Doing Web Apps

jQuery 3

Autor: Rüdiger Marwein
Letzte Änderung: 2014-11-12
Version: 1.1

Dieses Dokument darf – mit Nennung des Autoren - frei vervielfältigt, verändert und weitergegeben werden.

Der Inhalt ist sorgfältig recherchiert, mit dem Dokument ist jedoch keinerlei Garantie auf Fehlerfreiheit gewährleistet.

Dieser Inhalt ist unter einem Creative Commons Namensnennung Lizenzvertrag lizenziert. Um die Lizenz anzusehen, gehen Sie bitte zu http://creativecommons.org/licenses/by/2.0/de/ oder schicken Sie einen Brief an Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

Inhaltsverzeichnis

1.Einführung....3
2.Events anmelden....3
3.Events abmelden....3
4.Kurzschreibweise für Events....4
5.Event-Kontext....4
6.Events ohne Event....4
7.Events sparen....4
8.Eigene Events....5
9.Daten an DOM Elemente heften....6
10.CSS-Werte animieren....6
10.1.Animations-Geschwindigkeiten....7
11.Links zur Vertiefung....7

1.Einführung

Events in JavaScript sind in den verschiedenen Browsern unterschiedlich implementiert. jQuery vereinfacht die Verarbeitung von Events, indem es einheitliche Funktionen bereitstellt, die es überflüssig machen, sich über die Unterschiede den Kopf zu zerbrechen.

Auch Unstimmigkeiten bei Event-Namen (als Beispiel sei „transitionend“ genannt) werden intern so verarbeitet, dass man bei der Nutzung von jQuery daran nicht denken muss.

2.Events anmelden

Um eine Funktion für einen Event anzumelden, wird die Funktion on() verwendet. Seit jQuery 1.7 wird diese Funktion empfohlen und wird vermutlich die anderen existierenden Funktionen (bind, live) ablösen.

In den Untiefen des Internet werden Sie alte Beispiele mit „bind“ und „live“ finden. Seit 1.7 empfiehlt jQuery, stattdessen „on“ zu verwenden.

```
function doSomething(event) {
  alert(this.tagName + ' geklickt');
}
$('p').on('click', doSomething );
```

Beim Event-Typ-Namen wird je das „on“ (wie in „onclick“) ausgespart. Der erste Parameter ist der Event-Typ, der zweite Parameter ist die auszuführende Funktion. Sie erhält als ersten Parameter das Event-Objekt. Das ist das minimale Parameter-Setup; komplexere werden später beleuchtet.

„this“ ist das betroffene DOM-Objekt. Möchten Sie jQuery-Funktionen darauf anwenden, muss es zunächst mit der jQuery Funktion ummantelt werden.

Wichtig: Events werden nur für die Bubbling-Phase angemeldet.

Die Funktion on() ahmt bei Browser-Unterschieden, die ein „aufsteigen“ von Events nicht beherrschen , diese Funktion nach, so dass die Events in allen Browsern gleich funktionieren.

http://api.jquery.com/on/

3.Events abmelden

Um einen zugewiesenen Event wieder abzumelden, wird die Funktion off() verwendet.

```
$('p').off('click', doSomething );
```

Es ist offensichtlich, dass Events, denen eine anonyme Funktion zugeordnet wurde, auf diese Weise nicht abgemeldet werden können. Aber es können bspw. alle Events auf einmal entfernt werden.

```
$('p').off();
```

Oder alle click-Events

```
$('p').off('click', '**');
```

http://api.jquery.com/off/

4.Kurzschreibweise für Events

Für fast alle Events gibt es direkte Funktionen, die für einfache Event-Konstrukte verwendet werden können.

```
$('p').click( doSomething );
```

ist äquivalent zu

```
$('p').on('click',  doSomething );
```

Intern wird es sogar so verarbeitet. D.h. ein Event der über die Abkürzung registriert wurde, kann auch mit der Funktion „off“ wieder entfernt werden.

