jQuery
Teil 2

Verstecken und Einblenden eines Containers

CODERS.BAY

Die Funktion `$(function() { })` sorgt dafür, dass der danach folgender Code erst ausgeführt wird, wenn das Dokument geladen ist.

Durch `$('#beispiel')` wird das Element mit `id="beispiel"` ausgewählt. Danach folgt nach einem Punkt, was geschehen soll: Der Methodenaufruf `hide()` versteckt das Element. Nach einem weiteren Punkt steht ein weiterer Befehl, der angibt, was nach dem Verstecken geschehen soll. `fadeIn()` dient in jQuery dazu, ein Element langsam einzublenden. Die Angabe in Klammern bestimmt die Geschwindigkeit der Animation in Millisekunden.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <style>
            #beispiel {
                width: 400px;
                background-color: orange;
                color: black;
                font: 200% sans-serif;
                text-align: center;
                padding: 20px;
            }
        </style>
    </head>
    <body>
        <div id="beispiel">Guten Morgen</div>
        <script src="jquery.js"></script>
        <script>
            $(function() {
                $('#beispiel').hide().fadeIn(4000);
            });
        </script>

    </body>
</html>
```

Elemente mit jQuery auswählen

CODERS.BAY

Man kann in jQuery Elemente ganz komfortabel über die von CSS bekannten Selektoren auswählen

- Element nach Namen selektieren: `$('p')` wählt alle p-Elemente aus
- Elemente nach Klasse selektieren: `$('.meineKlasse')`
- Nachfahren eines Elementes selektieren: `$('nav a')`
- Elemente nach Attributen auswählen: `$('a[href$=doc]')` wählt alle `a`-Elemente aus, sofern sie ein `href`-Attribut haben, dessen Wert auf `doc` endet.
- auch möglich seit CSS3: `:nth-child()`, `:nth-child(even)`,…..
  - `:even` wählt gerade Kindelemente aus
  - `:odd` wählt ungerade aus
  - `:animated` wählt nur animierte Elemente aus
- Formularelemente
  - `:radio` selektiert Radiobuttons
  - `:submit` selektiert den Submit-Button

Formatierungen zuweisen und Elementinhalte bearbeiten

Das HTML-Dokument beinhaltet zwei Überschriften und eine Liste. Nach der Einbindung von jQuery steht ein weiteres `script`-Element. Innerhalb von `$(function() {` und `});` wird die `h1`-Überschrift ausgewählt und mithilfe von `.css()` die Formatierung verändert. Übergeben werden die gewünschten Formatierungen in der Objektliteralsyntax. Das heißt, man schreibt innerhalb der runden Klammern von `css()` geschweifte Klammern, die auf- und zugehen müssen `{}`. Innerhalb dieser geschweiften Klammern wird den Eigenschaften ihr Wert zugewiesen, indem ein Doppelpunkt dazwischen angegeben wird. Mehrere Anweisungen werden durch ein Komma getrennt

Interessant ist das Setzen von Formatierungen mit jQuery dann, wenn diese Formatierung dynamisch als Reaktion auf eine Aktion des Nutzers erfolgen

In den Browserentwicklungstools kann man gut sehen, wie jQuery gewirtk hat. Dort erkennt man, dass jQuery die Formatierungen über `style`-Attribute ergänzt hat. Der Befehl `.css()` ist praktisch, um direkt CSS-Formatierungen durzuführen, was im Fall dynamischer Zuweisungen oft gemacht wird.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Formatierung anpassen</title>
    </head>
    <body>
        <h1>Eine Überschrift</h1>
        <h2>Eine weitere Überschrift</h2>
        <ul>
            <li>Fenchel</li>
            <li>Minigruke</li>
            <li>Salat</li>
        </ul>
        <script src="jquery.js"></script>
        <script>
            $(function() {
                $('h1').css({
                    'background-clor' : '#aaa',
                    'color' : 'white',
                    'border' : '20px dashed black',
                });
                $('li:even').css('font-style', 'italic');
            });
        </script>

    </body>
</html>
```

Formatierungen zuweisen und Elementinhalte bearbeiten

CODERS.BAY

Schöner ist es jedoch, die Formatierung im CSS zu belassen und statt dessen Klassen hinzuzufügen über `addClass()`

```
<style>
    .ueb {
        background-color: #aaa;
        color: white;
        border: 20px dashed black;
    }
    .bet {
        font-style: italic;
    }
</style>
```

```
$(function() {
    $('h1').addClass('ueb');
    $('li:even').addClass('bet');
});
```

Parallel zu `addClass()` zum Ergänzen von Klassen gibt es `removeClass()`, das eine Klasse entfernt. Zusätzlich existiert `toggleClass()`, mit der eine Klasse hinzugefügt wird, wenn sie nicht vorhanden ist, oder entfernt, wenn sie bereits beim Element vorhanden ist.

