Dynamic jQuery Actions in APEX - oder was?

Ein Vortrag von Markus Dötsch - MuniQSoft GmbH

Bemerkung

Der eigentliche Vortrag erfolgt mit dem Einsatz einer APEX Applikation mit der Integration der JavaScript Bibliothek impress.js. Es handelt sich dabei um eine dynamische Darstellung mit ausführlicher Live Demo direkt in APEX, die sich nur schlecht auf Papier bringen lässt.

Daher gibt es eine kurze Zusammenfassung der Theorie und einem ausführlich beschriebenen Beispiel der Technologie am Ende.

Die Demo selbst gibt es nach der Präsentation zum selber durchklicken unter http://apex.oracle.com.

Einleitung

Der Einsatz von jQuery in APEX - oder brauch ich das noch - ich hab ja Dynamic Actions?

Durch die Etablierung von Dynamic Actions in APEX wurden viele manuelle Programmierungen unnötig. Durch die Integration in die Entwicklungsumgebung können dynamische Aktionen deklarativ erfolgen. Der Entwickler kann damit wie gewohnt, über einen Wizard gesteuert, die Aktionen zusammenstellen und muss sich nicht direkt mit der Syntax einer Programmiersprache, wie JavaScript oder einer Bibliothek, wie jQuery auseinander setzen.

Aber kommen wir in der Praxis immer mit dieser Funktionalität aus oder gibt es Dinge, die doch wieder ausprogrammiert werden müssen?

In diesem Vortrag wird versucht zu klären, ob und wenn ja, wann der Einsatz von jQuery im APEX Umfeld Sinn macht? Es wird vermittelt was ein Selektor ist, wie man diesen einsetzt und wie Animationen und Effekte verwendet werden können. Wie können Seiteninhalte dynamisch verändert werden und wie funktioniert der Einsatz von AJAX mit jQuery.

Auch auf die Zusammenhänge zwischen dem Einsatz von JavaScript-Funktionen und jQuery wird eingegangen. Zudem werden Tipps gegeben, wie mit den verschiedenen Browser-Tools die Seiten "gedebuggt" werden können.

Das Ganze ist eingebettet in kleine Funktionen und Techniken, die in bestehende oder neue APEX Applikationen eingebaut werden können. In den Unterlagen sind diese mit "Schritt für Schritt"-Anleitungen zum selber nachbauen enthalten.

Brauche ich noch jQuery oder JavaScript? Ich hab ja Dynamic Actions?

Hier gilt dieselbe Regel, wie mit allen anderen deklarativen Funktionen in APEX. Wenn man mit den vorhandenen Möglichkeiten auskommt, die die Entwicklungsumgebung hergibt, braucht man sich

mit Programmiersprachen nicht auseinander zu setzen. Sobald man aber an die Grenzen stößt, wäre es nicht schlecht, wenn man auch einen alternativen "Entwicklungs-Weg" gehen kann.

Die Entwickler von APEX versuchen zwar viele Möglichkeiten der Verwendung von dynamischen Funktionen in die Entwicklungsumgebung einzubetten, aber es lässt sich nicht alles in "Wizards" abbilden.

Nicht umsonst kann man auch mit Dynamic Actions beliebigen JavaScript-Code ausführen.

Wenn man sich den Quelltext einer APEX Seite ansieht, auf der eine Dynamic Action eingebunden ist, dann erkennt man, das Dynamic Actions jQuery-Code erzeugen. :)

Warum brauche ich überhaupt Dynamic Actions oder jQuery?

Für eine einfache Frage gibt es eine einfache Antwort. Um benutzerfreundliche Webanwendungen bereit zu stellen. Die meisten Anwender unserer APEX Anwendungen sind durch die Möglichkeiten auf den Webseiten im Internet bereits mit den Vorzügen von dynamischen Funktionen vertraut.

Damit unsere Anwendungen von den Benutzern auch angenommen werden, müssen wir Ihnen soweit entgegen kommen, wie die Entwicklungsumgebung, die verfügbare Zeit und unsere technischen Kompetenzen es zulassen.

Und eines ist sicher - an APEX liegt es sicher nicht. Alles was man an einer Webseite selbst erlebt hat, kann man auch in seinen eigenen APEX-Anwendungen umsetzen.

Wie heißt es im APEX Umfeld immer so schön: Akzeptanz durch Firlefanz!

Was sind die Vorteile bei der Verwendung von Dynamic Actions?

Das hervorstechendste Merkmal der Dynamic Actions sind die direkte Integration in die Entwicklungsumgebung. Wir wissen sofort wo wir sie finden können. Bei der Erstellung einer Dynamic Action können wir aus vorgefertigten Funktionen wählen und müssen nur die Dinge angeben, die für die gewünschte Funktion notwendig sind.

Durch die deklarative Einbindung erfolgt automatisch eine Art von Dokumentation der Funktionalität. Mit ein bisschen Fleiß ist über die Verwendung der APEX Views sogar eine aufgebereitete Dokumentation möglich. Diese kann als Report oder für den Ausdruck optimiert zur Verfügung gestellt werden.

Was sind die Vorteile beim Einsatz von jQuery?

jQuery ist eine auf JavaScript basierende Bibliothek, die dem Entwickler viele Arbeitsschritte erspart und damit die Entwicklung von dynamischen Funktionen erleichtert. jQuery ist keine eigene Sprache sondern basiert auf JavaScript. Normale Sprachentypische Funktionen wie Bedingungen und Schleifen sind nicht in jQuery enthalten. Diese müssen mit "normalem" JavaScript umgesetzt werden.

Für den erfahrenen Webentwickler bedeutet das, dass er wie gewohnt seinen JavaScript Code erzeugen kann, sowie ihn auf althergebrachte Art und Weise in eigenen Dateien kapseln kann, wie normalen JavaScript Code.

Wie analysiere ich JavaScript/jQuery/Dynamic Actions

Eine Entwicklung einer Web-Applikation ohne den Einsatz geeigneter Tools ist kaum mehr vorstellbar. Ein Vorreiter für die Entwicklung im Browser war das Addon Firebug im Firefox. Die aktuellen Browser liefern mittlerweile ebenfalls geeignete Tools mit. In der Live-Demo gibt es einen Einblick in die Entwicklungsumgebungen von Chrome und Firefox.

Analyse des aus der APEX Anwendung entstehenden HTML-Codes. Verstehen des DOM-Baumes einer Webseite.

