagramm-Engine, bei der man einen Chart-Typ und eine Datenquelle wählt und dann fertig ist.

Vielmehr ist D3.js ein auf HTML5 basierendes Javascript-Framework, welches es dem Entwickler einen Rahmen anbietet, solche Diagramme *zu programmieren*. Um mit D3.js erfolgreich arbeiten zu können, braucht es solide Kenntnisse von Web-Technologien wie Javascript, HTML5 und SVG. Abschrecken lassen sollte man sich davon aber nicht: Die Möglichkeiten sind so vielfälig, dass es die Einarbeitung absolut wert ist.

D3.js-Diagramme werden im Browser aus HTML-Elementen zusammengesetzt. Die HTML5-Standard hinzugekommenen SVG-Elemente bieten eine sehr komfortable Grundlage - Abbildung 2 zeigt eine kleine Auswahl.

Typ	Code	Ergebnis
Linie	`<line x1="100" y1="10" x2="300" y2="40" style="stroke: red; stroke-width: 2px"/>`	
Rechteck	`<rect x="100" y="10" width="200" height="80" style="fill: red; stroke: black"/>`	
Kreis	`<circle cx="200" cy="50" r="40" style="stroke: black; fill: red"/>`	
Polygon	`<polygon points="100,10 200,90 300,10 100,10" style="stroke: black; fill: red"/>`	
Pfad	`<path id="P1" d="M 110 10 L 120 20 250 60 300 10 110 90" style="fill: none; stroke: red; stroke-width: 3px"/>`	

Abbildung 2: HTML5 erlaubt das Einbinden geometrischer Primitive in eine Webseite

Eine wichtige Eigenschaft der SVG-Unterstützung in HTML5 ist, dass die SVG-Objekte Teil des DOM im Browser werden. Das HTML-Markup wird also nicht nur interpretiert und dargestellt; vielmehr können die Objekte mit Javascript angesprochen und verändert werden (Abbildung 3).

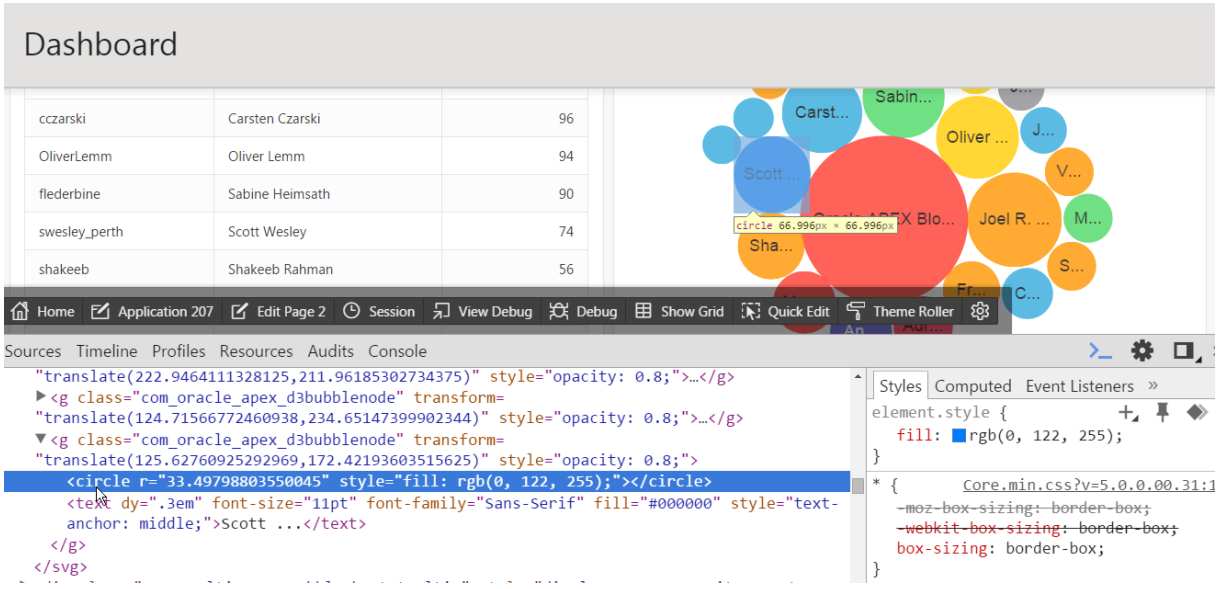

Abbildung 3: SVG-Objekte sind Teil des Browser-DOM und können mit Javascript angesprochen werden

