Oracle JET Charts in APEX: Verwendung der JavaScript API zur (nachträglichen) Anpassung von Diagrammen

Sven Jagic
merlin.zwo InfoDesign GmbH & Co. KG
Karlsruhe

Schlüsselworte
Oracle, Entwicklung, APEX, JET, JavaScript, jQuery

Einleitung
In APEX stehen seit der Version 5.1 auf Oracle JET basierende Diagramme zur Datenvisualisierung zur Verfügung. Ein solches Diagramm kann einfach „aus der Box heraus“ erstellt werden, d.h. ohne händisches Einbinden der von JET benötigten JavaScript-Bibliotheken. Anpassungen lassen sich einfach über die APEX-Oberfläche vornehmen. Speziellere Änderungen müssen dann schon mit JavaScript umgesetzt werden und erhöhen der Komplexitätsgrad erheblich. Nicht nur wegen der dafür benötigten Sprachkenntnisse, sondern auch weil die Dokumentation der Schnittstelle nicht leicht zu interpretieren ist. In diesem Vortrag sollen anhand eines konkreten, von einem Projekt abgeleiteten, Beispiels einige dieser Lücken geschlossen werden.

Was ist Oracle JET?
Oracle JET (JavaScript Extension Toolkit) basiert auf HTML5, CSS3 und JavaScript. Genauer gesagt ist es eine Sammlung von OpenSource JavaScript-Bibliotheken, zusammen mit einigen Bibliotheken von Oracle, die die Gestaltung ansprechender Benutzeroberflächen ermöglichen und vereinfachen soll. Um eine JET-Komponente einzubinden, werden verschiedene JavaScript-Bibliotheken benötigt. Dazu zählen insbesondere jQuery, Knockout.js, RequireJS und Hammer.js. Diese Arbeit wird seit Version 5.1 von APEX erledigt, denn hier kommt JET sozusagen als Bordmittel im Bereich der Datenvisualisierung in Form von Diagrammen zum Einsatz.
Dieser Vortrag beinhaltet Elemente eines Erfahrungsberichts, da er sich von einem konkreten Projekt ableitet und stützt sich dabei auf APEX 5.1.1. Diese Version verwendet wiederum JET 2.0.2. Das Interessante hieran ist, dass die aktuelle JET-Version die Nummer 4 trägt, die aktuelle Doku aber nicht zu Version 2.0.2 passt.

Erstellen eines Balkendiagramms in APEX mit Bordmitteln
Obwohl ein händisches Einbinden von JET-Komponenten unter Verwendung der oben genannten Bibliotheken durchaus möglich ist, begnügen wir uns mit der Erstellung eines Balkendiagramms mit einfachen Bordmitteln. Nicht zuletzt, weil diese Variante deutlich schneller und einfacher realisierbar ist und der zeitliche Schwerpunkt auf der Anpassung des Diagramms liegen soll.