➔ Live-Demo mit Beispielen

Was ist eigentlich ein jQuery Selektor

Mit einem jQuery Selektor können wir einen bestimmten Bereich des HTML-Quellcodes definieren um ihn als "Anker" für weitere Funktionen zu verwenden.

Dabei handelt es sich um die Grundlage aller Funktionen, die basierend auf den "Anker" ihre Tätigkeit durchführen. Auch Dynamic Actions benötigen diesen "Anker" und bieten im Gegenzug auch die Angabe eine jQuery Selektors in ihren Funktionen an.

➔ Live-Demo mit Beispielen

Wie sieht es mit der Sicherheit von dynamischen Aktionen aus?

Spätestens wenn der Einsatz von Ajax mit Dynamic Actions oder jQuery ins Spiel kommt, müssen wir uns als Entwickler über die Sicherheit unserer Applikation Gedanken machen. Die dynamischen Aktionen laufen direkt im Browser ab und lassen sich deshalb von uns nicht kontrollieren. Sollten wir also eine Änderung von Datenbankinhalten über dynamische Aktionen erlauben, müssen wir immer auf der PL/SQL Seite überprüfen ob diese Änderungen überhaupt erlaubt sind.

➔ Live-Demo der Datenmanipulation

Für was benötige ich Ajax noch einmal?

Durch den Einsatz von Ajax (Asynchrones JavaScript und XML) können wir mit den Dynamic Actions und jQuery auf die Datenbank direkt zugreifen ohne die Seite neu zu laden. Diese Funktionalität erhöht die Benutzerfreundlichkeit einer Anwendung oft sehr.

Hier können die Dynamic Actions voll ihre Stärken ausspielen, da durch die deklarative Auswahl von Objekten direkt die Quellen und Ziele einer solchen Aktion ausgewählt werden können.

In der Live-Demo wird ein Standard-Report mit einer Icon-Spalte verwendet, über das per Klick ein "Schalter" pro Zeile in der Datenbank betätigt werden kann. Dabei wird das Zusammenspiel von AJAX und PL/SQL deutlich gemacht.

➔ Live-Demo der Erstellung der Funktionalität

Beispiel für eine Koexistenz beider Welten - Dynamic Actions & jQuery

Darstellung eines Classic Reports in dem die Zeilen anklickbar sind und damit die Übergabe einer Datenbank-ID erfolgt um zum Datensatz gehörende Informationen aus der Datenbank zu laden und in einem separaten Bereich darzustellen.

➔ Live-Demo der Erstellung der Funktionalität

Beispiel für den Einsatz von Dynamic Actions und einem Plugin

Einsatz von modalem Bearbeitungs-Dialog und anschließendem Refresh des darunter liegenden Reports.

➔ Live-Demo der Erstellung der Funktionalität

Weitere Beispiele

- Darstellung von mehreren Regionen auf einer APEX Seite und Verwendung des Region Display Selectors und ausblenden des "Show All" Reiters.
- Editieren von Einträgen innerhalb eines Reports. Ähnlich der Funktion in den Websheets können wir den Anwendern erlauben in einem Report auf einen Wert zu klicken, damit dieser zu einer Eingabebox wird und der Wert über Ajax in der Datenbank gespeichert werden kann.
- Die Fehlermeldungen einer Formular-Validierung können in einem Overlay "über" der Navigation oder anderen statischen Bereichen dargestellt werden.
- Tooltip - Information, die entweder an einer festen oder der angeklickten Position dargestellt wird, wenn man mit der Maus über einen definierten Bereich geht
- Markierung einer Zeile in einem Report ohne das darunterliegende SQL zu manipulieren
- Auf- und zuklappbare Sidebar, die sich mit der "Haupt-Region" den Bildschirmplatz teilt.
- Veränderung von Regionsbereichen innerhalb einer APEX Seite über die Maus.

Ausführliches Beispiel für das Verändern von Daten aus einem Report ohne Neuladen der Seite

In vielen Datenmodellen werden Schalter für die Aktivierung oder Gültigkeit von Datensätzen verwendet. Hier kann zum Beispiel eine Spalte mit 0/1 oder Y/N belegt werden um damit zu steuern, welchen Status ein Datensatz haben soll.

Im Normalfall hinterlegen wir so eine Spalte in einem Formular mit einer Selectlist in der "Ja" oder "Nein" oder ähnliches ausgewählt werden kann. Wie wäre es aber, wenn wir den Anwendern die Möglichkeit geben würden diesen Schalter im Report zu ändern. Er könnte mehrere Datensätze hintereinander verändern ohne die Seite neu laden zu müssen.

Gehen wir einmal von folgender Tabelle aus.

```
CREATE TABLE  "EMP" (
        "EMPNO" NUMBER(4,0) NOT NULL,
        "ENAME" VARCHAR2(10),
        "JOB" VARCHAR2(9),
        "MGR" NUMBER(4,0),
        "HIREDATE" DATE,
        "SAL" NUMBER(7,2),
        "COMM" NUMBER(7,2),
        "DEPTNO" NUMBER(2,0),
        "ACTIVE" VARCHAR2(1),
        PRIMARY KEY ("EMPNO")
)
/
```

Es handelt sich dabei um die bekannte EMP-Tabelle, ergänzt um die Spalte "ACTIVE", die mit Y oder N befüllt werden soll. Idealerweise verwenden Sie eine bereits bestehende EMP-Tabelle und erweitern sie mit der neuen Spalte.

Um einen Default-Wert für den Status der Mitarbeiter zu setzen feuern wir noch ein Update-Statement um alle Mitarbeiter auf aktiv zu setzen.

```
update EMP set ACTIVE = 'Y';
```

