**Kostenloses eBook**

# LERNEN jQuery

Free unaffiliated eBook created from
**Stack Overflow contributors.**

#jquery

# Inhaltsverzeichnis


Über ... 1

Kapitel 1: Erste Schritte mit jQuery ... 2

Bemerkungen ... 2

Versionen ... 2

Examples ... 3

jQuery-Namespace ("jQuery" und "$") ... 3

Fertig machen ... 4

Erklärung des Codes ... 4

Script-Tag im Kopf der HTML-Seite einfügen ... 5

Namensraumkollisionen vermeiden ... 6

Laden von jQuery über die Konsole auf einer Seite, auf der es nicht vorhanden ist. ... 8

Das jQuery-Objekt ... 8

Laden von Namespace-Plugins ... 9

Kapitel 2: Ajax ... 11

Syntax ... 11

Parameter ... 11

Bemerkungen ... 11

Examples ... 12

Behandlung von HTTP-Antwortcodes mit $ .ajax () ... 12

Ajax zum Senden eines Formulars verwenden ... 12

Senden von JSON-Daten ... 13

Alles in einem Beispielen ... 14

Ajax Abbruch eines Anrufs oder einer Anfrage ... 16

Ajax-Datei-Uploads ... 16

1. Ein einfaches vollständiges Beispiel ... 16

2. Arbeiten mit Dateieingaben ... 17

3. Erstellen und Füllen der FormData ... 17

4. Senden der Dateien mit Ajax ... 18

Kapitel 3: Anhängen ... 19

Syntax ... 19

Parameter ... 19
Bemerkungen ... 19
Examples ... 19
Ein Element an einen Container anhängen ... 19
Effiziente fortlaufende Verwendung von .append () ... 20
HTML ... 20
JS ... 20
Mach das nicht. ... 21
Fügen Sie zu einem separaten Array hinzu, und fügen Sie die Schleife nach Abschluss an ... 21
Verwenden moderner Array. * -Methoden ... 22
Verwenden von HTML-Zeichenfolgen (anstelle der integrierten Methoden von jQuery) ... 22
Elemente manuell erstellen, an Fragment des Dokuments anhängen ... 23
Tauchen Sie tiefer ein ... 24
jQuery anhängen ... 24
Kapitel 4: Attribute ... 25
Bemerkungen ... 25
Examples ... 25
Ruft den Attributwert eines HTML-Elements ab ... 25
Einstellungswert des HTML-Attributs ... 26
Attribut entfernen ... 26
Differenz zwischen attr () und prop () ... 26
Kapitel 5: Breite und Höhe eines Elements ermitteln und einstellen ... 27
Examples ... 27
Breite und Höhe abrufen und einstellen (Rand ignorieren) ... 27
InnerWidth und innerHeight abrufen und einstellen ... 27
OuterWidth und outerHeight abrufen und einstellen ... 27
Kapitel 6: CSS-Manipulation ... 29
Syntax ... 29
Bemerkungen ... 29
Examples ... 30
Legen Sie die CSS-Eigenschaft fest ... 30

CSS-Eigenschaft abrufen ..... 30
Numerische Eigenschaften für Inkrement / Dekrement ..... 30
CSS - Getter und Setter ..... 31
CSS-Getter ..... 31
CSS Setter ..... 31
Kapitel 7: dokumentbereites Ereignis ..... 33
Examples ..... 33
Was ist dokumentenbereit und wie soll ich es verwenden? ..... 33
jQuery 2.2.3 und früher ..... 34
jQuery 3.0 ..... 34
Notation ..... 34
Asynchron ..... 34
Unterschied zwischen $ (document) .ready () und $ (window) .load () ..... 35
Ereignisse anfügen und das DOM in ready () bearbeiten ..... 35
Unterschied zwischen jQuery (fn) und vorherigem Ausführen Ihres Codes ..... 36
Kapitel 8: DOM Traversing ..... 38
Examples ..... 38
Wählen Sie die untergeordneten Elemente des Elements aus ..... 38
Iteration über eine Liste von jQuery-Elementen ..... 38
Geschwister auswählen ..... 39
nächste () Methode ..... 39
Holen Sie sich das nächste Element ..... 40
Vorheriges Element abrufen ..... 41
Auswahl filtern ..... 41
Das HTML ..... 41
Wähler ..... 42
Funktion ..... 42
Elemente ..... 42
Auswahl ..... 42
find () -Methode ..... 42
Kapitel 9: DOM-Manipulation ..... 44
Examples ..... 44

