APEX und HTML5: Anwendungen der nächsten Generation

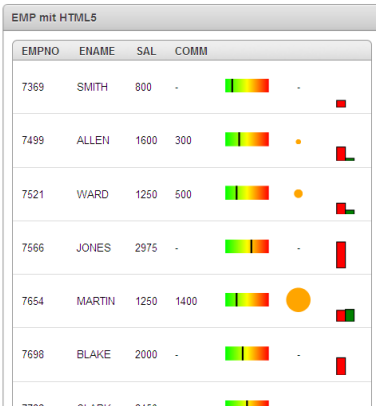
Carsten Czarski
ORACLE Deutschland B.V. & Co KG
München

Einleitung

HTML5 ist bei Web-Entwicklern schon seit einiger Zeit ein heißdiskutiertes Thema. Kein Wunder, denn der neue Standard schafft völlig neue Möglichkeiten für Web-Applikationen: Benutzeroberflächen lassen sich mit den Mitteln von HTML5 ganz anders gestalten.

Eine vollständige Abhandlung über HTML5 würde den Rahmen eines Konferenz-Manuskripts sprengen – daher stehen hier vor allem die Möglichkeiten im Mittelpunkt, die HTML5 dem APEX-Entwickler ohne weitere Eingriffe in die Server-Infrastruktur bietet. Vorgestellt wird also, wie man mit Hilfe von *SVG-Tags* oder dem *Canvas-Objekt* auf die Webseite "zeichnen", wie man hochzuladende Dateien, noch vor dem Upload, auslesen oder wie man auf die *GPS-Koordinaten* des Endgeräts zugreifen kann. Themen, die zusätzliche Server-Komponenten erfordern, wie bspw. HTML5 *WebSockets*, bleiben außen vor.

Neue Benutzeroberflächen: SVG und Canvas



Bekanntlich erzeugt eine Web-Anwendung HTML-Code, der vom Server an den Browser gesendet wird – dieser erzeugt dann die Anwendungsseite. Vor HTML5 erlaubte der HTML-Standard nur das Erzeugen einfacher Layout-Elemente – komplexere Dinge mussten mit (vom Server geladenen) Bildern gemacht werden. HTML5 schafft hier völlig neue Möglichkeiten.

SVG-Tags erlauben es, der Webseite geometrische Primitive wie Linienzüge oder Polygone mit HTML-Tags hinzuzufügen. Alle Objekte sind Teil des DOM, können also mit "id"-Attributen versehen, später mit JavaScript angesprochen und bei Bedarf entfernt oder verändert werden. Der HTML-Code in folgendem Listing zeigt ein Beispiel:

```
<p>
 Der erste Versuch: SVG in einer APEX-Anwendung
</p>
<script type="image/svg+xml">
<svg width="200" height="200"
     style="background-color: #D2B48C;
            display: block; margin-bottom: 5px;">
  <rect x="0" y="0" width="140" height="360" fill="#AAAAAA" />
  <rect x="20" y="130" width="100" height="200" fill="#FF8800" />
  <circle id="c1" r="50" cx="70" cy="70"  fill="#880088" />
</svg>
</script>
```

Im Canvas-Objekt (*Canvas* = "Leinwand") kann dagegen nur mit JavaScript gearbeitet werden. Mit HTML-Tags wird nur das Canvas selbst, und dessen CSS-Angaben, definiert.

```
<body>
  <h1>Canvas Objekt</h1>
  <canvas id="leinwand" width="500" height="400"></canvas>
</body>
```

Auch für spätere DOM-Zugriffe steht nur das Canvas-Objekt selbst zur Verfügung. Mit JavaScript-Kommandos kann nun in das Canvas hinein gezeichnet werden. Die folgenden Kommandos zeichnen zwei farbige Rechtecke in das Canvas.

```
var lCanvas = document.getElementById("leinwand");
var lCtx = lCanvas.getContext("2d");
lCtx.beginPath();
lCtx.fillStyle = "#ff0000";
lCtx.fillRect(20,20,100,100);
lCtx.fillStyle = "#00ff00";
lCtx.fillRect(50,50,100,100);
```

Der Unterschied zwischen den beiden Methoden wird sofort deutlich: Die beiden Rechtecke auf dem Canvas-Objekt wurden mit JavaScript-Aufrufen gezeichnet, *aber nicht* als Objekte dem DOM hinzugefügt. Die JavaScript-Kommando manipulieren *die Pixel* des Canvas direkt. Einzelne Objekte können also – im Gegensatz zu SVG – später nicht mehr angesprochen, entfernt oder verändert werden. Neben dem Zeichnen geometrischer Primitive können auch Bilder ins Canvas geladen oder die Pixelinformationen direkt ausgelesen bzw. verändert werden.

Es ist klar, dass sich SVG- und Canvas-Objekte für viele Anforderungen eignen. Gängig ist sicherlich die Darstellung von Diagrammen, die APEX selbst nicht beherrscht, aber auch das Erzeugen und Darstellen von Bar- oder QR-Codes kann auf diese Weise sehr elegant und direkt im Browser erledigt werden.

