Bekannte Probleme

Windows Phone
Im Emulator wird immer die `Connection.UNKNOWN` zurückgegeben.

iOS und Bada
Es wird leider nur unterschieden, ob es eine mobile oder WiFi-Verbindung gibt. Der Grad der mobilen Verbindung (2G, 3G, 4G) wird nicht genannt. Der Wert für eine mobile Datenverbindung ist dann immer CELL_2G.

5.4 Die Benachrichtigung (Notification)

Die Notification bzw. Benachrichtigungsfunktionen von PhoneGap sollten Sie nicht mit den Push-Benachrichtigungen der Smartphone-Hersteller verwechseln. Hier geht es um lokale Benachrichtigungen an den Anwender, die Sie sowohl textuell, per Audio oder per Vibration des Gerätes einstellen können. Generell geht es darum, den Anwender über ein Ereignis zu benachrichtigen, z.B. einen abgeschlossenen Download, oder aber auch nur darum, die Systemdialoge für Auswahlen zu nutzen. PhoneGap stellt hierfür vier Methoden zur Verfügung.

5.4.1 Ein kurzes Beispiel

```
<!DOCTYPE html>
<html>

  <head>
    <title>Notification-Beispiel</title>
    <script type="text/javascript" charset="utf-8"
    src="cordova-2.0.0.js"></script>
    <script type="text/javascript" charset="utf-8">
      // Warten, bis PhoneGap geladen ist
      //
      document.addEventListener("deviceready", onDeviceReady, false);

      // PhoneGap ist geladen.
      //
      function onDeviceReady() {
        navigator.notification.vibrate(3000);
      }
    </script>
  </head>

  <body></body>

</html>
```

Listing 5–16 *Notification-Beispiel*

Dieses kleine Beispiel lässt das Smartphone drei Sekunden lang vibrieren. Die möglichen lokalen Benachrichtigungen werden dabei unter dem Namensraum `notification` zusammengefasst.

Verfügbare Methoden

- `notification.alert`
- `notification.confirm`
- `notification.beep`
- `notification.vibrate`

Es handelt sich dabei um zwei Systemdialoge (`alert`, `confirm`) und eine Benachrichtigung per Piep (`beep`) und per Vibration (`vibrate`) , also sogenannte visuelle, Audio- und taktile Benachrichtigungen. Lassen Sie uns daher die einzelnen Methoden durchgehen und mit dem `notification.alert` anfangen.

5.4.2 notification.alert – native Meldung ausgeben

iOS	Android	Windows Phone	BlackBerry	webOS	Bada
x	x	x	x	x	x

Als erste Benachrichtigungsoption soll eine Alertbox bzw. ein System-Alarm-Dialog dienen. Dabei wird mit dem folgenden Aufruf der Dialog ausgelöst:

```
navigator.notification.alert(message, alertCallback, [title], [buttonName]);
```

Hier werden dem `alert`-Aufruf dabei einige Parameter übergeben. Lassen Sie uns diese kurz durchgehen.

- `Message`
  Die Nachricht, die im Dialog angezeigt werden soll.
- `alertCallback`
  Eine Funktion, die ausgeführt wird, wenn der Dialog geschlossen wird.
- `title`
  Die Überschrift des Alert-Dialogs. Dieser Parameter ist optional. Wenn er nicht definiert wurde, ist *Alert* die Vorbelegung.
- `buttonName`
  Dieser Parameter ist optional. Wenn er nicht definiert wurde, ist OK die Vorbelegung.

Damit kennen Sie nun den `alert`-Aufruf. Sicherlich könnten Sie auf den meisten Browsern auch die Browser-Funktion `alert()` in JavaScript nutzen, aber diese Ausgabe ist nicht so gut anpassbar wie der `alert`-Dialog von PhoneGap.

Windows-Phone

Da unter Windows Phone 7 der sonst oft verwendete Browseraufruf `alert()` nicht funktioniert, kann hierfür leicht `notification.alert` genutzt werden.

Nun folgt ein komplettes Beispiel zur Implementierung der `alert`-Benachrichtigung.