Erstellen von DOM-Elementen ... 44
Elementklassen bearbeiten ... 44
Andere API-Methoden ... 47
.html () ... 48
Elemente sortieren ... 48
Mach es niedlich ... 49
Fügen Sie eine Sortierschaltfläche hinzu ... 50
Legen Sie den Anfangswert der Sortierrichtung fest ... 50
Zwischenspeichern Sie unsere DOM-Elemente und sortList() hier, um die DOM-Verarbeitung zu ... 50
Wickeln Sie alles in eine doSort() Funktion ... 50
Klick-Handler für $('#btn-sort') ... 50
Jetzt alle zusammen ... 50
Mehrstufige Sortierung (Gruppierung sortierter Elemente) ... 51
Fügen Sie eine weitere Schaltfläche hinzu, um die deaktivierte Gruppensortierung umzuschal ... 52
Kapitel 10: Element Sichtbarkeit ... 53
Parameter ... 53
Examples ... 53
Überblick ... 53
Möglichkeiten umschalten ... 53
Kapitel 11: Jede Funktion ... 56
Examples ... 56
Grundlegende Verwendung ... 56
jQuery jede Funktion ... 56
Kapitel 12: jQuery .animate () - Methode ... 57
Syntax ... 57
Parameter ... 57
Examples ... 58
Animation mit Rückruf ... 58
Kapitel 13: jQuery Verschobene Objekte und Versprechen ... 60
Einführung ... 60
Examples ... 60

Grundversprechen schaffen ... 60
Asynchrone Versprechen Verkettung ... 60
jQuery ajax () Erfolg, Fehler VS .done (), .fail () ... 61
Holen Sie sich den aktuellen Stand eines Versprechens ... 62
Kapitel 14: Kontrollkästchen Alle auswählen, wenn andere Kontrollkästchen geändert werden ... 63
Einführung ... 63
Examples ... 63
2 Markieren Sie alle Kontrollkästchen mit den entsprechenden Kontrollkästchen ... 63
Kapitel 15: Plugins ... 64
Examples ... 64
Plugins - Erste Schritte ... 64
jQuery.fn.extend () -Methode ... 66
Kapitel 16: Selektoren ... 67
Einführung ... 67
Syntax ... 67
Bemerkungen ... 67
Examples ... 67
Arten von Selektoren ... 68
Selektoren kombinieren ... 70
Nachkommen- und Kindselektoren ... 71
Andere Kombinatoren ... 71
Gruppenauswahl: "," ... 71
Mehrfachauswahl: "" (kein Zeichen) ... 72
Benachbarte Geschwisterauswahl: "+" ... 72
Allgemeine Geschwisterauswahl: "~" ... 72
Überblick ... 72
Grundlegende Selektoren ... 72
Beziehungsoperatoren ... 72
Caching-Selektoren ... 73
DOM-Elemente als Selektoren ... 74
HTML-Strings als Selektoren ... 74

Kapitel 17: Veranstaltungen ... 76
Bemerkungen ... 76
Examples ... 76
Event-Handler anhängen und trennen ... 76
Hängen Sie einen Ereignishandler an ... 76
HTML ... 76
jQuery ... 76
Trennen Sie einen Ereignishandler ... 76
HTML ... 76
jQuery ... 77
jQuery ... 77
Delegierte Ereignisse ... 77
Beispiel HTML ... 77
Das Problem ... 77
Hintergrundinformationen - Weitergabe von Ereignissen ... 78
Lösung ... 78
Im Detail, wie die Lösung funktioniert ... 79
Ereignis zum Laden von Dokumenten .load () ... 79
Ereignisse zum Wiederholen von Elementen ohne Verwendung von IDs ... 80
originalEvent ... 81
Scroll-Richtung abrufen ... 81
Ein- und Ausschalten bestimmter Ereignisse über jQuery. (Named Listeners) ... 81
Kapitel 18: Voranstellen ... 83
Examples ... 83
Ein Element an einen Container übergeben ... 83
Methode voranstellen ... 83
Credits ... 85

# Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: jquery

It is an unofficial and free jQuery ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official jQuery.



Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

# Kapitel 1: Erste Schritte mit jQuery

## Bemerkungen

jQuery ist eine JavaScript-Bibliothek, die DOM-Vorgänge, Ereignisbehandlung, AJAX und Animationen vereinfacht. Außerdem werden viele Probleme mit der Browserkompatibilität in zugrunde liegenden DOM- und Javascript-Engines behoben.

Jede Version von jQuery kann von https://code.jquery.com/jquery/ sowohl im komprimierten (minimierten) als auch im unkomprimierten Format heruntergeladen werden.