Dateizugriff: FileReader

Das *FileReader-Objekt* erlaubt den Zugriff auf vom Benutzer ausgewählte Dateien mit JavaScript. Die Möglichkeit zur Dateiauswahl schafft man, wie schon bislang auf Webseiten, mit dem HTML-Tag <**input**> und dem Attribut **type=file**.

```
<input id="P1_FILEUPLOAD" type="file">
```

In APEX verwendet man ein Formularelement vom Typ **Datei durchsuchen**. Auf der Webseite selbst entsteht dann das bekannte Element zur Auswahl einer Datei. Nur auf Dateien, die auf diese Art und Weise ausgewählt wurden, kann mit dem FileReader zugegriffen werden – das direkte Ansprechen von Dateien auf dem lokalen Dateisystem bleibt (sinnvollerweise) auch mit HTML5 verboten.

Nachdem die Datei ausgewählt wurde, lässt sie sich also per JavaScript auslesen. Als APEX-Entwickler erstellt man nun also am besten eine dynamische Aktion, die ausgelöst wird, wenn das Element **P1_FILEUPLOAD** sich verändert hat (*onChange*). Der folgende JavaScript-Code liest die ersten 100 Bytes der Datei in ein Array und stellt, in einer HTML-Region, die Hexwerte dar.

```
var inf = $x("div-file-info");

var f = this.triggeringElement.files[0];
var r = new FileReader();
var blob =  f.slice(0, 100);    // Bytes von 0 - 100 lesen

var infostr = f.name + "\n" + f.type + "\n" + f.size + "\n\n";

r.onloadend = function(evt) {
 if (evt.target.readyState == FileReader.DONE) {
  infostr = infostr + {getHexString}(evt.target.result);
  inf.innerHTML = "<pre>" + infostr + "</pre>";
 }
}

r.readAsBinaryString(blob);
```

Kombiniert man *Canvas* und *FileReader*, so lässt sich die eingangs erwähnte *Bildvorschau vor dem Upload* sehr einfach realisieren. Interessant (und neu) ist, dass die Bildvorschau *unmittelbar* nach Auswahl der hochzuladenden Datei erscheint. Die folgenden Abbildungen zeigen mögliche APEX-Anwendungsseiten: Einmal werden die Bytes der Datei als Hexwerte dargestellt, das zweite Beispiel lädt das ganze Bild in ein Canvas.

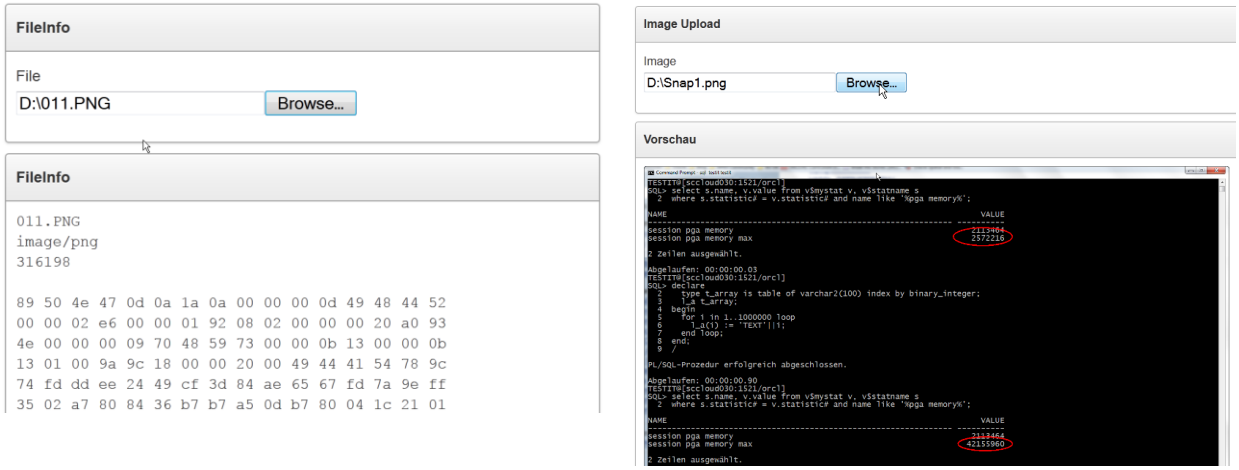


Abbildung 1: Nutzung des HTML5 FileReader

Wo ist der Anwender? HTML5 Geolocation

Vor allem für mobile Endgeräte (aber nicht nur für diese) ist es interessant, die Position des Endanwenders herauszufinden. Auch hierfür sieht HTML5 JavaScript-Kommandos vor: Mit dem Objekt **navigator.geolocation** nimmt man Zugriff auf die Ortungsdienste.

```
navigator.geolocation.getCurrentPosition (
  {handle-location-function},
  {handle-error-function}
);
```

Das Abrufen der Koordinaten findet immer asynchron statt – aus zwei Gründen:

- Beim ersten Mal wird der Nutzer gefragt, ob die Webseite Zugriff auf die Koordinaten bekommen darf – das braucht ein wenig Zeit.
- Das konkrete Abrufen der Position kann (bei einem mobilen Endgerät) auch nochmals ein paar Sekunden in Anspruch nehmen.