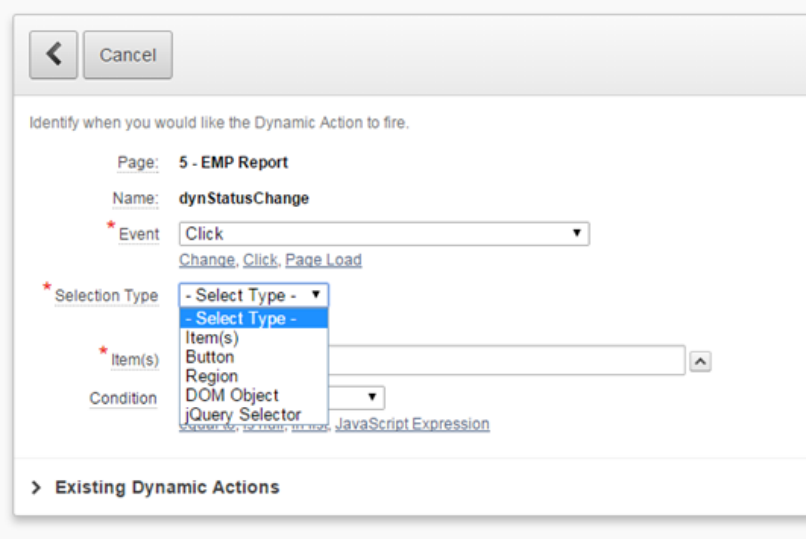
In APEX erstellen wir einen klassischen Report über alle Spalten der EMP Tabelle.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	Active
7369	SMITH	CLERK	7902	17-DEC-80	800		20	Y
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30	Y
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30	Y
7566	JONES	MANAGER	7839	02-APR-81	2975		20	Y
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30	Y
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30	Y
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10	Y
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20	Y
7839	KING	PRESIDENT		17-NOV-81	5000		10	Y
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30	Y
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20	Y
7900	JAMES	CLERK	7698	03-DEC-81	950		30	Y
7902	FORD	ANALYST	7566	03-DEC-81	3000		20	Y
7934	MILLER	CLERK	7782	23-JAN-82	1300		10	Y

Nun ist die Idee auf das Y zu klicken und ein N daraus zu machen, sowie natürlich anders herum.

Um auf den Mausklick eines Anwenders reagieren zu können müssen wir einen Eventhandler aktivieren, der auf das anzuklickende Objekt horcht und dann bei Klick eine Aktion ausführt.

Im APEX-Umfeld haben wir die Möglichkeiten dies mit den Dynamic Actions umzusetzen. Problem dabei ist, dass wir die Spalte eines Reports nicht als "Selection Type", also als Ziel eine "Klicks" auswählen können.

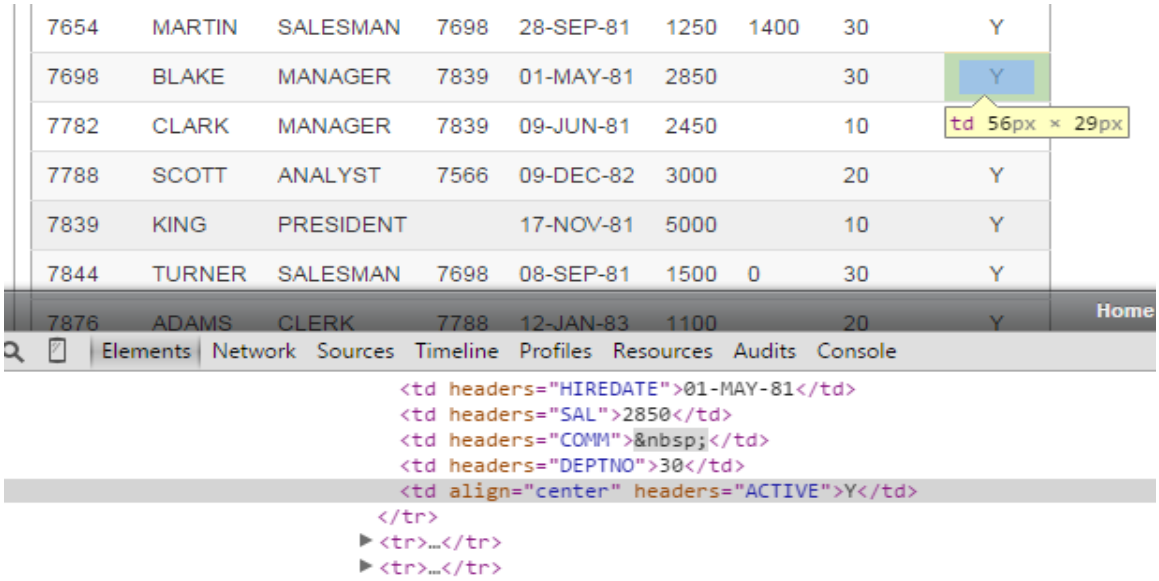


Also müssen wir uns mit der HTML Struktur der Tabelle auseinandersetzen. Diese kann je nach gewählten Theme unterschiedlich sein. In diesem Beispiel handelt es sich um das Theme 25 in der APEX Version 4.2.6.

Eine Analyse des von APEX erzeugten HTML Codes erfolgt am besten mit den Entwicklertools der aktuellen Browsergenerationen. Die Browser Firefox und Chrome bringen ausgereifte und mit vielen Funktionen versehene Oberflächen mit. Um sie zu aktivieren genügt es mit der rechten Maustaste auf das zu untersuchende Objekt auf der Webseite zu klicken und "Element untersuchen" auszuwählen.

Es öffnet sich in der Regel am unteren Bildschirmrand ein zweigeteilter Bereich, in dem links der HTML Code und rechts das wirkende CSS angezeigt wird.

Bewegt man die Maus über den Quellcode wird der Bereich der Webseite markiert, der durch diesen Code definiert wird.



So erhalten wir die Information, das die Ausgabe von Y oder N in einem "td" erfolgt, das mit dem Parameter "headers" und dem Wert "ACTIVE" versehen wird. Es handelt sich dabei um die von uns gewählte Überschrift der Spalte. Das gilt für alle Zeilen unseres Reports.

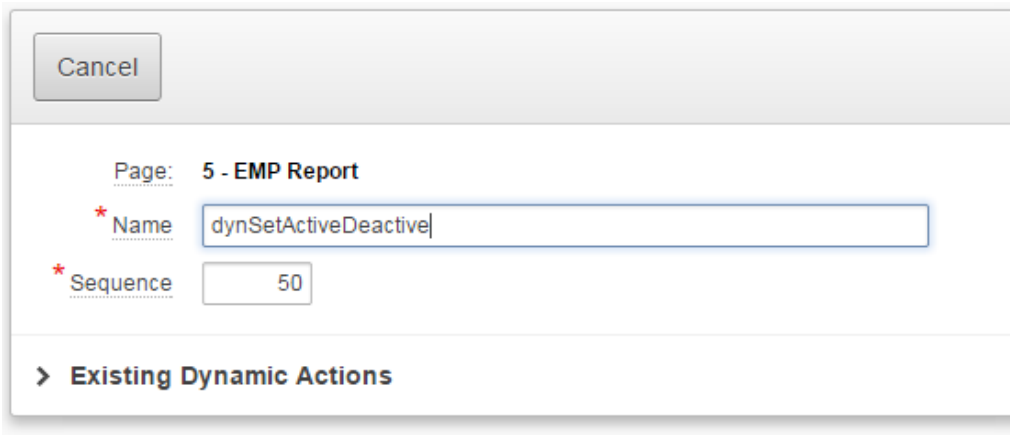
Mit dieser Information ausgerüstet können wir einen jQuery Selektor definieren, den wir in der Dynamic Action verwenden können.

