Mai 2010

Eine Einführung in FBJS

Eigene Javascript-Tabs auf Facebook Pages

von Mario Bartlack (bytepark GmbH)

Der nachfolgende Beitrag ist eine Einführung in das Thema FBJS und beantwortet, was FBJS eigentlich ist, was man damit machen kann und was nicht. Zur Demonstration sind einige kurze Beispiele und natürlich auch Programmcode eingefügt. Die Einführung richtet sich als vor allem an diejenigen, die bereits Erfahrung mit Javascript sammeln konnten und ihre Fähigkeiten auch auf der Facebook-Plattform anwenden möchten.

FBJS (Facebook Javascript) ist Facebooks Antwort für Entwickler, die Javascript in ihren Facebook-Anwendungen nutzen wollen. Auch wenn einem nicht der komplette Satz an Javascript-Techniken zur Verfügung steht, bietet Facebook dennoch sehr vielfältige Bibliotheken für verschiedenste Aufgaben an. Es gibt Methoden für DOM-Manipulationen, Animationen, Event-Handling, Dialoge sowie AJAX-Funktionalitäten. Mit diesen Werkzeugen lassen sich einfachste „Hello-World“-Funktionen aber auch komplexe und interaktive Anwendungen entwickeln.

FBJS innerhalb eines Tabs läuft in einer Sandbox. Das bedeutet, das Javascript von Facebook vor der Ausgabe verändert wird, um sicherzustellen, dass keine vorhandenen Objekte von Facebook überschrieben werden. Es dürfen also nur die Variablen und Funktionen genutzt werden, die man selbst geschrieben hat und die auch von Facebook erlaubt sind.

So wird z. B.

```
var prop = „someProperty“;
function someMethod(param) {
}
```

umgeschrieben in

```
var a12345678_prop = „someProperty“;
function a12345678_someMethod(param) {
}
```

a12345678 steht hier stellvertretend für die App-ID. Damit wird quasi ein Namespace erzwungen, in dem euer Javascript ausgeführt wird.

Einschränkungen

Den Möglichkeiten, die von Facebook bereitgestellt werden, stehen zum Teil erhebliche Einschränkungen gegenüber.

- Grundsätzlich sind keine externen Skripte erlaubt
- Es besteht kein Zugriff auf das globale window-Objekt
- Die Alert-Funktion wurde entfernt, für Benachrichtigungen kann aber sehr leicht die Dialog-Klasse genutzt werden
- Die Browser-Konsole funktioniert weiterhin
- Automatisch ausgeführtes Javascript wird unterdrückt (zur Aktivierung ist erst eine einmalige Benutzeraktion (bspw. Mausklick) erforderlich, bevor der von euch geschriebene Javascript-Block ausgeführt wird)

- Ebenso betroffen sind eventuell vorhandene Mouseover-/Mouseout-Events
- Arrays müssen als Literal erstellt werden (sollte man sowieso generell so machen)

```
var arr = new Array(); //falsch

var arr = []; //richtig
```

Durch die o. g. Einschränkungen ist leider auch der Einsatz von Javascript-Bibliotheken wie jQuery, YUI, prototype etc. nicht möglich. Manch einer von euch wird sich an dieser Stelle vielleicht fragen: „Keine externen Bibliotheken? Muss ich jetzt etwa alles alleine schreiben? Das ist mir viel zu aufwendig."

In der Tat sind mir diese Gedanken anfangs auch durch den Kopf geschossen. Glücklicherweise stellt Facebook ein Toolset zur Verfügung, mit dem man in die Lage versetzt wird, einen Großteil an allgemeinen und speziellen Aufgaben und Anforderungen zu lösen. Welche dies sind, stelle ich euch im folgenden Abschnitt vor.

DOM-Manipulationen

Viele der regulären Methoden in Javascript für die Manipulation vom DOM funktionieren weiterhin (*appendChild, insertBefore, removeChild, cloneNode*). Einige Eigenschaften wie *parentNode, nextSibling, src, href* wurden aber durch entsprechende Getter/Setter ersetzt. Andere wiederum wurden ersatzlos gestrichen (bspw. *innerHTML*).