http://api.jquery.com/category/events/

5.Event-Kontext

Nicht in allen Situationen kann das Closure Konzept verwendet werden, um für Kontext zu sorgen. Damit die angemeldete Funktion Informationen über den Kontext erhält in dem sie angemeldetwurde, kann optional ein Daten-Objekt über die Funktion „on“ angegeben werden.

```
function doSomething(event) {
  alert('Kontext - Anzahl: ' + event.data.list.length);
}

function myFunction() {
  var theList = $('#myList').children();
  $('p').on('click', {'list': theList}, doSomething );
  // oder äquivalent
  // $('p').click( {'list': theList}, doSomething );
}
```

6.Events ohne echten Event

jQuery verwendet ein eigenes Event-Objekt, dass an die angemeldeten Funktionen übergeben wird. Es enthält alle Eigenschaften eines echten Event-Objekt. Das Original-Event-Objekt steckt zusätzlich in der Eigenschaft „originalEvent“.

Es ist möglich, Events ohne die „echte“ Aktion auszulösen.

```
$('p').on('click', function(e) {
  console.log(e);
});

window.setTimeout(function() {
  $('p').click( );
});
```

Wichtig: wird ein Event ohne die „echte“ Aktion ausgelöst, fehlt das „`originalEvent`“ Objekt im jQuery-Event-Objekt und das Event-Objekt enthält wesentlich weniger Eigenschaften, die den Event beschreiben.

Events, die regulär über JavaScript registriert wurden, werden auch hier mit aufgerufen, jedoch sind auch hier viele Eigenschaften leer, die der Event im Regelfall erhalten hätte.

7.Events sparen

Meldet man einen Event bspw an allen <li> einer <ul> Liste an ...

```
$('ul#liste > li').on('click', function(event) {
  console.log(this.tagName + ' geklickt');
});
```

… existiert diese Event-Bindung sehr oft im Speicher. Des weiteren hat obiges Beispiel den Nachteil, dass nachträglich dynamisch hinzugefügte <li> Elemente keine Event-Anmeldung besitzen und somit ausgeschlossen sind bzw nicht richtig reagieren.

```
$('ul#liste').append( $('<li>NEU</li>') );
$('ul#liste > li').last().click(); // nichts.
```

Besser ist hier, die Events vom Eltern-Element aus zu delegieren. Dafür bietet die „on"-Funktion einen optionalen selector-Parameter.

```
$('ul#liste').on('click', 'ul#liste > li', function(event) {
  console.log(this.tagName + ' geklickt');
});
```

Beim Auslösen des Events am Eltern-Element (ul#liste) werden alle enthaltenen Elemente durchlaufen und mit dem Selektor verglichen. Bei Übereinstimmung wird der Event am Kind-Element ausgelöst. „this" ist im Beispiel das <li>.

8.Eigene Events

Eigene Events lassen sich ebenfalls mit „on" an ein Element anmelden. Da sie jedoch nicht von JavaScript selbst ausgelöst werden können, muss man das mit der Funktion „trigger" selbst tun.

```
$('div').on('hutzlmohle', function(event){
  console.log(event);
});
$('div').trigger('hutzlmohle');
```

Eigene Events werden benötigt, wenn man größere Anwendungen erstellt und Zustandsänderungen als Ereignis nach außen bekannt machen möchte/muss.

Angenommen eine fiktive Funktion, die bei fertiger Initialisierung einen Event „allDone" auslöst.

```
function mitEvent(selector) {
      var liste = [];
      $(selector).find('img').each(function(key,value) {
            liste.push(value);
      });
      // alles fertig, Event triggern
      $(selector).trigger('allDone', [ liste, 1 ]);
}
$('div#galerie').on('allDone', function(event, liste, zahl) {
      console.log(event);
      console.log(liste);
      console.log(zahl);
});
mitEvent('div#galerie');
```

Mögliche Arten eigener Events sind zum Beispiel:

- Animation beendet
- Scrollen abgeschlossen
- Punktestand geändert
- Schwellwert bei einer Berechnung überschritten