Die JavaScript Bibliothek jQuery bietet sich an, da es sich zu einem "Quasi-Standard" im Internet etabliert hat und auch von APEX selbst verwendet wird. Mit einer leicht zu erlernenden Notation können wir einen Bereich innerhalb des HTML Codes definieren, der als Anker oder Startpunkt für unsere gewünschten Aktionen dienen kann.

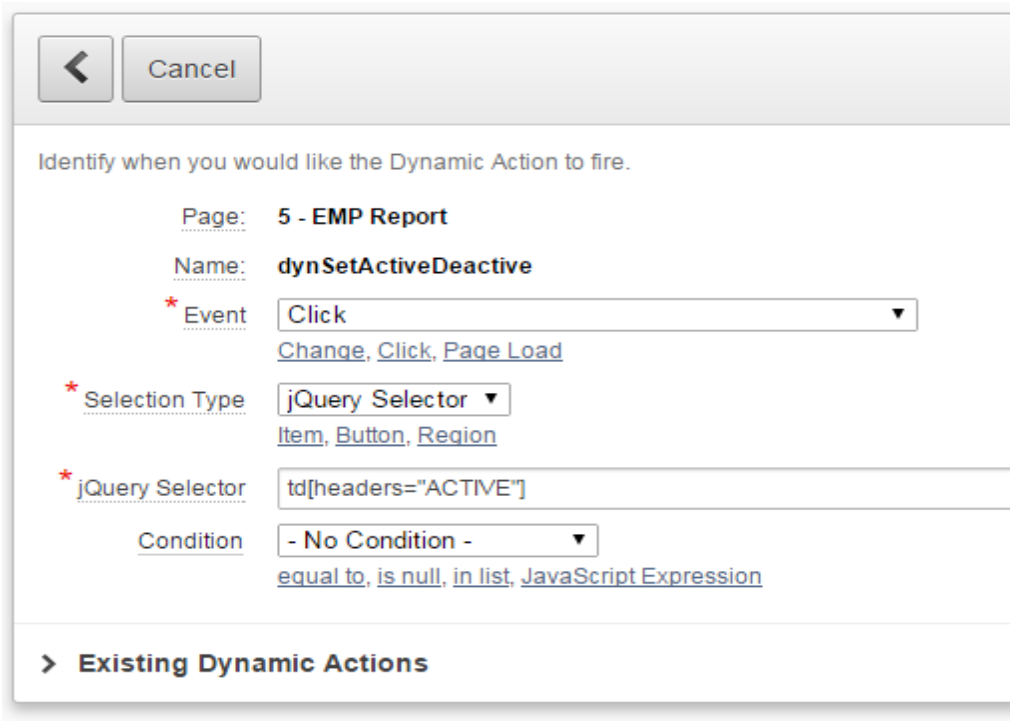
In diesem konkreten Fall möchten wir gerne auf einen "td" klicken können, der den "Header" "ACTIVE" besitzt. Also definieren wir den Selektor für jQuery folgendermaßen:

```
td[headers="ACTIVE"]
```

Erstellen wir also nun eine neue Dynamic Action und verpassen ihr einen sprechenden Namen.

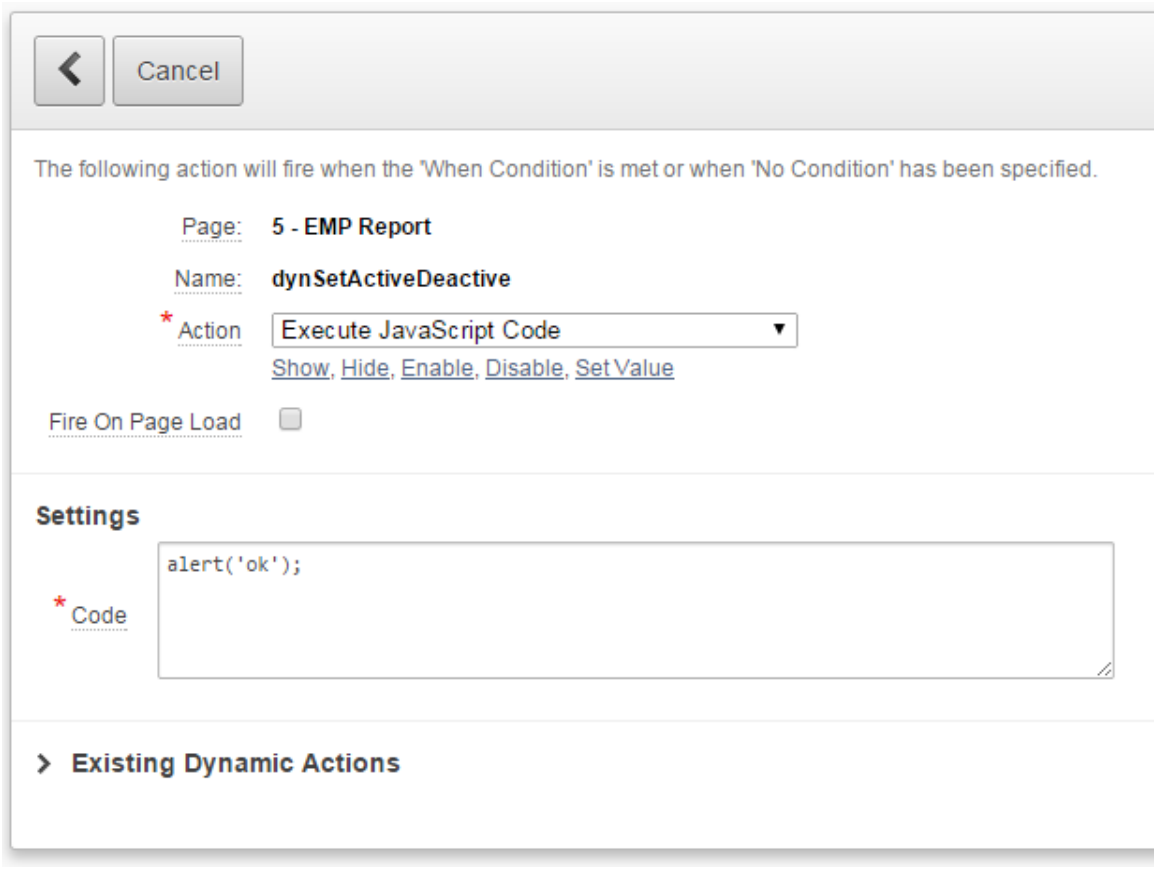


Sobald der Event "Click" auftritt auf dem gewählten jQuery Selector soll ...