Es lässt sich leicht erkennen, dass der in Abbildung 3 dargestellte Bubble Chart aus Instanzen des SVG-Elements <circle> zusammengesetzt wurde. Programmiert man ein Diagramm mit D3.js, so ist man nach wie vor für das Setzen der SVG-Elemente verantwortlich - man setzt keinen HTML-Code, vielmehr fügt man die Elemente mit

Javascript-Aufrufen zur Webseite hinzu. Das Framework D3.js spielt seine Stärken an zwei speziellen Punkten aus.

- D3.js verwaltet die dem Diagramm zugrundliegenden Daten und bietet dem Entwickler Werkzeuge, die das Verknüpfen der Daten mit den SVG-Elementen des Diagramms stark vereinfachen.
- D3.js liefert Layout-Klassen mit, welche die konkreten Parameter für die SVG-Elemente anhand der Daten berechnen – bei einem D3.js "Bubble Chart" kommt beispielsweise das *Pack-Layout* zum Einsatz – es berechnet für jeden Kreis die Koordinaten des Mittelpunkts und den Radius.

Der Autor von D3.js, Charles Bostock, hat neben dem Framework selbst auch eine Reihe von Beispielen ins Internet gestellt, anhand dieser kann man gut nachvollziehen, wie D3.js Charts funktionieren und wie man das Framework nutzt.

- D3 Bubble Chart (mit Code und Beispieldaten)
  http://bl.ocks.org/mbostock/4063269
- D3 Bar Chart (mit Code und Beispieldaten)
  http://bl.ocks.org/mbostock/3885304