Inhalte verändern

Eine weiterer praktischer jQuery-Befehl ist **`html()`**. Diesen kann man nutzen, um HTML-Code in ein Element zu schreiben.

mit **`alert()`** wird eine Meldung ausgegeben, es ist ein klassischer JavaScript-Befehl für Meldungsfenster. Im Beispiel dient es dazu, das Skript anzuhalten, damit man den ursprünglichen Text des Absatzes sehen kann. Danach wird mit **`$('.weg')`** das Element mit dem Attribut **`class="weg"`** ausgewählt. Über **`html()`** wird ein neuer Text in dieses Element geschrieben

Übriges kann man `html()` nicht nur nutzen, um HTML-Code in Elemente zu schreiben. Man kann es auch zum Auslesen von HTML-Code einsetzen. Dazu braucht man nur den Parameter weglassen.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>HTML-Code schreiben</title>
        <style>
          body {
            font-size: 120%;
            font-family: sans-serif;
          }
          p {
            background-color: #ddd;
            padding: 20px;
          }
        </style>
    </head>
    <body>
        <p class="weg">Ich bin dann mal weg...</p>>
        <script src="jquery.js"></script>
        <script>
            $(function() {
                alert('Ein Klick bitte!');
                $('.weg').html('Und jetzt komme ich');
            });
        </script>
    </body>
</html>
```

Ereignisse in jQuery

CODERS.BAY

Das Interessante an JavaScript und damit auch an jQuery ist, dass man bereits im Client auf die Aktionen des Benutzers reagieren kann - man kann etwas Bestimmtes tun, wenn ein Benutzer mit dem Mauszeiger über einen Bereich der Webseite fährt, einen Bereich verlässt, etwas anklickt, eine Option bei einem Auswahlfeld wählt oder Ähnliches. All das könnte man hingegen mit PHP nicht, denn PHP kann immer erst tätig werden, nachdem der Link geklickt ist oder nachdem das Formular abgesendet wurde.

Die Aktionen des Benutzers lösen in JavaScript sogenannte Ereignisse oder *Events* aus. Die Reaktion auf diese Ereignisse nennt man Eventhandling-

Beispiel

Wenn Inhalte wechselnd ein- und ausgeblendet werden sollen, kann auf **`toggle()`** zurückgegriffen werden.

Bei einem Klick auf den ersten Absatz soll der zweite entweder ein- oder ausgeblendet werden.

Zuerst soll das Element mit **`id="inhalt"`** versteckt werden.

Außerdem soll etwas mit dem ersten Absatz geschehen. Über **`css()`** wird die Eigenschaft **`cursor`** auf **`pointer`** gesetzt. Zusätzlich soll der Klick abgefangen werden.

Genauso wie das Klickereignis kann man auch mit anderen Events arbeiten: Mit **`mouseover`** kann abgefangen werden, dass der Besucher einer Webseite die Mause über einen Bereich bewegt, mit **`mouseout`** kann darauf reagiert werden, wenn jemand eine Bereich mit der Maus wider verlässt. **`change`** erlaubt es bestimmten Code ausführen zu lassen, wenn jemand bei einer Auswahlliste eine andere Option wählt.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Ein-Ausblenden</title>
        <style>
          body {
            font-size: 120%;
            font-family: sans-serif;
          }
          p {
            background-color: #ddd;
            padding: 20px;
            width: 300px;
          }
        </style>
    </head>
    <body>
        <p class="einaus">Bitte klicken</p>
        <p class="inhalt">
           Dieser Inhalt wird wechselnd ein-ausgeblendet
        </p>
        <script src="jquery.js"></script>
        <script>
            $(function() {
                $('#inhalt').hide();
                $('#einaus').css('cursor', 'pointer')
                            .on('click', function() {
                    $('#inhalt').toggle();
                })
            });
        </script>
    </body>
</html>
```

Kurz vorgestellt: Ajax

Mit Ajax kann man Anfragen an den Server asynchron ausführen, das heißt im Hintergrund.

Ajax steht für Asynchronous JavaScript And XML. XML, weil als Dateiformat für die Antwort oft mit XML gearbeitet wurde. Der Begriff Ajax wird heute weiterverwendet, weil er kurz ist - auch wenn man meist nicht mehr XML als Rückgabeformat einsetzt, sondern beispielsweise JSON (*JavaScript Object Notation*) oder direkt Text.

Das Entscheidende an Ajax ist der erste Bestandteil der ausgeschriebenen Notation - der Begriff "asynchron". Bei der asynchronen Kommunikation erfolgt "das Senden und Empfangen von Daten zeitlich versetzt und ohne Blockieren des Prozesses durch bspw. Warten auf die Antort des Empfängers".