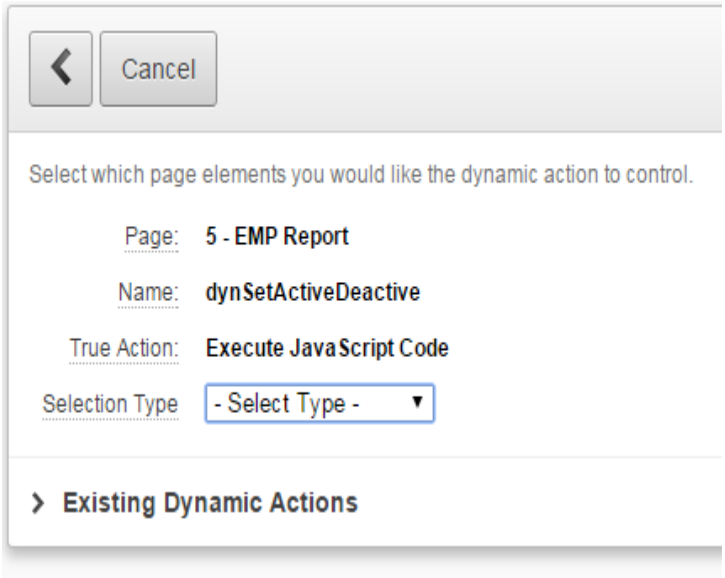
... die Action "JavaScript" ausgeführt werden. Um unsere Eventhandler zu testen geben wir zuerst nur einen Alert aus der bei Klick angezeigt werden soll.

```
alert('ok');
```



Den Haken bei "Fire On Page Load" deaktivieren wir, da wir nicht möchten, dass die Aktion beim Laden der Seite gefeuert wird.

Da wir alle Aktionen mit JavaScript ausführen werden, benötigen wir im nächsten Schritt kein Element, das benutzt werden soll und wählen hier nichts aus und schließen die Erstellung der Dynamischen Aktion ab.

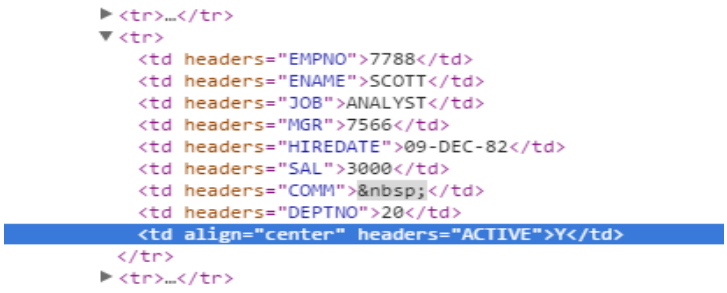


Wir haben den Test bestanden, wenn wir nun im Report auf das "Y" klicken können und eine Alert-Box erscheint. Ein Klick auf einen anderen Bereich der Webseite sollte dagegen den Event nicht auslösen.

Jetzt haben wir zwar den Klick auf "Y" mit dem Eventhandler abgefangen, wir wissen aber noch nicht auf welches "Y", sprich welche Zeile geklickt wurde. Irgendwann müssen wir ja in der Lage sein ein Update-Statement an die Datenbank zu senden, um den angeklickten Datensatz zu verändern.

In unserem Fall müssen wir die "EMPNO" der angeklickten Zeile ermitteln. Wir können das mit jQuery erreichen, da die Empno in diesem Fall mit im Report angezeigt wird.

Wenn wir uns im Browser noch einmal den Quellcode anzeigen lassen, entdecken wir den von uns gesuchten Wert in dem ersten "td" einer jeden Zeile.



Die Idee ist nun, ausgehend von dem angeklickten "td" den ersten "td" der Zeile ("tr") zu ermitteln. Zurück in der von uns angelegten Dynamic Action ersetzen wir unsere Debug-Ausgabe mit folgendem jQuery-Selector.

```
alert(
  $(this.triggeringElement).parent().children().html()
);
```

Bei dem Dollarzeichen ($) handelt es sich um die Kurzform der JavaScript-Funktion, mit der wir jQuery aufrufen können. Dieser Funktion geben wir das von APEX befüllte Object "this.triggeringElement" als Selector mit. Dann ermitteln wir mit der Funktion "parent" das Elternelement unseres angeklickten "td"s. Es handelt sich dabei um das übergeordnete "tr". Von dort springen mit der Funktion "children" wieder zum ersten Kind-Element zurück. Von diesem geben wir mit der Funktion "html" den HTML-Inhalt aus.

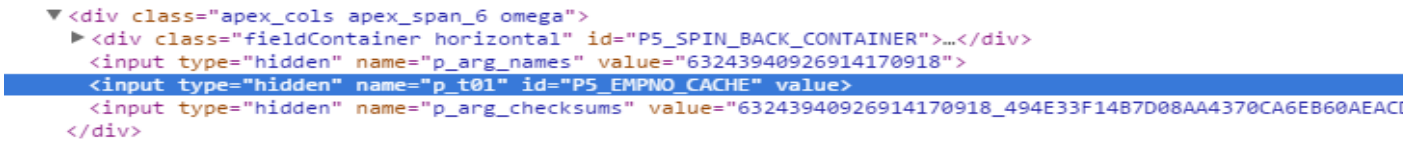
Wer sich ein bisschen mit jQuery beschäftigt hat, wird sicherlich wissen, dass es noch einundrelfzig weitere Wege gegeben hätte sich mit jQuery in der HTML-Struktur zu bewegen. So hätten wir mit Kombinationen der Funktionen "first", "next" oder "prev" auf dasselbe Ergebnis kommen können. Auch der Einsatz des bereits verwendeten aber leicht abgewandelten "td[headers='EMPNO']" wäre möglich gewesen. Dabei ist aber zu beachten, dass alle Bewegungen in der Struktur immer ausgehend vom angeklickten Element starten müssen.

Nun haben wir also erreicht, dass bei Klick in der Alert-Box die Empno der angeklickten Zeile ausgegeben wird. Wir benötigen diese Information aber in einem von APEX verwendbaren Element um es an die Datenbank senden zu können.

Dazu bietet sich ein Hidden-Element an, das wir mit JavaScript befüllen und dann mit PL/SQL auslesen und unseren Update formulieren.