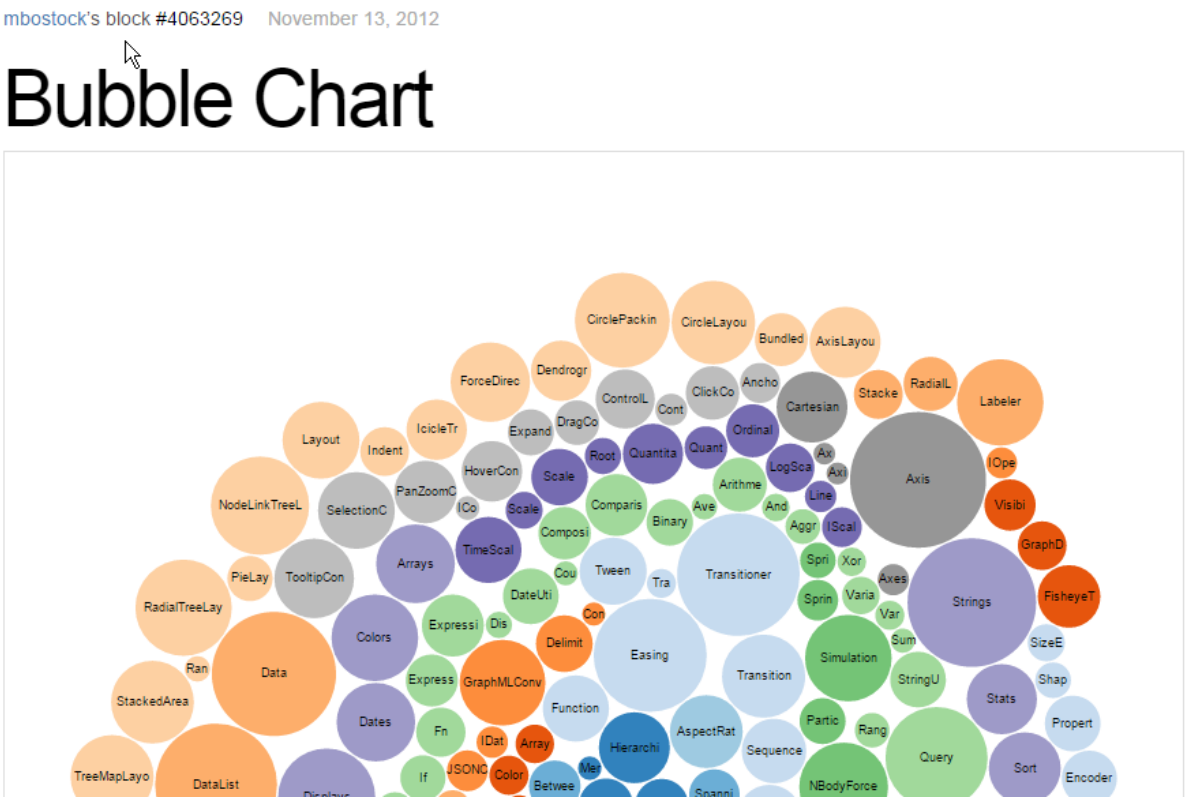

Abbildung 4: Ergebnis des Bubblechart-Beispiels von Charles Bostock

Schaut man sich das Bubblechart-Beispiel näher an, so erkennt man Javascript-Code und JSON-Daten.

 # flare.json


```
{
 "name": "flare",
 "children": [
  {
   "name": "analytics",
   "children": [
    {
     "name": "cluster",
     "children": [
      {"name": "AgglomerativeCluster", "size": 3938},
      {"name": "CommunityStructure", "size": 3812},
      {"name": "HierarchicalCluster", "size": 6714},
      {"name": "MergeEdge", "size": 743}
     ]
    },
    {
     "name": "graph",
     "children": [
      {"name": "BetweennessCentrality", "size": 3534},
      {"name": "LinkDistance", "size": 5731},
      {"name": "MaxFlowMinCut", "size": 7840},
      {"name": "ShortestPaths", "size": 5914},
      {"name": "SpanningTree", "size": 3416}
     ]
    }
```


Abbildung 5: JSON-Daten des D3.js Bubblechart Beispiels

An einer Stelle des Javascript-Codes werden die JSON-Daten mit der Funktion D3.json in den Browser-DOM geladen; danach stehen sie dem Entwickler zum Generieren eines Diagramms zur Verfügung.

```
var svg = d3.select("body").append("svg")
    .attr("width", diameter)
    .attr("height", diameter)
    .attr("class", "bubble");

d3.json("flare.json", function(error, root) {
  if (error) throw error;

  var node = svg.selectAll(".node")
      .data(bubble.nodes(classes(root))
      .filter(function(d) { return !d.children; }))
    .enter().append("g")
      .attr("class", "node")
      .attr("transform", function(d) { return "translate(" + d.x + "," + d.y + ")"; });
```

Abbildung 6: Laden der JSON-Daten in den Browser-DOM mit der Javascript-Funktion D3.json

Wie man an Abbildung 6 erkennen kann, ist die Datenquelle für den D3.js Programmierer lediglich eine URL, die (in diesem Fall) JSON zurückliefert. D3.js lässt sich also hervorragend mit REST-Services kombinieren: Der REST-Service liefert die JSON-Daten (Format beachten); der Javascript-Code baut das Diagramm.

Daten als REST-Service mit ORDS bereitstellen

Da REST-Webservices völlig unabhängig von einer konkreten Programmiersprache oder einem Entwicklerframework sind, lassen sich D3.js Diagramme im Grunde genommen sogar nur mit einem Browser realisieren. Alles, was man bereitstellen muss, ist ein REST-Service, der die Daten liefert.

Liegen die Daten in der Oracle-Datenbank, bieten sich die **Oracle REST Data Services (ORDS)** an. Ab Release 3.0 können REST-Services sogar ohne Application Express und allein mit PL/SQL definiert werden. Nachdem ORDS erfolgreich installiert und eine Datenbankverbindung eingerichtet wurde, lässt sich ein REST-Service, der Daten aus der Tabelle EMP liefert, wie folgt bereitstellen. Zunächst muss das Datenbankschema für REST-Dienste mit ORDS aktiviert werden.

```
begin
  ords.enable_schema (
    P_ENABLED        => true,
    P_SCHEMA         => 'SCOTT',
    P_AUTO_REST_AUTH => false
  );
end;
```