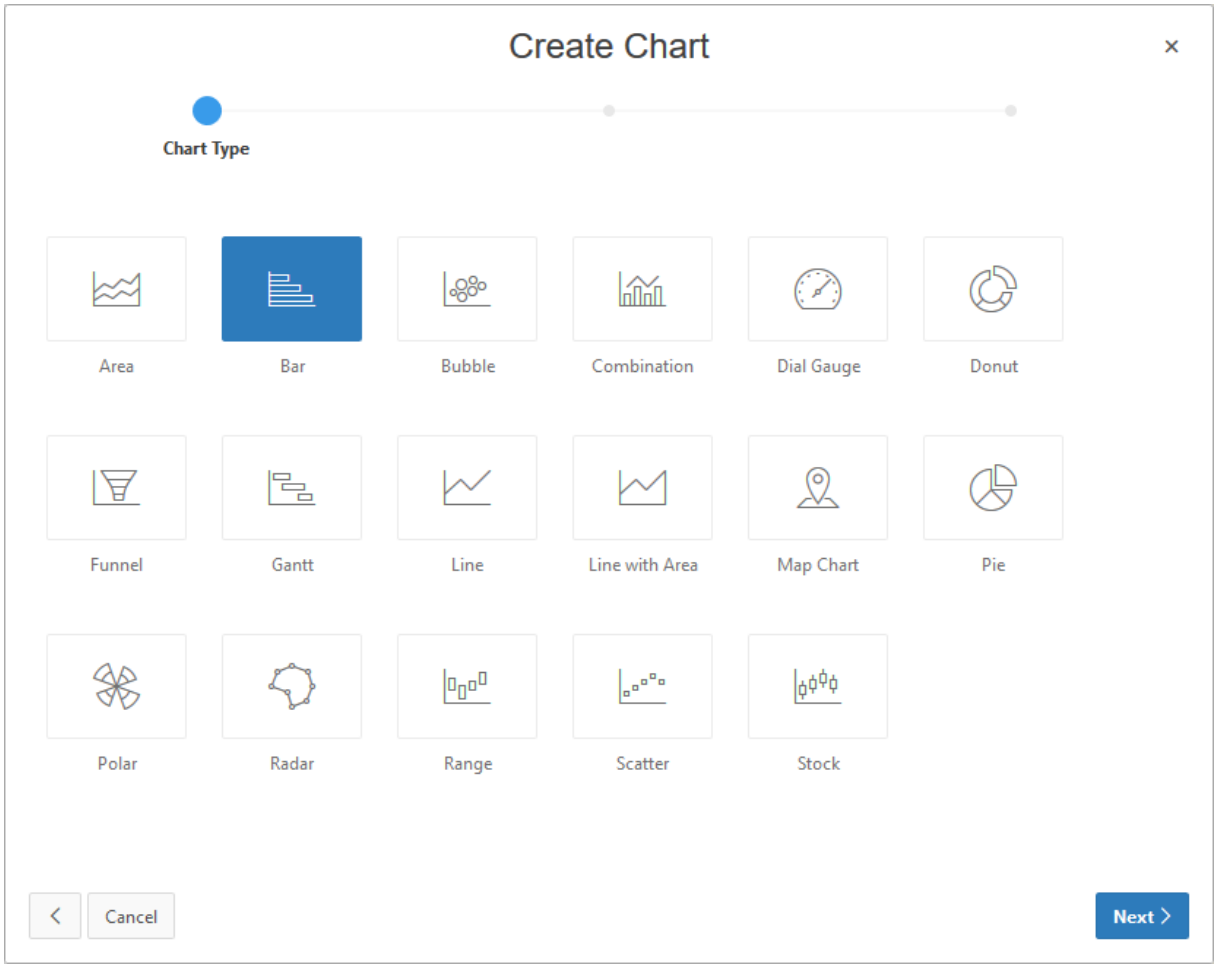

Abb. 1: Erstellen eines Balkendiagramms mit Bordmitteln

Danach wird man sofort mit den Begriffen der „Serie“ bzw. „Reihe“ sowie der „Gruppe“ konfrontiert. Am Beispiel von Verkaufszahlen sähe das folgendermaßen aus: Möchte man ein Balkendiagramm erstellen, welches die monatlichen Verkaufszahlen dreier verschiedener Produkte visualisiert, so wären „Produkt A“, „Produkt B“ und „Produkt C“ drei Serien. Die Bezeichnungen „Januar 2017“, „Feburar 2017“, etc., wären Gruppen. Die eigentlichen Verkaufszahlen pro Produkt pro Monat würden sich in der Balkengröße niederschlagen.

Der nächste Schritt ist die Formulierung einer SQL-Abfrage. Es gibt prinzipiell drei Möglichkeiten diese zu hinterlegen:

1. Eine Abfrage für jede Serie in der jeweiligen Serie.
2. Eine Abfrage in der Region. Die Source der Serie muss dann auf „Region Source“ eingestellt sein.
3. Eine Abfrage in einer Serie oder der Region, durch welche Serien dynamisch erzeugt werden.