Erster Schritt dazu ist das Anlegen eine Page Items im Format Hidden. Da wir in diesem Element den Wert der Empno zwischenspeichern wollen und ich mich mit meinem Report auf der Seite 5 befinde, nenne ich das Hidden Element "P5_EMPNO_CACHE".

Nach einer kurzen Analyse des HTML Outputs meiner Seite erkenne ich, dass das von mir erzeugte Element von APEX mit dem Parameter ID versehen wurde, mit dem von mir vergebenen Namen.



Damit haben wir alle Informationen um dem Hidden-Item den Wert mit JavaScript zu übergeben. Wir ersetzen also in der Dynamic Action den JavaScript Code mit folgendem Inhalt:

```
$('#P5_EMPNO_CACHE').val(
  $(this.triggeringElement).parent().children().html()
);
```

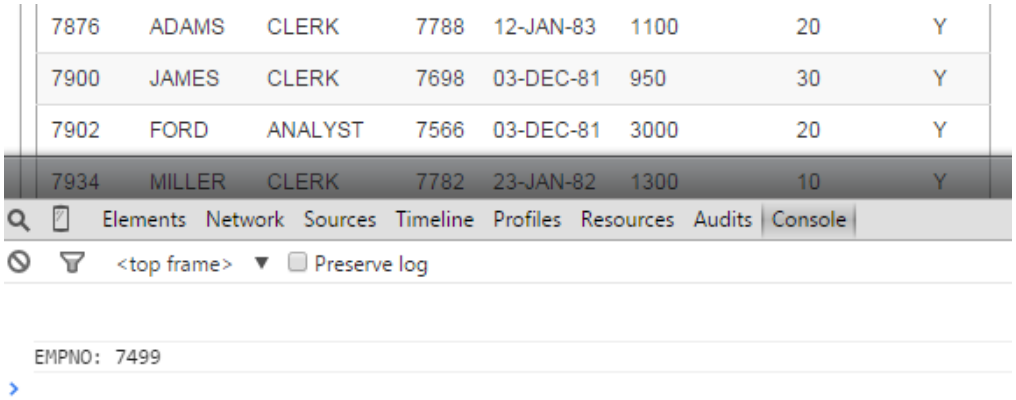
Hier kommt wieder ein jQuery Selektor zum Einsatz. Das Dollarzeichen ist die jQuery-Funktion selbst. Als Parameter übergeben wir in Hochkommas die Information, dass es sich um eine ID handelt - mit der Raute (#) und den Namen der ID "P5_EMPNO_CACHE". Dann setzen wir mit der Funktion "val" die "Value" des schon vorher ermittelten Wertes.

Um das Ganze zu debuggen geben wir noch zusätzlich den folgenden Code in den JavaScript Bereich:

```
console.log(
  "EMPNO: " + $('#P5_EMPNO_CACHE').val()
);
```

Mit dem Befehl "console.log" können wir Ausgaben in die JavaScript Konsole der Entwicklungsumgebung machen. Das ist in der Regel angenehmer als die Ausgabe über den Alert Befehl.

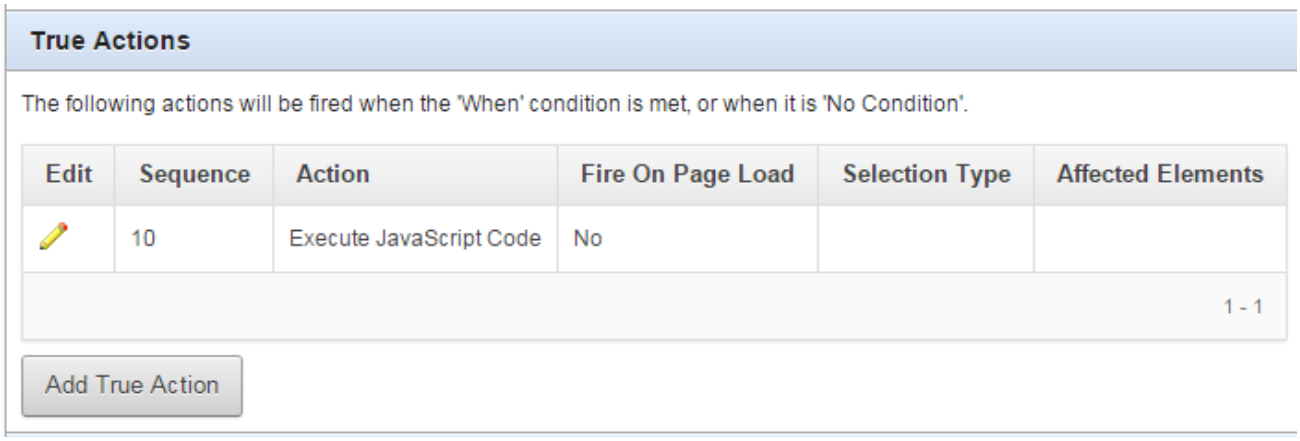
Aufrufen können wir die Console über die rechte Maustaste und "Element untersuchen". Dann klicken wir im Menü der Entwicklungsumgebung auf den Punkt "Console".



Dort sehen wir nach einem Klick auf ein "Y" nun die Ausgabe unseres Log-Befehls. Da wir damit sichergestellt haben, dass das Hidden Element die richtige Empno enthält müssen wir jetzt den nächsten Schritt gehen.

Übergabe des Hidden-Wertes zur Datenbank. Natürlich könnten wir einfach einen "Submit" ausführen und der Wert wäre der Datenbank bekannt. Wir möchten aber gerne ohne Neuladen der Webseite auskommen.

Daher erzeugen wir eine weitere "True Action" in unserer bestehenden Dynamic Action.