9.Daten an DOM Elemente heften

Um an DOM Elementen Daten zu hinterlegen (nicht nur Text, auch Objekt etc) kann die Funktion data() genutzt werden. So lange die Seite nicht neu geladen wird, bleiben die Daten mit dem DOM-Baum verknüpft.

```
$('#spielball').data('slot', 3);
```

Über data-* Attribute können HTML-seitig auch Daten hinterlegt werden, die im Programm ausgewertet werden können.
Beispiel:

```
<img id="spielball" src="ball-klein.jpg"
     data-big="ball-groß.jpg" />

$('#spielball').mouseover(function() {
  $(this).data('small', $(this).attr('src') );
  $(this).attr('src', $(this).data('big') );
});
```

Wichtig: mit data() hinterlegte Informationen haben Vorrang vor data-* Attributen.

Es können auch Objekte an die data Eigenschaft geheftet werden.

```
var interval = window.setInterval(function() {
  console.log('Doing something');
},200);

$('#spielball').data('myInterval', interval );

$(window).click(function(e) {
  var theInterval = $('#spielball').data('myInterval');
  window.clearInterval(theInterval);
});
```

10.CSS-Werte animieren

Animationen ergeben sich durch die kontinuierliche Änderung von Größen- und Positions-Angaben. Mit anderen Worten: das Ändern von CSS-Werten in einer Schleife / einem Intervall sorgt für eine Animation.

jQuery vereinfacht das drastisch, mit der Funktion animate().

animate() wird auf ein DOM Element angewendet und ändert gezielt Eigenschaften über einen gewissen Zeitraum von einem Start-Wert (aktueller Wert) zu einem Zielwert.

Der erste Parameter ist ein Objekt mit den CSS-Zielwerten der jeweiligen Eigenschaften.

Die weiteren 4 Angaben sind optional. Es handelt sich um die Dauer der Animation, die Art der Animations-Kurve (easing), eine Funktion, die aufgerufen wird, sobald der Zielwert erreicht wurde und ein Objekt mit einer Funktion, die bei jedem Zwischenschritt aufgerufen wird.

```
$('#spielball').css({"width": "10px"});

$('#spielball').animate(
  { "width": "100px" },
  {
    duration: "0.2s",
    easing: "linear",
    complete: function() {
      console.log('FERTIG!');
    },
    step: function(now, fx) {
       console.log( fx.elem.id + ' ' + fx.prop + ': ' + now );
    }
  }
);
```

Für die einfachste Art dieser Animation genügt dieser Aufruf:

```
$('#spielball').animate({"width": "100px"});
```

10.1.Animations-Geschwindigkeiten

0.1s	Dauer in Sekunden
slow	langsame Animation
fast	schnelle Animation

Animations-Kurven

linear	Geradlinige Änderung des Wertes ohne Kurve.
swing	Langsamerer Start, Geschwindigkeit nimmt langsam zu.

Als Animations-Ziel kann auch „toggle“ angegeben werden. Der Wert wird auf 0 heruntergefahren und bei erneutem „toggle“ zum Ursprungswert zurückgeführt.

Auch generische Zielwerte wie „+=50“ sind möglich.

Weitere Animations-Kurven können als als Plugin zugeladen werden:

http://gsgd.co.uk/sandbox/jquery/easing/

```
<script
  src="http://gsgd.co.uk/sandbox/jquery/easing/jquery.easing.1.3.js"
  type="text/javascript"></script>

$('#spielball').css({"width": "10px"}, {
  duration: 1000,
  easing: "easeInOutElastic"
});
```

11.Links zur Vertiefung

- http://api.jquery.com/animate/
- http://www.robertpenner.com/easing/
- http://www.robertpenner.com/easing/easing_demo.html
- http://webdesignledger.com/tutorials/13-excellent-jquery-animation-techniques
- http://matthiasschuetz.com/jquery-tutorial-animationen-und-effekte-erzeugen
- http://addyosmani.com/blog/jquery-sprite-animation/