Ein komplettes Beispiel

```
<!DOCTYPE html>
<html>

  <head>
    <title>notification.alert-Beispiel</title>
    <script type="text/javascript" charset="utf-8"
    src="cordova-2.0.0.js"></script>
    <script type="text/javascript" charset="utf-8">
      // Warten, bis PhoneGap geladen ist
      //
      document.addEventListener("deviceready", onDeviceReady, false);

      // PhoneGap ist geladen.
      //
      function onDeviceReady() {

      }

      function alertDismissed() {}

      // Anzeige einer Alert-Notification
      //
      function showAlert() {
        navigator.notification.alert(
          'Sie haben gewonnen!',     // Nachricht
          alertDismissed,            // Callback
          'Spielende',               // Titel
          'Fertig'                   // buttonName
        );
      }
    </script>
  </head>

  <body>
    <p>
      <a href="#" onclick="showAlert(); return false;">
        Alert anzeigen
      </a>
```

```
    </p>
  </body>

</html>
```

Listing 5–17 *alert-Beispiel*

Damit Sie sehen, wie sich der Dialog jeweils an das Smartphone anpasst, habe ich hier das Listing auf verschiedenen Plattformen laufen lassen.

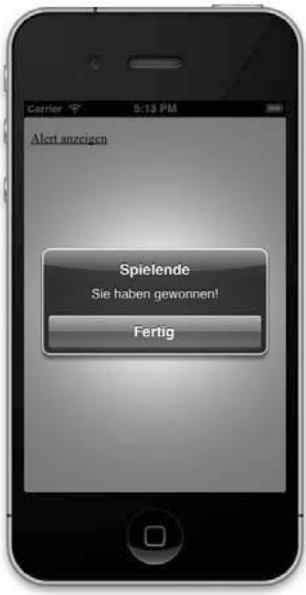

Abb. 5–8 *alert-Beispiel am iPhone*

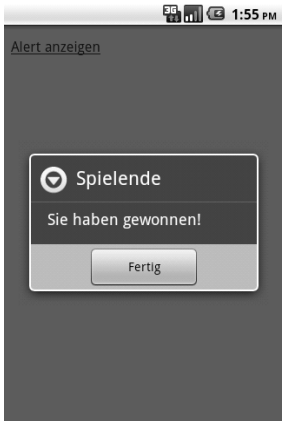

Abb. 5–9 *alert-Beispiel auf Android*

Da aber eine einfache Ausgabe manchmal zu wenig ist, gibt es noch eine zweite Möglichkeit, den Anwender per Dialog zu benachrichtigen: den `notification.confirm`.

Achtung

Windows Phone 7 ignoriert den Parameter `buttonName`, es wird immer der String OK angezeigt.

5.4.3 notification.confirm – nativen Auswahldialog anzeigen

iOS	Android	Windows Phone	BlackBerry	webOS	Bada
x	x	x	x	–	x

Die nächste Möglichkeit der Benachrichtigungen ist der Confirm-Dialog. Anders als bei der `alert`-Benachrichtigung haben Sie mit `confirm` die Möglichkeit, dem Anwender mehrere Auswahloptionen anzubieten.

- `Message`
  Die Nachricht, die im Dialog angezeigt werden soll.
- `confirmCallback`
  Eine Funktion, die ausgeführt wird, wenn der Dialog geschlossen wird.
- `Title`
  Die Überschrift des `Confirm`-Dialogs. Dieser Parameter ist optional. Wenn er nicht definiert wurde, ist *Confirm* die Vorbelegung.
- `buttonName`
  Dieser Parameter ist optional. Wenn er nicht definiert wurde, ist *OK,Cancel* die Vorbelegung.

Lassen Sie uns zum besseren Verständnis nun ein komplettes Beispiel anschauen.

Ein komplettes Beispiel

```
<!DOCTYPE html>
<html>

  <head>
    <title>notification.confirm-Beispiel</title>
    <script type="text/javascript" charset="utf-8"
    src="cordova-2.0.0.js"></script>
    <script type="text/javascript" charset="utf-8">
      // Warten, bis PhoneGap geladen ist
      //
      document.addEventListener("deviceready", onDeviceReady, false);
```

```
      // PhoneGap ist geladen.
      //
      function onDeviceReady() {}

      // Bearbeiten der Dialogeingabe
      function onConfirm(button) {
        alert('Sie haben ' + button + 'ausgewählt!');
      }

      // Anzeige einer confirm-Notification
      //
      function showConfirm() {
        navigator.notification.confirm(
          'Sie haben gewonnen!',     // Nachricht
          onConfirm,                 // Callback
          'Spielende',               // Titel
          'Neustart,Ende'            // buttonLabels
        );
      }
    </script>
  </head>

  <body>
    <p>
      <a href="#" onclick="showConfirm(); return false;">
        Confirm anzeigen
      </a>
    </p>
  </body>

</html>
```