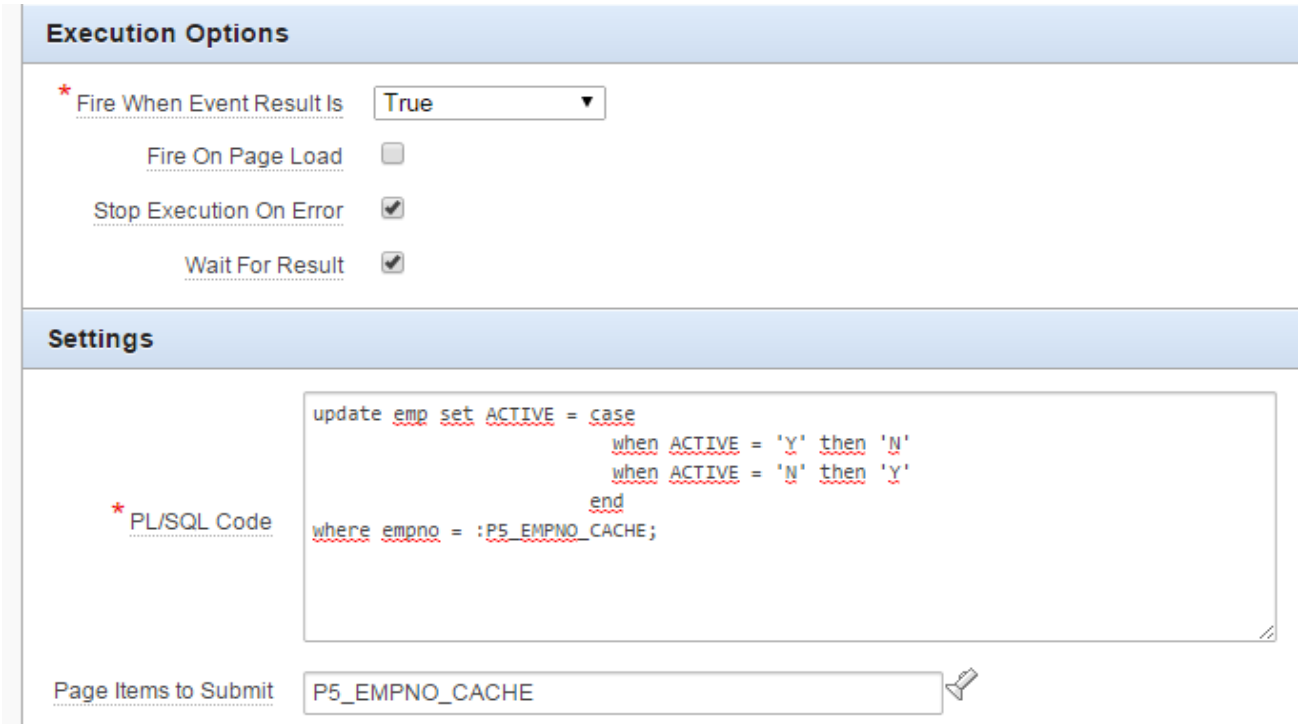


Diesmal möchten wir PL/SQL Code ausführen. Bei den Optionen lassen wir "Fire On Page Load" deaktiviert. Die Optionen "Stop Execution On Error" und "Wait For Result" lassen wir ebenfalls auf dem Default-Wert (angehakt). Vor allem letztere ist wichtig, da erst der PL/SQL Teil abgearbeitet werden soll und dann der nächste Schritt ausgeführt werden soll.

Als PL/SQL Code tragen wir folgendes ein.

```
update emp set ACTIVE = case
                  when ACTIVE = 'Y' then 'N'
                  when ACTIVE = 'N' then 'Y'
                end
where empno = :P5_EMPNO_CACHE;
```

Damit die Datenbank mit diesem PL/SQL Code aber auch auf den Wert in P5_EMPNO_CACHE zugreifen kann muss das Item noch bei "Page Items to Submit" eingetragen werden. Damit stellt die Dynamic Action sicher, dass der Wert auch per AJAX übertragen wird.

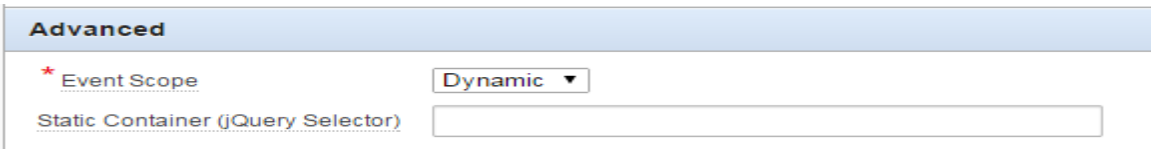


Wenn wir jetzt auf ein Y oder N klicken wird der Wert in der Datenbank geändert, wir sehen aber das Ergebnis erst, wenn wir den Report neu laden. Es fehlt also noch ein Refresh des Reports.

Dazu legen wir einen eine neue "True Action" in unserer Dynamic Action an. Diesmal vom Typ "Refresh". Als "Affected Element" wählen wir "Region" und dann die von uns verwendete Region aus.

Bei einem Test stellen wir fest: Es funktioniert. Aber es funktioniert nur einmal. Dies liegt daran, dass wir einen Eventhandler für das abfangen des Klicks aufbauen, dieser aber beim Refresh des Reports "zerstört" wird.

Daher müssen wir in unserer Dynamic Action unter "Advanced" den "Event Scope" auf "Dynamic". Dann klappt unsere umswitchen des Status auch mehrere Male hintereinander.



Sollte der Refresh überhaupt nicht funktionieren kann es sein, dass im Report unter "Report Attributes" im Bereich "Layout and Pagination" der Schalter "Enable Partial Page Refresh" auf "Nein" steht. Das sollte man dann ändern. :)

Erweiterung um schöne Icons

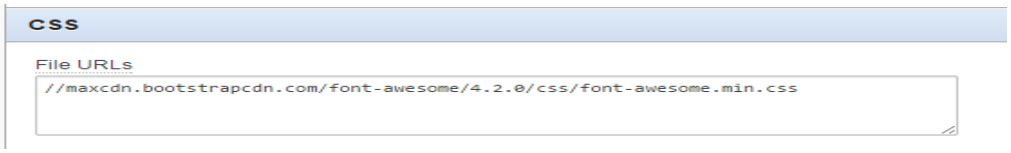
Jetzt funktioniert zwar alles wie gewünscht, aber so richtig deutlich lassen sich die Y und N im Report nicht unterscheiden. Wäre es nicht schöner wir hätten ein rotes X für Nein und einen grünen Haken für Ja.

Hier könnten wir uns mit einem Tipp aus der APEX Community behelfen. Font Awesome. https://apex.oracle.com/pls/apex/GERMAN_COMMUNITIES.SHOW_TIPP?P_ID=3181

Mit der Einbindung eines "Webfonts", der nur aus Icons besteht können wir diese einfach in unsere Anwendung integrieren. Die einfachste Variante diesen Font in unserer Applikation bekannt zu machen ist den folgenden Pfad entweder in unseren Templates oder auf der Seite einzutragen.

//maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-awesome.min.css

In der Page unter "CSS" -> "File URLs".



Auf der Seite http://fontawesome.io/get-started/ werden mehrere Verfahren beschrieben, wie die Datei eingebunden werden kann. Nach der erfolgreichen Integration können wir über die Klassennamen auf die Icons zugreifen.