Beim Aufrufen von **navigator.geolocation.getCurrentPosition** übergibt man daher *Callback-Funktionen* für den Erfolgs- und den Fehlerfall. In diesen Funktionen hinterlegt man die eigentliche Verarbeitung der Koordinaten, die, wie gesehen, durchaus auch einige Sekunden später stattfinden kann. Durch die asynchrone Arbeitsweise blockiert der Browser bis dahin jedoch nicht. Die einfachste Form der Verarbeitung ist sicherlich die Übernahme in APEX-Elemente vom Typ *Hidden* und die Darstellung auf einer Karte.

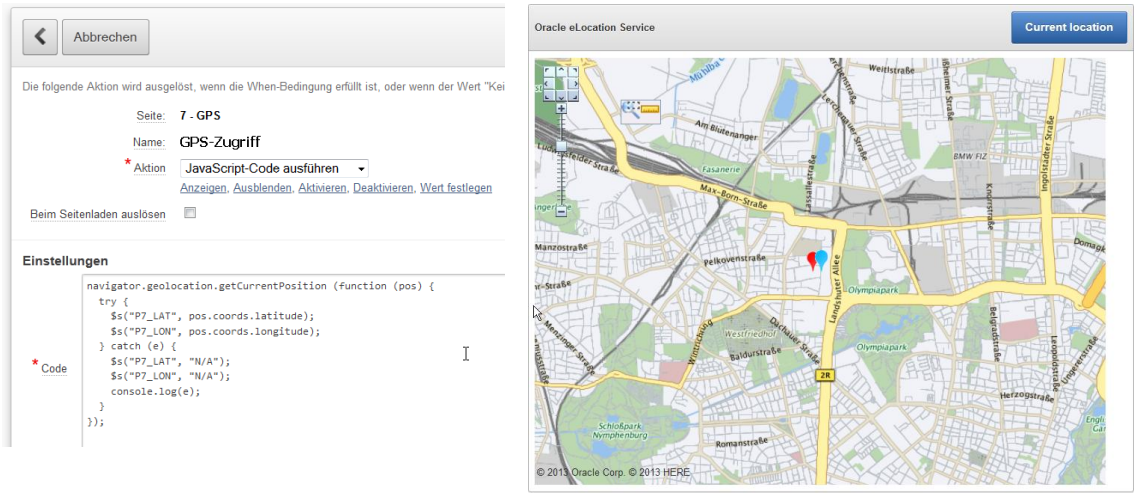


Abbildung 2: Nutzung von HTML5 Geolocation und Darstellung auf einer Karte

Neben den Koordinaten (als Längen- und Breitengrad) kann auch die Höhe, Geschwindigkeit und Blickrichtung zurückgegeben werden – Voraussetzung ist natürlich, dass diese Informationen vom Endgerät ermittelt und bereitgestellt werden.

Fazit und Ausblick

HTML5 ist sicherlich für jeden Entwickler browsergestützter Anwendungen und damit auch für APEX-Entwickler nicht nur wichtig, sondern hochinteressant. Mit HTML5-Techniken können auch komplexere Sachverhalte visualisiert bzw. Nutzeroberflächen wesentlich intuitiver gestaltet werden. Mit der Nutzung von **HTML5 Geolocation** oder des **Canvas Objektes** kann der APEX-Entwickler einfache *Quick Wins* für seine Anwendungen erzielen.

Doch damit ist das Ende der Fahnenstange nicht erreicht. HTML5 bietet dem APEX-Entwickler noch mehr Möglichkeiten an – allerdings erfordert die Nutzung von Websockets, LocalStorage oder die profunde Nutzung von CSS3 etwas mehr Einarbeitung.

Weitere Informationen

[1] HTML5 auf Wikipedia
http://de.wikipedia.org/wiki/HTML5

[2] Details zum Umgang mit HTML5 Geolocation
http://www.w3schools.com/html/html5_geolocation.asp

[3] Deutschprachige APEX und PL/SQL Community
http://tinyurl.com/apexcommunity

[4] Blog des Autors: "SQL und PL/SQL in Oracle"
http://sql-plsql-de.blogspot.com

Kontaktadresse:

Carsten Czarski
ORACLE Deutschland B.V. & Co KG
Riesstr. 25, 80992 München

Telefon:	+49 (0) 89 1430 2116
E-Mail	carsten.czarski@oracle.com
Internet:	www.oracle.de
Blog des Autors	http://sql-plsql-de.blogspot.com
Twitter	@cczarski