Nur mit der dritten Option hat man die Möglichkeit, die Anzahl der Serien dynamisch zu gestalten. Eine solche Abfrage könnte sich wie folgt gestalten:

```
SELECT monat            AS gruppe,
       produkt          AS serie,
       sum(absatzmenge) AS werte
  FROM verkaufszahlen
 GROUP BY monat, produkt;
```

Mit folgendem Ergebnis:

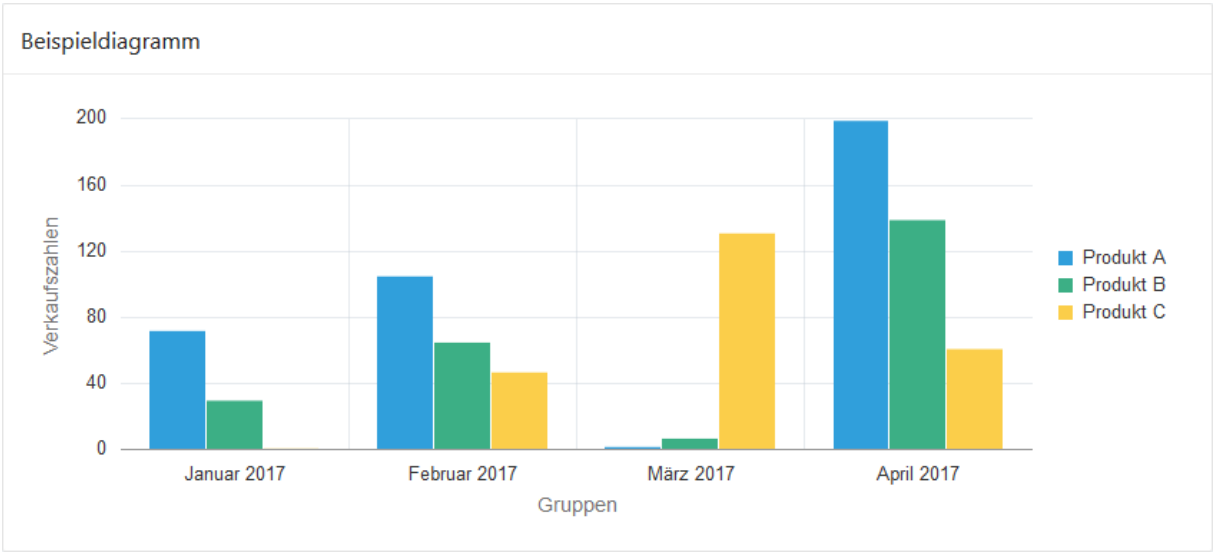

Abb. 2: Beispieldiagramm

Das HTML des Diagramms und warum CSS nicht funktioniert

Nun, da das Diagramm erstellt wurde, möchte man vielleicht einen Blick in das daraus resultierende HTML werfen. Hier ein kurzer Ausschnitt:

```
<line y1="92.4" x2="1479" y2="92.4" shape-rendering="crispEdges" stroke="rgba(196,206,215,0.4)" pointer-events="none"></line>
<line y1="46.2" x2="1479" y2="46.2" shape-rendering="crispEdges" stroke="rgba(196,206,215,0.4)" pointer-events="none"></line>
<line x2="1479" shape-rendering="crispEdges" stroke="rgba(196,206,215,0.4)" pointer-events="none"></line>
<g clip-path="url(#chart1000172535644$cp11)">
  <polygon points="46 147.8 138 147.8 138 231 46 231" cursor="pointer" fill="rgb(48, 159, 219)"></polygon>
▶ <g cursor="pointer" role="img" aria-label="Serie: Produkt B; Gruppe: Januar 2017; Wert: 30,00. Gewählt">…</g>
  <polygon points="231 229.8 323 229.8 323 231 231 231" cursor="pointer" fill="rgb(251, 206, 74)"></polygon>
▶ <g cursor="pointer" role="img" aria-label="Serie: Produkt A; Gruppe: Februar 2017; Wert: 105,0. Auswahl aufgehoben">…</g>
▼ <g id="_dvtActiveElement186231023" cursor="pointer" role="img" aria-label="Serie: Produkt B; Gruppe: Februar 2017; Wert: 65,00
    <polygon points="508 155.9 601 155.9 601 231 508 231" fill="#5a5a5a"></polygon>
    <polygon points="510 157.9 599 157.9 599 229 510 229" fill="#ffffff" pointer-events="visiblePainted"></polygon>
    <polygon points="511 158.9 598 158.9 598 228 511 228" fill="rgb(60, 175, 133)" pointer-events="visiblePainted"></polygon>
  </g>
▶ <g cursor="pointer" role="img" aria-label="Serie: Produkt C; Gruppe: Februar 2017; Wert: 47,00. Auswahl aufgehoben">…</g>
  <polygon points="786 228.7 877 228.7 877 231 786 231" cursor="pointer" fill="rgb(48, 159, 219)"></polygon>
```