Setzt man Ajax ein, so kann das folgendermaßen ablaufen: Der Benutzer gibt etwas in das Formular ein oder wählt etwas aus einer Auswahlliste. Wenn er das getan hat, findet im Hintergrund eine Anfrage an den Server statt. Der Server liefert die Antwort. Währenddessen ist die Seite mit dem Formular im Browser aber nicht verschwunden (wie im klassischen Fall), sondern der Benutzer kann weiter an ihr arbeiten. Und wenn Antworten vom Server eintreffen, können diese direkt in die vorhandene Seite mit dem Formular eingefügt werden.

Das Ganze funktioniert über das asynchrones JavaScript. Mit JavaScript kann man auch Anfragen an den Server wegschicken und die Antworten, in dem Moment, in dem sie eintreffen, entgegennehmen und die Inhalte der vorhandenen Webseite aktualisieren.

Asynchron Inhalte mit GET versenden

CODERS.BAY

Es gibt ein Formular mit einer Auswahlliste. Wenn der Benutzer etwas auswählt, werden die zu seiner Auswahl passenden Inhalte auf der Seite dargestellt. Das Entscheidende: Das Ganze funktioniert, ohne dass ein Absende-Button gedrückt wurde; auch wird die Seite nicht neu geladen. Die Datenübertragung zwischen Client und Server findet im Hintergrund statt.

Im div-Element unterhalb des Formulars sollen im Beispiel die Ergebnisse der Abfrage angezeigt werden.

```
<form>
    <p>
        <label for="art">Pflanzenart</label>:
        <selet id="art" name="art">
            <option value="">wähle</option>
            <option value="Nutzpflanze">Nutzpflanze</option>
            <option value="Zierpflanze">Zierpflanze</option>
        </selet>
    </p>
</form>
<div id="ausgabe"></div>
```

In diesem Beispiel wird der Eventhandler **change** verwendet. Damit reagieren wir darauf, dass eine Änderung beim Auswahlfeld mit **id="art"** ausgeführt wird.

Zuerst wird überprüft, ob der aktuelle Wert nicht gleich dem Leerstring ist. **val()** ist ein jQuery-Befehl, um den Wert (Inhalt) von Formularfeldern auszulesen.

Die Aktion wird ausgeführt, wenn es zur Überprüfung kommt mit **if($(this).val() != "")**
Nur wenn eine Option mit einem Wert für **value** ausgewählt wurde, wird der Code ausgeführt

```
$(function() {
    $('#art').on('change', function() {
        if($(this).val != "") {
            $.get(
                "daten.php",
                { wahl: $(this).val() },
                function(daten) {
                    $('#ausgabe').html(daten);
                }
            );
        }
    });
});
```

Asynchron Inhalte mit GET versenden

CODERS.BAY

`$.get()` ist eine jQuery-Methode, um GET-Requests asynchron durzuführen. Sie erwartet mehrere Parameter:

- Zuerst den Namen des Skripts, das asynchron aufgerufen werden soll. Im Beispiel heißt es *daten.php*
- Als weiten Parameter können Daten übergeben werden, die an das Skript übergeben werden sollen. Im Beispiel wird ein Parameter mit dem Namen `wahl` übergeben
- Als dritten Parameter gibt es eine Funktion, die aufgerufen wird, wenn die Antwort des Servers eintrifft. Diese Funktion enthält als Parameter die Daten, die das Skript zurückliefert; was mit ihnen geschehen soll, steht im Funktionsrumpf

Das PHP-Skript muss die per GET übertragenen Daten entgegennehmen und darauf basierend Inhalte zurück liefern

Asynchron Inhalte mit GET versenden

Zuerst wird überprüft, ob der GET-Parameter gesetzt ist und den Wert Nutzpflanzen hat. Ist dies der Fall, wird ein Array mit Nutzpflanzen definiert.

Im Beispiel wird direkt ein Array definiert. An dieser Stelle könnte natürlich auch eine Datenbankabfrage stattfinden, die eine Ergebnismenge zurückliefert

Dann wird eine Funktion **`ausgeben()`** aufgerufen, die für eine ordentliche Ausgabe sorgt.

Wenn das Skript pur aufgerufen wird, also beispielsweise mit **`daten.php?wahl=Nutzpflanzen`**, so liefert es - ohne sonstigen HTML-Code drum herum - eine Liste mit den angegebenen Nutzpflanzen.

Im Skript jedoch wird es per JavaScript/jQuery aufgerufen. JavaScript/jQuery wartet auf die Rückgabe des PHP-Skripts und baut den zurückgegebenen Inhalt in das Element mit **`id="ausgabe"`** ein.