Listing 1: Das Datenbankschema SCOTT für REST-Services mit ORDS freischalten

```
begin
  ords.create_service(
    p_module_name => 'd3bubble',
    p_base_path =>   '/d3bubble/',
    p_pattern =>     'getdata',
    p_method =>      'GET',
    p_source_type => ords.source_type_query,
    p_source =>      q'#select ename as name, job as packageName, '||
                        sal as value from emp#'

  );
end;
```

Listing 2: Einrichtung des REST-Service als Datenquelle für den D3 Chart

Ruft man den REST-Service nun mit dem Browser auf, so werden bereits JSON-Daten ausgeliefert.

Abbildung 7: Vom neuen REST-Service gelieferte JSON-Daten

Entspricht die JSON-Struktur der Daten ganz genau dem Beispielcode, so braucht man nur noch im Javascript-Code die URL im Aufruf von **D3.json** zu ändern und schon generiert der Javascript-Code den Bubblechart auf Basis der eigenen Tabellendaten. Hat das JSON eine andere Struktur, so muss man ggfs. noch weitere Code-Änderungen vornehmen.

```
:
d3.json(
  "http://{server}:{port}/ords/scott/d3bubble/getdata",
   function(error, root) {
:
```

Listing 3: Anpassung der JSON-Datenquelle im Javascript-Code

Als Ergebnis erhält man den Bubblechart anhand der Daten in der Tabelle EMP. Analog dazu lässt sich auch das andere Beispiel (Bar Chart) anwenden.

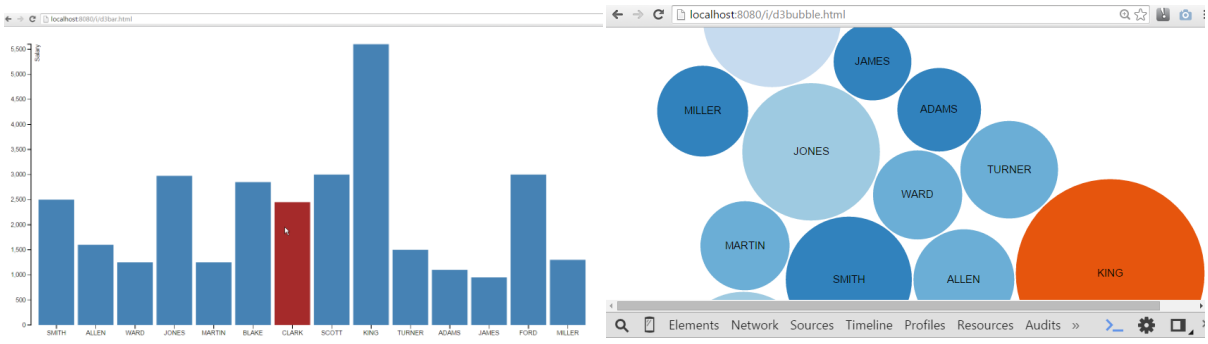

Abbildung 8: D3-Charts auf Daten der Tabelle EMP

Daten in einer APEX-Anwendung per Ajax-Callback liefern

In einer APEX-Anwendung funktioniert die Einbindung eines D3.js-Diagramms ähnlich. Ein eigener REST-Service wird hier aber nicht benötigt, da APEX die Daten selbst im JSON-Format bereitstellen kann. Alles, was es hierzu braucht, ist ein sog. *AJAX-Callback*. Der PL/SQL-Code, der beim Aufrufen desselben ausgeführt wird, generiert die JSON-Daten für das D3.js-Diagramm.