Abb. 3: Durch ein JET-Diagramm erzeugtes HTML

Wer sich ein aussagekräftiges Stück Code erhofft hat wird an dieser Stelle leider enttäuscht. Das liegt zum einen an der unerwarteten Struktur, zum anderen an ungewöhnlichen Tags wie *<line>*, *<g>* oder *<polygon>*. Auch einige HTML-Attribute wirken exotisch. Wider Erwarten wird die Balkenfarbe nämlich nicht über Standard-CSS definiert, sondern über das Attribut *fill*, welches einen RGB-Code beinhaltet.

Ein eleganter Eingriff mit CSS, beispielsweise um die Balkenfarbe zu ändern, ist somit nicht möglich – man könnte die Farben höchstens überschreiben. Zudem bräuchte man einen Einstiegspunkt, also

einen Selektor, welcher zunächst nicht vorhanden ist. Diese Lösung ist allerdings nicht nur wenig elegant, sondern schlicht unmöglich, denn beim Hovern mit dem Mauszeiger über die Balken wird das zugrundeliegende HTML einfach ausgetauscht. Somit werden auch alle selbst zugewiesenen Klassen und Inline-Styles verworfen. Ein Eingriff mit jQuery (direkt, ohne die designierte API zu verwenden) schlägt aus denselben Gründen fehl und erübrigt sich somit ebenfalls.

Anpassen des Diagramms bei dessen Erstellung
Das Anpassen eines Diagramms über die Grenzen der APEX-Entwicklungsoberfläche hinaus gestaltet sich also als nicht-triviale Aufgabe. Genauer gesagt: Vermutlich führt kein Weg an der Verwendung der API vorbei, wenn man andere als die durch APEX angebotenen Eigenschaften ändern möchte. Das ist soweit kein Beinbruch. Aber wie funktioniert diese API?

Klickt man im APEX Page Designer auf das Balkendiagramm bzw. die „Attributes“ hat man unter „Advanced“ die Möglichkeit JavaScript zu hinterlegen. Klickt man nun auf den Reiter „Help“ erscheint folgender Beispielcode:

```
function( options ){
    // Setup a callback function which gets called when data is retrieved,
it allows to manipulate the series
    options.dataFilter = function( data ) {
        data.series[ 0 ].color = "#00FF00";
        return data;
    };
    // Set chart initialization options
    options.type = "line";
    return options;
}
```

Es gibt also eine Möglichkeit das Diagramm zu manipulieren. Und es gibt mindestens solche Eigenschaften wie *data.series[0].color* und *options.type*. Das Auffinden einer umfassenden Liste aller Eigenschaften bedarf einer etwas intensiveren Recherche. Mit etwas Geduld gelangt man zur offiziellen Dokumentation von Oracle – nicht das sogenannte „Cookbook“, sondern die „API documentation“. Mit etwas Glück stimmt auch die Version der Doku mit der Version des in APEX integrierten JET überein. Hier gibt es eine Klasse *ojChart* mit einigen Eigenschaften und Methoden, darunter auch *series.color* und *type* – diese haben wir schon im Beispielcode entdeckt.
Mit dem bisher erlangten Wissen lässt sich nun schon einiges machen, wobei ein wenig „Trial and Error“ immer mit von der Partie sein wird.