Hier ein paar Beispiele:

Javascript	Getter	Setter
className	getClassName	setClassName
style	getStyle	setStyle
href	getHref	setHref

CSS-Styles lassen sich sehr einfach über die *setStyle*-Methode ändern.

```
obj.setStyle({color: 'black', background: 'white'});
obj.setStyle('color', 'black');
```

Achtung! Styles müssen in *CamelCase* geschrieben werden. Ansonsten funktioniert die Zuweisung nicht.

```
obj.setStyle('background-color', 'black'); //geht nicht
obj.setStyle('backgroundColor', 'black'); // geht
```

Beachtet ebenfalls die Angabe von px bei z. B. Größenangaben. Dies sollte nicht vergessen werden.

```
obj.setStyle('width', '100'); //geht nicht
obj.setStyle('width', '100px'); // geht
```

Eine Änderung, die nicht auf den ersten Blick als solche zu erkennen ist, ist die Methode *document.getElementById*. Diese kann wie gewohnt verwendet werden, nur dass die Rückgabe ein von Facebook erzeugtes Objekt ist und nicht direkt das erwartete HTML-Objekt.

Es gibt noch eine Vielzahl mehr. Eine gute Übersicht über die einzelnen Methoden und Eigenschaften findet ihr zusätzlich hier:
http://wiki.developers.facebook.com/index.php/FBJS#FBJS_DOM_Objects

Events

Events können über die W3C-Methode addEventListener hinzugefügt werden. Dabei wird das Eventobjekt als Argument an den Eventhandler übergeben. Im Eventhandler hat man dann Zugriff auf die Eigenschaften *target, type, pageX, pageY, ctrlKey, keyCode, metaKey* und *shiftKey* sowie auf die Methoden *stopPropagation* und *preventDefault*.

Das Hinzufügen von Eventlistener funktioniert so:

```
obj.addEventListener('mouseover',mouseOverHandler);
function mouseOverHandler(event) {
     //do some crazy stuff here
}
```

Animationen

Facebook stellt uns eine leistungsstarke und robuste Animationsbibliothek zur Verfügung. Mit geringem Aufwand kann man recht schnell komplexe CSS-Animationen erstellen.

Die Basissyntax sieht wie folgt aus:

```
Animation(object).to('property', 'value').go();
```

Möchte man mehrere Eigenschaften gleichzeitig animieren, verkettet man ganz einfach die Methoden dafür.

```
<a href="#" onclick="Animation(this).to('background','#aa77ff').to('color',
'#00bbcc').go(); return false;">Highlight</a>
```

Eine sehr interessante Möglichkeit für sequentielle Animationen bieten die Checkpoints. Mit diesen kann man beliebig viele Animationen hintereinander ausführen lassen. Kopiert das Skript einfach mal in eure Static-FBML-Box, klickt anschließend auf das Quadrat und schaut, was passiert.

```
<style>
* {
      width: 100%;
      height: 400px;
}
#square {
      width: 100px;
      height: 100px;
```

```
        background-color: maroon;
        position: relative;
}
</style>
<div id="square" onclick="Animation(this).to('left',
'200px').duration(2000).ease(Animation.ease.end).checkpoint().to('top',
'200px').duration(2000).ease(Animation.ease.end).go();"></div>
```

Es gibt noch einige zusätzliche Funktionen wie by, from, hide, show und blind. Eigene easing-Funktionen sind ebenfalls möglich.

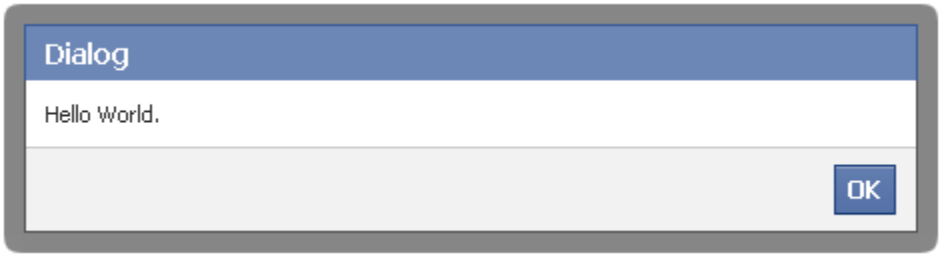
Für dieses Thema findet sich auch im Wiki eine gute Übersicht.
http://wiki.developers.facebook.com/index.php/FBJS/Animation

Dialoge

Mit der Dialog-Klasse lassen sich sehr einfach Dialogfenster im Stil von Facebook anzeigen.
Eine Auswahl zwischen den beiden Typen *Dialog.DIALOG_POP* und Dialog. *DIALOG_CONTEXTUAL* ist möglich. Es gibt Eventhandler für *onconfirm* und *oncancel*. Titel, Inhalt und die Label der Buttons sind anpassbar.