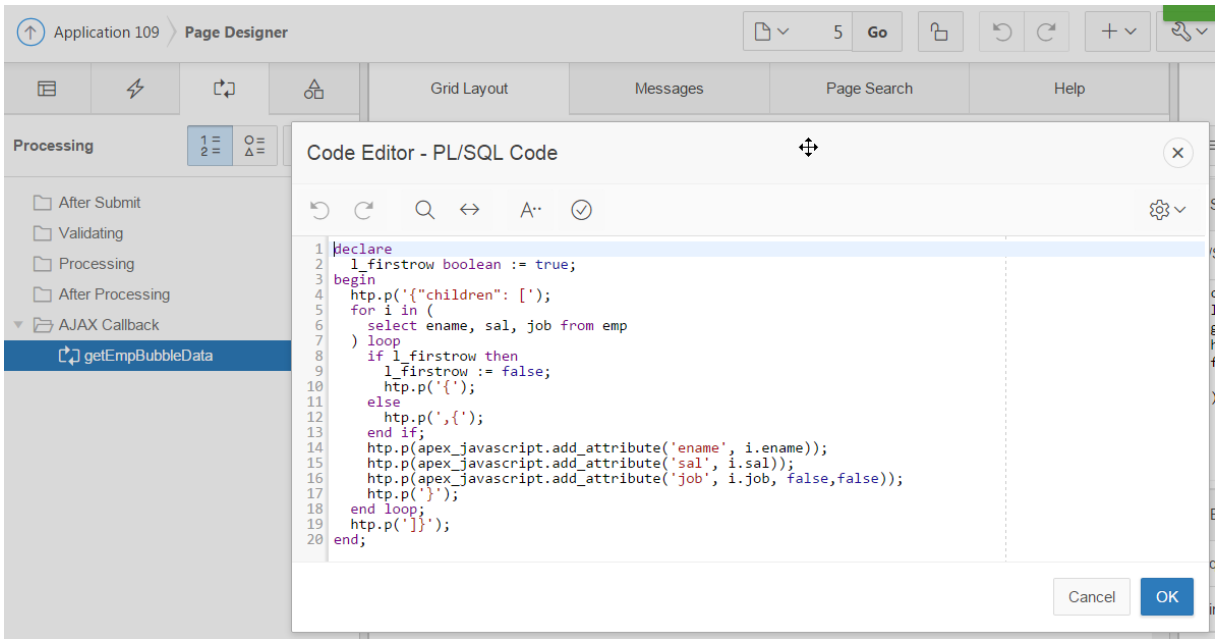

Abbildung 9: AJAX-Callback in Application Express: JSON-Daten für D3.js erzeugen

Auch der Javascript-Code wird in die APEX-Applikation integriert – hierfür gibt es in APEX spezielle Seitenattribute (*Javascript Function and Global Variable Declaration*). **D3.json** wird allerdings durch einen APEX-Spezifischen Javascript-Aufruf ersetzt.

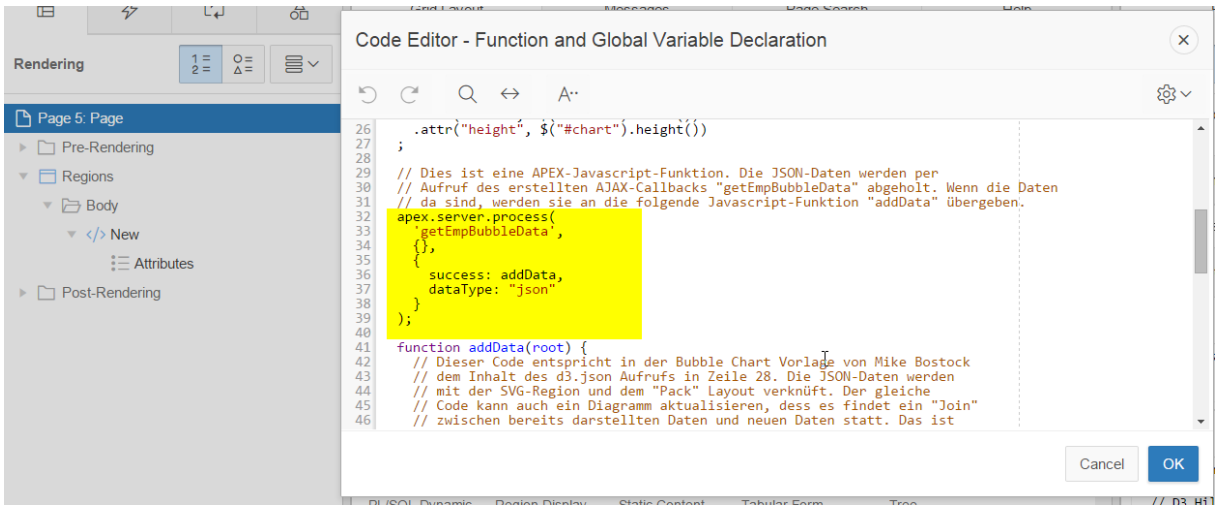

Abbildung 10: Aufruf des AJAX-Callbacks mit einer Javascript-Funktion von Application Express