Listing 5–18 *Confirm-Beispiel*

Das Beispiel zeigt sehr gut, wie einfach der Confirm-Dialog zu nutzen ist. Nachdem die PhoneGap-Bibliothek eingebunden und geladen ist, wird dem Benutzer ein Hyperlink angeboten.

```
<a href="#" onclick="showConfirm(); return false;">Confirm anzeigen</a>
```

Hiermit startet der Anwender die JavaScript-Funktion `showConfirm()`. Diese übernimmt den Aufruf des Dialogs per

```
navigator.notification.confirm(
  'Sie haben gewonnen!',     // Nachricht
  onConfirm,                 // Callback
  'Spielende',               // Titel
  'Neustart,Ende'            // buttonLabels
);
```

Dabei wird dem Anwender die Nachricht *Sie haben gewonnen* angezeigt, wobei das Dialogfenster den Titel *Spielende* trägt. Dabei kann er dann zwischen *Neustart* und *Ende* als Option wählen. Die Auswahl des Anwenders wird dann in der angegebenen Callback-Funktion `onConfirm` per HTML-`alert()`-Anweisung ausgegeben.

```
function onConfirm(button) {
   alert('Sie haben ' + button + 'ausgewählt!');
}
```

Damit kennen Sie nun die Confirm-Notification und können sie einsetzen. In den folgenden drei Abbildungen 5–10 bis 5–12 sehen Sie noch einmal die Ausgabe des Confirm-Dialogs inklusive des Aufrufes seines HTML-Pendants. Auch hier gilt wie bei `alert`, dass die Browser-Funktion `confirm` sich nicht so gut anpassen lässt wie die Funktion aus PhoneGap.

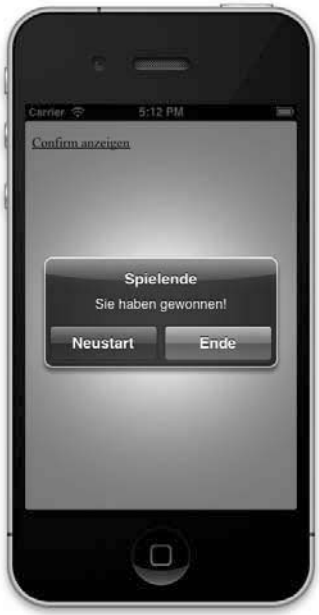

Abb. 5–10 *Confirm auf iOS*

Abb. 5–11 *Confirm-Dialog auf Android*

Abb. 5–12 *HTML-Confirm-Dialog unter android*

Achtung

Windows Phone 7 ignoriert den Parameter buttonLabels, er wird immer mit OK und $Cancel$ angezeigt.

5.4.4 notification.beep – das Smartphone piepen lassen

iOS	Android	Windows Phone	BlackBerry	webOS	Bada
x	x	x	x	–	x

Wenn Ihre Anwendung etwas mehr Akustik benötigt, um den User zu benachrichtigen, ist die beep-Notification das Richtige für Sie. Durch den Aufruf von

```
navigator.notification.beep(1);
```

wird es einen Piepton auf dem Smartphone geben. Dabei können Sie per Parameterübergabe festlegen, wie oft das Gerät piepen soll. Die Übergabe einer Zahl reicht dabei aus. Die Länge des Signals ist dabei allerdings vom Smartphone abhängig und kann nicht verändert werden.

Auch hier möchte ich Ihnen ein komplettes Beispiel vorstellen.

Ein komplettes Beispiel

```
<!DOCTYPE html>
<html>

  <head>
    <title>notification.beep-Beispiel</title>
    <script type="text/javascript" charset="utf-8"
    src="cordova-2.0.0.js"></script>
    <script type="text/javascript" charset="utf-8">
      // Warten, bis PhoneGap geladen ist
      //
      document.addEventListener("deviceready", onDeviceReady, false);

      // PhoneGap ist geladen.
      //
      function onDeviceReady() {}

      // Piepe drei mal
      //
      function playBeep() {
        navigator.notification.beep(3);
      }
    </script>
  </head>

  <body>
    <p>
      <a href="#" onclick="playBeep(); return false;">
        Beep abspielen
      </a>
    </p>
  </body>

</html>
```