Verändern des Diagramms nach dessen Erstellung
Neben den Initialisierungsparametern gibt es vielleicht auch einige Dinge, die erst im Nachhinein geändert werden sollen. Das könnte die Ausrichtung des Diagramms sein (horizontal oder vertikal), die Farbe einer Serie, die Farbe des aktiven Balkens und vieles mehr. Das Hauptproblem besteht dabei darin das JavaScript-Objekt der in der Doku beschriebenen Klasse richtig anzusprechen: Dies funktioniert nämlich nur über geschickte Spielerei mit der ID der APEX-Region. Hierfür kann sich ein Blick in die von APEX mitgelieferten „Packaged Apps“ als sehr lohnenswert erweisen.
Nachdem auch diese Hürde genommen wurde, lassen sich recht bequem grundlegende Änderungen vornehmen. Für die Manipulation tieferliegender Elemente, wie zum Beispiel einzelner Balken, bedarf es allerdings einer bestimmten Methode des *ojChart*-Objekts – *option* – deren Verwendungsweise jedoch keineswegs üblich ist. Insbesondere wenn es um verschachtelte Elemente, wie einzelne „items“ einer „series“, geht. Diese werden nämlich über einen einzelnen String referenziert – mit Punkt als Trennzeichen...

Inhalt der Präsentation

Die Präsentation führt Sie zunächst kurz durch die Schritte, die notwendig sind, um in APEX ein einfaches Balkendiagramm zu erstellen. Dazu gehört das Formulieren von kurzen, dem Diagramm zugrundeliegenden SQL-Abfragen. Einerseits, um die Begriffe „Serie“ und „Gruppe“ zu erklären und andererseits, um eine gemeinsame terminologische Basis zu schaffen, sodass die gezeigten Beispiele weniger abstrakt und für Sie greifbarer sind.

Danach schauen wir uns das für das Diagramm generierte HTML an, wobei erklärt wird, warum Eingriffe mit Standard-CSS und JavaScript bzw. jQuery (d.h. ohne JET-API) nicht möglich sind, um ein Diagramm auf stabile Art und Weise anzupassen.

Der Kern der Präsentation besteht aus Beispielen, wie ein JET-Diagramm – exemplarisch ein Balkendiagramm – mit JavaScript und jQuery bei dessen Erstellung sowie im Nachhinein angepasst werden kann. Hier wird auch ganz konkret auf die Verwendung der API eingegangen. Im Zuge dessen werden auch die Hindernisse die zum Schließen der „Dokumentationslücke“ erforderlich waren angesprochen.

Abschließen wird die Präsentation mit einem Ausblick zur Schnelllebigkeit von Web-Frameworks am Beispiel von JET: Seit der Durchführung des Projekts, von welchem sich dieser Vortrag ableitet, ist nur ein knappes Jahr vergangen. Oracle JET hat in dieser Zeit zwei Versionssprünge hinter sich gebracht, nämlich von 2.0.2 auf 4. In dieser Zeit hat sich *ojChart* laut Doku von einer „Klasse“ über eine „Komponente“ zu einem „Element“ weiterentwickelt. Konkret bedeutet das: In der Dokumentation von Version 4 wird die zur nachträglichen Anpassung eines JET 2.0.2 Diagramms benötigte Methode nicht mehr erwähnt, da dies mittlerweile auf eine andere Art und Weise geschieht. In APEX 5.1.1 ist jedoch noch immer JET 2.0.2 implementiert.

Kontaktadresse:
Sven Jagic
merlin.zwo InfoDesign GmbH & Co. KG
Elsa-Brändström-Straße 14
D-76228 Karlsruhe

Telefon:	+49 (0) 721-132096 45
Fax:	+49 (0) 721-132096 99
E-Mail	sven.jagic@merlin-zwo.de
Internet:	www.merlin-zwo.de