Die Verarbeitung der Daten und der Aufbau des Diagramms funktionieren wie gehabt; es gibt hier keine Unterschiede zur einfachen Beispielseite von Charles Bostock. Im Ergebnis erscheint dass D3.js-Diagramm als Teil der APEX-Anwendung.

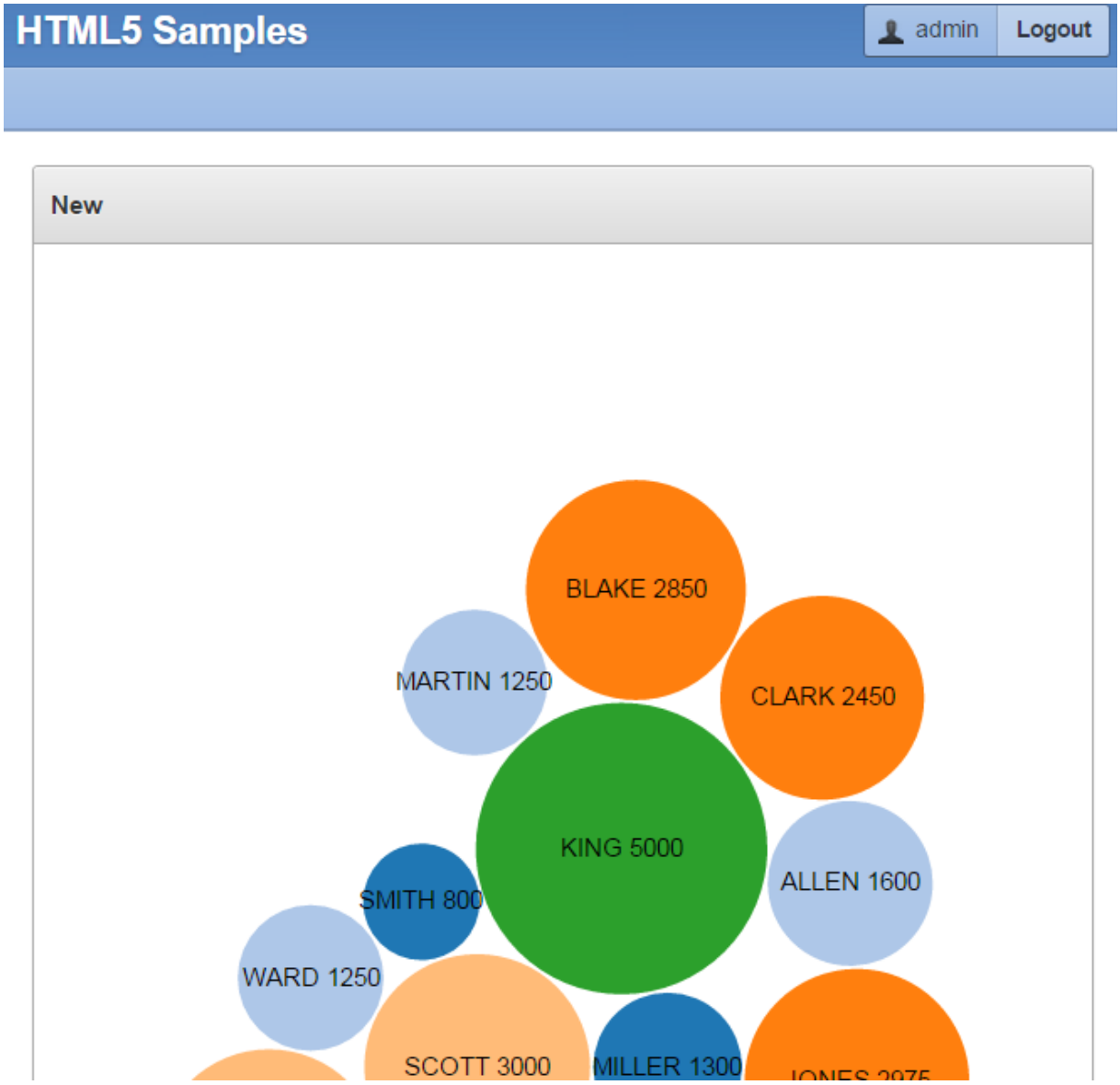

Abbildung 11: D3.js-Diagramm in einer APEX-Anwendung

Zusammenfassung und Ausblick

Man sieht deutlich, dass das Erstellen eines Diagramms mit D3.js einen gewissen Programmieraufwand mit sich bringt. Im Gegenzug erhält man als Entwickler eine nahezu unerhörte Freiheit beim Erstellen der Diagramme. Da man die SVG-Elemente direkt setzt und alle Attribute steuern kann, ist auch das Verändern kleinerer Details wie Liniendicken oder des Abstands der Blasen voneinander überhaupt kein Problem mehr. Im Zweifel lässt sich alles mit Javascript-Code und den Funktionen des D3-Frameworks umsetzen.

Auf die gleiche Art und Weise können Sie nun übrigens auch andere D3-Diagrammtypen umsetzen. Nimmt man die Beispiele des D3js Autors Mike Bostock als Vorlage, ist die Vorgehensweise immer gleich.

- Es braucht eine Datenquelle im JSON-Format. In APEX kann man mit einem AJAX-Callback arbeiten, ansonsten leisten REST-Services gute Dienste. Das JSON-Format muss zum verwendeten Diagrammtyp passen, ansonsten muss man Anpassungen am Javascript-Code vornehmen.
- Es braucht ein HTML-Gerüst, in dem das Diagramm dargestellt wird – typischerweise ist das ein sog. *DIV-Container*, dessen Größe mit Style-Attributen definiert ist und der per "id"-Attribut für Javascript referenzierbar ist. In APEX erzeugt man dazu eine Region vom Typ *Static Content*.