Listing 5–19 *notification.beep-Beispiel*

Lassen Sie uns das Beispiel kurz durchgehen. Nach dem obligatorischen Laden der PhoneGap-Bibliothek wird dem Benutzer ein Hyperlink zur Verfügung gestellt. Dieser verweist auf die JavaScript-Funktion `playBeep()`. Die Funktion tut nichts weiter, als die Zeile `navigator.notification.beep(3)` auszuführen, und ver-

anlasst damit das Smartphone, dreimal zu piepen. Sicherlich werden Sie diese Funktion meist zusammen mit den `alert`- oder `confirm`-Dialogen nutzen. Was aber, wenn das Smartphone in der Hosentasche ist und der Piep ungehört bleibt? Dafür gibt es noch eine Benachrichtigungsform, nämlich `notification.vibrate`, das wir im nächsten Abschnitt besprechen.

5.4.5 notification.vibrate – das Smartphone vibrieren lassen

iOS	Android	Windows Phone	BlackBerry	webOS	Bada
x	x	x	x	–	x

Die sogenannte taktile Benachrichtigung ist das Vibrieren des Smartphones. Damit können Sie z.B. den Eingang einer Nachricht diskret dem Benutzer mitteilen. Der Aufruf ist wie bei seinem Vorgänger `notification.beep` sehr einfach. Es genügt ein

```
navigator.notification.vibrate(3000);
```

um die Vibration zu starten. Dabei kann durch Übergabe eines Parameters die Dauer der Vibration in Millisekunden eingestellt werden. Im obigen Aufruf sind es drei Sekunden. Schauen wir uns dazu ein komplettes Beispiel an.

Ein komplettes Beispiel

```
<!DOCTYPE html>
<html>

  <head>
    <title>notification.vibrate-Beispiel</title>
    <script type="text/javascript" charset="utf-8"
    src="cordova-2.0.0.js"></script>
    <script type="text/javascript" charset="utf-8">
      // Warten, bis PhoneGap geladen ist
      //
      document.addEventListener("deviceready", onDeviceReady, false);

      // PhoneGap ist geladen.
      //
      function onDeviceReady() {}

      // Vibriere 2 Sekunden lang
      //
      function vibrate() {
        navigator.notification.vibrate(2000);
      }
```

```
    </script>
  </head>

  <body>
    <p>
      <a href="#" onclick="vibrate(); return false;">
        Vibriere
      </a>
    </p>
  </body>

</html>
```

Listing 5–20 *vibrate-Beispiel*

Ähnlich dem Beep kann in diesem Beispiel der Anwender selber die Benachrichtigung auslösen. Nach dem Laden der PhoneGap-Bibliothek wird ein Hyperlink bereitgestellt, der die JavaScript-Funktion `vibrate()` aufruft. Diese wiederum führt nur den `vibrate`-Befehlt per

```
navigator.notification.vibrate(2000);
```

aus. Das waren auch schon die wichtigsten Zeilen des Beispiels.

Auch die `vibrate`-Benachrichtigung werden Sie wahrscheinlich in der Praxis oft in der Kombination mit den `alert`- oder `confirm`-Dialogen nutzen.

Damit haben Sie den API-Teil der Benachrichtigungen bzw. Notifications erfolgreich abgeschlossen. Im nächsten Abschnitt geht es um Geräte-Events.

5.5 Geräte-Events II – (Tasten- & System-Events)

5.5.1 batterystatus – Akku-/Gerätestatus auslesen

iOS	Android	Windows Phone	BlackBerry	Symbian	webOS	Bada
x	x	x	x	–	–	–

Nachdem Sie in Abschnitt 4.2 die wichtigsten Events von PhoneGap kennengelernt haben, möchte ich dies hier noch etwas vertiefen. Dabei geht es nicht um den Lifecycle, sondern um weitere mögliche Events. Die API stellt Ihnen auch Zugriff auf den Batteriestatus des Smartphones zur Verfügung. Dabei können Sie Änderungen des Batteriezustandes als Event gesendet bekommen. Auch hier gilt: Das Registrieren des Events erfolgt wieder per Eventlistener.

```
window.addEventListener("batterystatus", onBatteryStatus, false);
```

Zu beachten ist allerdings, dass die Callback-Funktion zusätzlich ein Objekt übergeben bekommt.