Beachte aber, dass dieses Skript wirklich nur aus dem Im Listing gezeigten Code besteht. Es beinhaltet kein HTML-Grundgerüst. Das ist typisch für Daten, die per Ajax angefordert werden.

```php
<?php
// daten.php
if((isset($_GET["wahl"])) && ($_GET["wahl"] == "Nutzpflanze")) {
    $pflanzen = [
        "moschus Erdbeere",
        "Chester Thornless",
        "Annelises Rudolph",
        "Roter Weinbergpfirsich",
        "Pfälzer Fruchtfeige"
    ];
    echo ausgeben($pflanzen);
} else if((isset($_GET["wahl"])) && ($_GET["wahl"] == "Zierpflanze
n")) {
    $pflanzen = [
        "Gelber Frauenshuh",
        "Knabenkraut",
        "Duftschneeball",
        "Stundentenblume",
        "Storchschnabel"
    ];
    echo ausgeben($pflanzen);
}
function ausgeben($was) {
    $str = "<ul>";
    foreach($was as $w) {
        $str .= "<li>$w</li>";
    }
    $str .= "</ul>";
    return $str;
}
```

AJAX: FORMULARDATEN PER POST VERSENDEN

Zuerst speichern wir einen Zugriff auf das Formular in der Variablen $form. Mit der **on()**-Methode reagieren wir auf das Submit-Ereignis. Hier wird das Ergebnis **submit** übergeben und als zweiten Parameter eine anonyme Funktion mit dem Parameter e. Dieser repräsentiert das Ereignis selbst. Wir könnten darüber weitere Informationen über das Ereignis erhalten. Im Beispiel hier der benötigt, um die Standardaktion mit **preventDefault()** zu unterbinden. Die Standardaktion bei einem **submit** ist ein Absenden des Formulars. Da wir den Versand jedoch selbst übernehmen wollen, müssen wir die Standardaktion außer Kraft setzen. Danach lesen wir die URL aus dem Formular aus. Für den Zugriff auf Attribute bietet jQuery die Methode **attr()**, der man als Attribut übergibt, das was man auslösen möchte. Danach kommt **$.post()**, über das sich ein POST-Request per Ajax durchführen lässt. Zuerst übergeben wir den Skriptnamen (in der Variablen **url** gespeichert), das per Ajax aufgerufen werden soll. Dann folgen die Daten, die verssendet werden sollen. Dafür kann man **serialize()** verwenden, das alle Formulardaten in serialisierter Form weiterreicht.

Die darauffolgende anonyme Funktion wird aufgerufen, wenn der Datenversand erfolgreich war, und enthält die Daten aus dem externen Script. Diese erscheinen jetzt im **.container**-Element und ersetzen dort vorher stehende Formular

```html
<div class="container">
    <form id="meinformular" action="send.php" method="post">
        <div class="textfeld">
            <label for="Vorname">Vorname</label>
            <input type="text" id="Vorname"  name="Vorname">
        </div>
        <div class="textfeld">
            <label for="Nachname">Nachname</label>
            <input type="text" id="Nachname"  name="Nachname">
        </div>
        <div>
            <input type="submit" value="Absenden">
        </div>
    </form>
</div>
```

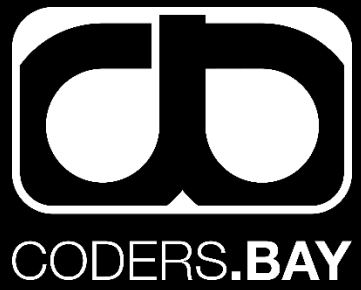
```javascript
var $form = $('#meinformular');
$form.on('submit', function (e) {
    e.preventDefault();
    var url = $form.attr('action');
    $.post(
        url,
        $form.serialize(),
        function(daten) {
            $('.container').html('Vielen Dank: ' + daten);
        }
    )
});
```

AJAX: FORMULARDATEN PER POST VERSENDEN

CODERS.BAY

Im PHP-Skript werden alle erhaltenen Daten ausgegeben - sie erscheinen dann wieder auf der ursprünglichen Seite. Hier würde bei echten Beispielen eine Aktion wie ein E-Mai-Versand oder das Eintragen der Daten in ein Formular stattfinden - und die Meldung "Vielen Dank für ihre Anfrage" o. Ä. würde als Anzeigen genügen.
Wie du siehst, macht jQuery die Arbeit mit Ajax sehr einfach, man muss sich nicht um Details kümmern, sondern nur die notwenigsten Angaben hinschreiben - alles andere geschieht im Hintergrund

Wenn du nicht jQuery nutzen willst, aber trotzdem Unterstützung bei Ajax suchen, so solltest du dir einmal axio ansehen.



Quelle

Quelle: Maurice, Florence. PHP 7 und MySQL: Ihr praktischer Einstieg in die Programmierung dynamischer Websites
(German Edition) dpunkt.verlag. Kindle-Version.