- Schließlich muss der Javascript Code hinterlegt werden, welcher die Daten vom AJAX-Callback holt und das D3.js-Diagramm erzeugt. Hier nimmt man ebenfalls am besten die Vorlagen von Mike Bostock als Ausgangspunkt und passt sie Schritt für Schritt an die eigene Umgebung an.

Wenn das fertige D3.js Diagramm auf dem Bildschirm erscheint, hat man den ersten Schritt getan. Von da aus lässt sich das Diagramm allerdings noch sehr stark verbessern.

- **Responsiveness** – wenn das Browserfenster vergrößert oder verkleinert wird, soll sich das Diagramm automatisch anpassen.
- **Legende** – links, rechts oder unter dem Diagramm könnte eine Legende erscheinen.
- **Tooltips** – werden Diagrammelemente mit der Maus "überfahren", so erscheinen zusätzliche Informationen.
- **Barrierefreiheit** – anstelle der Maus kann man mit den Cursortasten durch das Diagramm navigieren.

All diese Dinge sind nicht aus dem Stand Teil des Diagramms – D3.js ist ja ein Framework und keine fertige Diagrammengine – man muss es also programmieren. Von der einfachen Darstellung des Diagramms bis hin zum komplexen Chart, der viele Features anbietet, ist es also ein weites Spektrum.

APEX-Anwender finden einige fertige D3.js Diagramme als Plug-ins in der Packaged Application **Sample Charts**. Diese Plug-ins bringen die oben genannten Funktionen bereits mit und können in eigenen APEX-Anwendungen verwendet werden.