Unter der Adresse http://fontawesome.io/icons/ finden wir eine gruppierte Auflistung der mitgelieferten Icons und deren Klassennamen.

Um die Icons in unserem Report darstellen zu können müssen wir unser SQL Statement verändern.

```
select
 EMPNO,
 ENAME,
 JOB,
 MGR,
 HIREDATE,
 SAL,
 COMM,
 DEPTNO,
 case ACTIVE
   when 'Y' then '<i class="fa fa-check my_green"></i>'
   when 'N' then '<i class="fa fa-times my_red"></i>'
 end ACTIVE
from EMP
```

Einzige Änderung ist die Ausgabe unserer "Active-Spalte". Dort soll nicht mehr "Y" und "N" ausgegeben werden, sondern die entsprechenden Icons. Die CSS-Klassen "fa", "fa-check" und "fa-times" werden von Font Awesome mitgeliefert. Das einfärben in rot und grün müssen wir aber selber übernehmen. Daher wurden hier die Klassen "my_green" und "my_red" angegeben. Diese müssen wir noch hinterlegen.

Aber zunächst müssen wir die "Active-Spalte" noch umstellen, damit sie den HTML-Code nicht ausgibt sondern interpretiert. Dies machen wir in der Spalte selbst unter "Column Attributes". Dort stellen wir "Display as" um auf "Standard Report Column".

Jetzt werden zwar "Haken" und "X'e" dargestellt aber von rot und grün noch keine Spur. Dies erledigen wir entweder über unsere zentrale CSS-Datei oder wieder direkt auf der Seite. Diesmal unter CSS -> Inline.

```
.my_green {
  color: green;
  cursor: pointer;
}
.my_red {
  color: red;
  cursor: pointer;
}
```

Wir definieren die beiden Klassen für grüne und rot und ändern gleichzeitig noch den Cursor zu einem Pointer, wenn der Benutzer die Maus über die Icons bewegt. So wird der Anwender darauf hingewiesen, dass er hier klicken kann.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	ACTIVE
7839	KING	PRESIDENT		17-NOV-81	5000		10	✔
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30	✔
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10	✔
7566	JONES	MANAGER	7839	02-APR-81	2975		20	✘
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20	✔
7902	FORD	ANALYST	7566	03-DEC-81	3000		20	✔
7369	SMITH	CLERK	7902	17-DEC-80	800		20	✔
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30	✔
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30	✔
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30	✘
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30	✘
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20	✔
7900	JAMES	CLERK	7698	03-DEC-81	950		30	✔
7934	MILLER	CLERK	7782	23-JAN-82	1300		10	✔

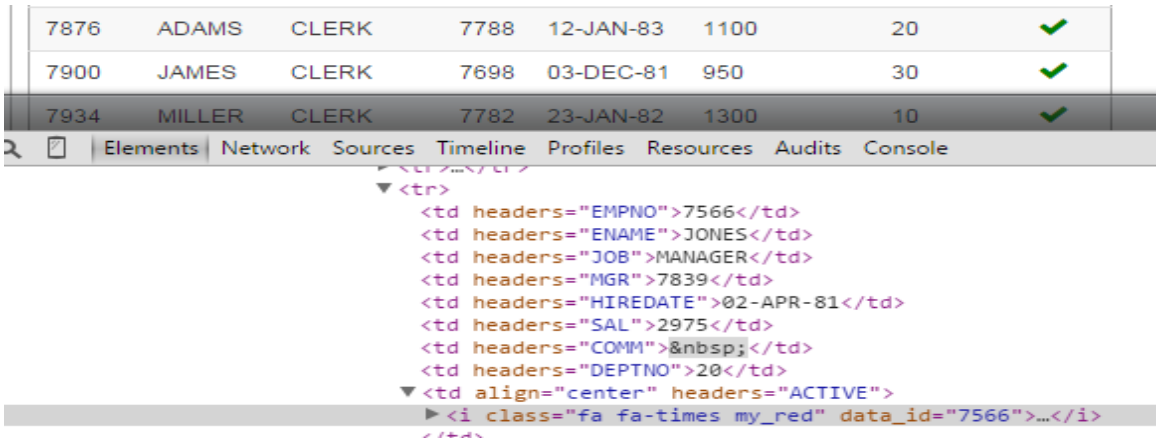
Verwendung von Data-Object in Parameter

Dieses Beispiel basiert darauf, dass der von uns benötigte Primary Key - also unsere EMPNO als Spalte mit im Report ausgegeben wurde. Da dies oft nicht der Fall ist können wir diese Information auch als "Data-Object" als Parameter eines Elements mitgeben und dann darauf zugreifen.

Wir ergänzen also unser bekanntes SQL Statement mit der Ausgabe der Empno in einem Data-Object.

```
select
 EMPNO,
 ENAME,
 JOB,
 MGR,
 HIREDATE,
 SAL,
 COMM,
 DEPTNO,
 case ACTIVE
   when 'Y' then '<i class="fa fa-check my_green" data_id="' || EMPNO || '"></i>'
   when 'N' then '<i class="fa fa-times my_red" data_id="' || EMPNO || '"></i>'
 end ACTIVE
from EMP
```

Im Quellcode haben wir nun die Empno als Parameter in jedem Icon hinterlegt.



Diesen Parameter können wir in unserer Dynamic Action nun abgreifen. Dazu ändern wir in der Dynamic Action den jQuery Selektor auf ...

```
td[headers="ACTIVE"] i
```

Der kleine aber feine Unterschied ist hier, dass wir nicht mehr das "td" anklickbar machen, sondern direkt das "darin" liegende HTML Tag "i".

Den Code unserer ersten True Action, den JavaScript Bereich müssen wir ebenfalls anpassen.

```
$('#P5_EMPNO_CACHE').val(
  $(this.triggeringElement).attr('DATA_ID')
);
```

Statt Väter und Kinder zu suchen greifen wir hier mit der Funktion "attr" direkt auf das Attribut "DATA_ID" des angeklickten Icons zu und ermitteln den Wert.

Ansonsten funktioniert alles wie gehabt. Wir haben nun aber den Vorteil, dass die Position der Spalte Empno keine Rolle mehr spielt und wir sie sogar überhaupt nicht mehr im Report anzeigen müssten.

Fazit

Unserer Phantasie sind sowieso keine Grenzen gesetzt, aber mit dem Einsatz von Dynamic Actions und jQuery lassen sich auch alle Wünsche bei der Gestaltung von benutzerfreundlichen Webseiten umsetzen.