Nachfolgend seht ihr zwei Beispiele mit den dazugehörigen Dialogfenstern.

```
<a href="#" onclick="new Dialog().showMessage('Dialog', 'Hello World.');
return false;">Vanilla DIALOG_POP.</a><br />
```




```
<a href="#" onclick="new
Dialog(Dialog.DIALOG_CONTEXTUAL).setContext(this).showChoice('Dialog',
'Hello World.', 'Foo', 'Bar'); return false;">CONTEXTUAL_DIALOG with two
```

```
buttons: Foo and Bar</a><br />
```

Weitere Beispiele (inklusive die o. g.) findet ihr hier:

http://wiki.developers.facebook.com/index.php/FBJS/Examples/Dialogs

Anscheinend gibt es mit dieser Klasse aber ein Problem im IE7. In der Bugbase sind Einträge vorhanden, die noch offen sind. Testet es jedenfalls vorher in allen relevanten Browsern um keine unliebsame Überraschungen zu vermeiden.

AJAX

Facebook stellt eine leistungsstarke AJAX-Bibliothek zur Verfügung, deren Verwendung sehr einfach ist, wie das folgende Beispiel demonstriert:

```
var ajax = new Ajax();
ajax.responseType = Ajax.JSON; //Ajax.FBML, Ajax.RAW
ajax.ondone = function(data) {
     //Rückgabe vom Server
}
ajax.onerror = function() {
     //Fehler
}
var postParams = { "param1" : value1, "param2" : value2 };
var getParams = 'getparam=test';
ajax.post('http://example.com/testajax.php?'+getParams, postParams);
```

Parameter, die man per *GET* übertragen möchte, hängt man einfach an die URL ran, Parameter, die man per *POST* übertragen möchte, als zusätzlichen Parameter in der Funktion post.

Weiterführende Informationen und Beispiele findet ihr hier:

http://wiki.developers.facebook.com/index.php/FBJS/Examples/Ajax

Trotz der eingangs genannten Einschränkungen decken die Möglichkeiten, die von Facebook angeboten werden, dennoch einen sehr großen Teil von dem ab, was einem im täglichen Umgang mit Javascript über den Weg läuft. Sicher ist es so, dass man an manchen Stellen ein wenig der Komfort von den etablierten Javascript-Bibliotheken vermissen lässt. Allerdings tut dies der Entwicklung von Anwendungen mit FBJS keinen allzu großen Abbruch. Auf jeden Fall sollte man aber einen entsprechenden Editor benutzen, die Textarea auf Facebook eignet sich in keinster Weise dafür.

Damit Ihr euch ein Bild davon machen könnt, wie das letztlich alles aussieht, habe ich zwei einfache Beispiele vorbereitet, an denen ich die verschiedenen o. g. Punkte demonstrieren werde.

Die Skripte sind per Copy&Paste lauffähig. Einfach kopieren und in eure FBML-Box einfügen.

Beispiel Tabbar

Hier haben wir eine einfache Tabbar. Bei Klick auf einen Tab bewegt sich der dazugehörige Inhalt in den Anzeigebereich. Wie ihr seht, ist der FBJS Teil in diesem Fall der kleinste Teil der Anwendung.

```
<style>
	* {margin: 0; padding: 0; text-decoration: none; font-family: Arial;}
	a {color: #121212; display: block; width: 103px; height: 14px;
background-color: #f5f5f4; padding: 4px 2px;}
	li {list-style-type: none;}
	.activeTab {background-color: #3a71a9 !important; color: #fff !
important;}
	.tabbar li {float: left; text-align: center; margin-right: 1px;
cursor: pointer; font-size: 11px; height: 14px;}
	.tabbar {height: 22px;}
	.tabbar li a:hover {color: #fff; background-color: #3a71a9; text-
decoration: none;}
	#wrapper {clear: left; width: 755px; height: 400px; overflow: hidden;
position: relative;}
	#content {width: 5292px; position: relative;}
	#content li {width: 755px; height: 400px; background-color: #c2bcb4;
float: left; border-right: solid 1px black;}
</style>

<ul class="tabbar">
	<li><a id="firstTab" class="activeTab" href="#" onclick="move(this,
0);">Item 1</a></li>
```

```
      <li><a href="#" onclick="moveContent(this, 1);">Item 2</a></li>
      <li><a href="#" onclick="moveContent (this, 2);">Item 3</a></li>
      <li><a href="#" onclick="moveContent (this, 3);">Item 4</a></li>
      <li><a href="#" onclick="moveContent (this, 4);">Item 5</a></li>
      <li><a href="#" onclick="moveContent (this, 5);">Item 6</a></li>
      <li><a href="#" onclick="moveContent (this, 6);">Item 7</a></li>
</ul>
<div id="wrapper">
      <ul id="content">
            <li>Inhalt 1</li>
            <li>Inhalt 2</li>
            <li>Inhalt 3</li>
            <li>Inhalt 4</li>
            <li>Inhalt 5</li>
            <li>Inhalt 6</li>
            <li>Inhalt 7</li>
      </ul>
</div>

<script>
      var content = document.getElementById("content");
      var activeTab = document.getElementById("firstTab");

      function moveContent(tab, i) {
            if (activeTab) activeTab.removeClassName("activeTab");
            tab.setClassName("activeTab");
            activeTab = tab;
            Animation(content).to("left", (-i *
756)+"px").duration(600).ease(Animation.ease.end).go();
      }
</script>
```