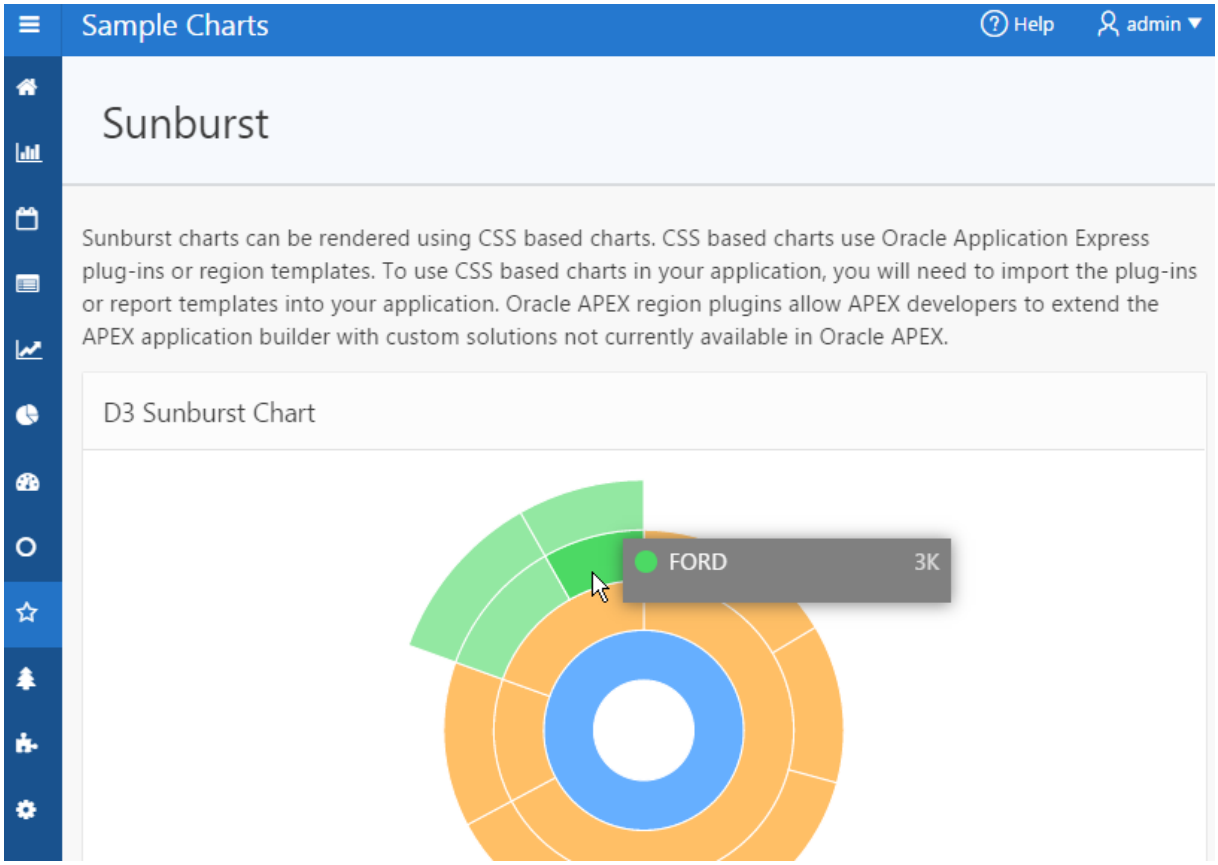

Abbildung 12: D3.js Sunburst Chart als fertiges APEX-Plugin in der Sample Charts Beispielanwendung

In Oracle Application Express ist D3.js als Diagrammframework also zumindest teilweise angekommen – das APEX Entwicklerteam hat selbst einige Diagrammtypen als Plug-in bereitgestellt; weitere gibt es in der internationalen Entwicklercommunity. Für Entwickler, die mit modernen Web-Technologien wie SVG, HTML5, Javascript und REST-Services vertraut ist, ist die Nutzung von D3.js und das Erstellen individueller Diagramme ein Leichtes.

Weitere Informationen

- D3.js Webseite im Internet
  http://www.d3js.org
- APEX Community-Posting: "D3js" in APEX-Anwendungen integrieren
  https://apex.oracle.com/pls/apex/GERMAN_COMMUNITIES.SHOW_TIPP?P_ID=3481
- Blog "JSON, REST und die Oracle-Datenbank": REST-Services und D3.js
  http://json-rest-oracledb.blogspot.co.uk/2015/07/ords-und-javascript-im-einsatz-d3js.html
- APEX Packaged Application "Sample Charts"

Kontakt:

Carsten Czarski

Carsten.Czarski@oracle.com

http://twitter.com/cczarski

http://sql-plsql-de.blogspot.com