Beispiel Formular

Hier habe ich ein einfaches Formular mit einer Auswahl an verschiedenen Formularelementen, welches die grundlegende Verarbeitung aufzeigt, erstellt. In diesem Beispiel verzichte ich vollständig auf die Formatierung über CSS. Hier zählt nur die Funktionalität. Dialog-Fenster sowie Ajax finden hier Verwendung. Eine sehr nützliche Helfermethode ist ‚*serialize*' für die Verarbeitung von Formularen, welche einem sehr viel Arbeit abnimmt.

```
<form id="form">
      <input id="t" name="textfield" type="text" />Text<br />
      <input id="cb1" name="checkbox1" type="checkbox" value="checkbox1"
checked="checked" />Checkbox 1<br />
      <input id="cb2" name="checkbox2" type="checkbox" value="checkbox2" /
>Checkbox 2<br />
      <input id="rb1" type="radio" name="radiogroup" value="radio1" />Radio
1<br />
      <input id="rb2" type="radio" name="radiogroup" value="radio2" />Radio
2<br />
      <select name="selectbox">
```

```
            <option value="option1">Option 1</option>
            <option value="option2">Option 2</option>
            <option value="option3">Option 3</option>
      </select>
      <br />
      <input type="button"
onclick="sendForm(document.getElementById('form')); return false;"
value="Abschicken" />
</form>

<script>
function sendForm(form) {
      var values = form.serialize(); //sehr nützliche Funktion für die
einfache Auswertung eines Formulars
      //console.log(values) //ein Blick in die Variable lohnt sich
      var error = false;
      if (values.textfield == "") { //Textfeld ist leer
            new Dialog().showMessage("Fehler", "Bitte Text eingeben");
            error = true;
      }
      if (!values["radiogroup"]) { //Kein Radiobutton ausgewählt
            new Dialog().showMessage("Fehler", "Bitte einen Radiobutton
auswählen");
            error = true;
      }

      if (error) return; //Abbruch bei Fehler

      var ajax = new Ajax();
      ajax.responseType = Ajax.JSON;
      ajax.ondone = function(data) {
            new Dialog().showMessage("Nachricht vom Server", data.message);
      }
      ajax.onerror = function() {
            new Dialog().showMessage("Fehler", "Bei der Verarbeitung ist
ein Fehler aufgetreten!");
      }
      ajax.post("http://www.domain.de/form.php", values);
}
</script>
```

Das dazugehörige PHP-Skript sieht so aus.

```
<?php
// Zugriff auf die Werte aus dem Forumlar über $_POST
$text = $_POST["textfield"]; //Wert aus dem Textfeld
$cb1 = $_POST["checkbox1"]; //Checkbox1 wurde angeklickt
$cb2 = $_POST["checkbox2"]; //Checkbox2 wurde angeklickt
$rb = $_POST["radiogroup"]; //Wert des Radiobuttons
$sb = $_POST["selectbox"]; //Wert der Selectbox

echo '{"message":"Formular wurde erfolgreich verarbeitet."}';
?>
```

Beachtet, dass Werte, die nicht im Formular angeklickt wurden (Checkboxen) auch nicht mit übertragen werden.

Ich hoffe, dass ich euch einen Überblick über FBJS verschaffen konnte und in Zukunft immer mehr Entwickler diese Technologie nutzen, um den Benutzer ein noch besseres Erlebnis auf Facebook bieten zu können. Vielen Dank für die Aufmerksamkeit.

Über den Autor:

Mario Bartlack arbeitet für bytepark (www.bytepark.de) und ist RIA-Entwickler mit Fokus auf die Flash-Plattform. Privat ist er als Moderator auf Flashhilfe.de zu finden.

FACEBOOKMARKETING.DE

Jetzt Fan werden:
facebook.com/marketingde

Philipp Roth & Jens Wiese

kontakt@facebookmarketing.de