## Versionen

| Ausführung | Anmerkungen | Veröffentlichungsdatum |
|---|---|---|
| 1,0 | Erste stabile Veröffentlichung | 2006-08-26 |
| 1.1 | | 2007-01-14 |
| 1.2 | | 2007-09-10 |
| 1.3 | Sizzle in den Kern eingeführt | 2009-01-14 |
| 1.4 | | 2010-01-14 |
| 1,5 | Deferred Callback Management, ajax Modul umschreiben | 2011-01-31 |
| 1.6 | Deutliche Leistungssteigerungen bei den `attr()` und `val()` | 2011-05-03 |
| 1.7 | Neue Ereignis-APIs: Ein `on()` und `off()` . | 2011-11-03 |
| 1.8 | Sizzle neu geschrieben, verbesserte Animationen und `$(html, props)` Flexibilität. | 2012-08-09 |
| 1,9 | Entfernen von veralteten Schnittstellen und Code-Bereinigung | 2013-01-15 |
| 1.10 | Inkorporierte Fehlerbehebungen und Unterschiede aus den Betazyklen 1.9 und 2.0 | 2013-05-24 |
| 1.11 | | 2014-01-24 |
| 1.12 | | 2016-01-08 |
| 2,0 | Unterstützung für IE 6–8 für | 2013-04-18 |

| Ausführung | Anmerkungen | Veröffentlichungsdatum |
|---|---|---|
| | Leistungsverbesserungen und Größenreduzierung gestrichen | |
| 2.1 | | 2014-01-24 |
| 2.2 | | 2016-01-08 |
| 3,0 | Massive Beschleunigungen für einige benutzerdefinierte jQuery-Selektoren | 2016-06-09 |
| 3.1 | Keine stillen Fehler mehr | 2016-07-07 |

# Examples

## jQuery-Namespace ("jQuery" und "$")

`jQuery` ist der Ausgangspunkt zum Schreiben von jQuery-Code. Es kann als Funktion `jQuery(...)` oder als Variable `jQuery.foo` .

`$` ist ein Alias für `jQuery` und die beiden können normalerweise gegeneinander ausgetauscht werden (außer wenn `jQuery.noConflict();` verwendet wurde - siehe Namensraumkollisionen `jQuery.noConflict();` ).

Angenommen, wir haben dieses HTML-Snippet -

```
<div id="demo_div" class="demo"></div>
```

Wir möchten vielleicht jQuery verwenden, um diesem Div etwas Textinhalt hinzuzufügen. Dazu können wir die jQuery-Funktion `text()` verwenden. Dies könnte entweder mit `jQuery` oder `$` . dh -

```
jQuery("#demo_div").text("Demo Text!");
```

Oder -

```
$("#demo_div").text("Demo Text!");
```

Beides führt zum gleichen abschließenden HTML -

```
<div id="demo_div" class="demo">Demo Text!</div>
```

Da `$` prägnanter als `jQuery` , ist dies die bevorzugte Methode zum Schreiben von jQuery-Code.

jQuery verwendet CSS-Selektoren, und im obigen Beispiel wurde ein ID-Selektor verwendet. Weitere Informationen zu Selektoren in jQuery finden Sie unter Selektortypen .

### Fertig machen

Erstellen Sie eine Datei `hello.html` mit folgendem Inhalt:

```
<!DOCTYPE html>
<html>
<head>
    <title>Hello, World!</title>
</head>
<body>
    <div>
        <p id="hello">Some random text</p>
    </div>
    <script src="https://code.jquery.com/jquery-2.2.4.min.js"></script>
    <script>
        $(document).ready(function() {
            $('#hello').text('Hello, World!');
        });
    </script>
</body>
</html>
```

Live Demo auf JSBin

Öffnen Sie diese Datei in einem Webbrowser. Als Ergebnis sehen Sie eine Seite mit dem Text: `Hello, World!`

## Erklärung des Codes

1. Lädt die jQuery-Bibliothek vom jQuery- CDN :

   ```
   <script src="https://code.jquery.com/jquery-2.2.4.min.js"></script>
   ```

   Dadurch wird die globale Variable `$` eingeführt, ein Alias für die `jQuery` Funktion und den Namespace.

   > *Beachten Sie, dass einer der häufigsten Fehler beim Einschließen von jQuery das Laden der Bibliothek fehlschlägt, BEVOR andere Skripts oder Bibliotheken, die davon abhängig sind oder von ihr Gebrauch machen.*

2. Gibt eine Funktion zurück, die ausgeführt werden soll, wenn das DOM ( Document Object Model ) von jQuery als "ready" erkannt wird:

   ```
   // When the `document` is `ready`, execute this function `...`
   $(document).ready(function() { ... });

   // A commonly used shorthand version (behaves the same as the above)
   $(function() { ... });
   ```

3. Sobald das DOM bereit ist, führt jQuery die oben gezeigte Rückruffunktion aus. In unserer Funktion gibt es nur einen Aufruf, der zwei Hauptaufgaben erledigt:

1. Ruft das Element mit dem `id` Attribut gleich `hello` (unser Selektor `#hello` ). Die Verwendung eines Selektors als übergebenes Argument ist der Kern der Funktionalität und Benennung von jQuery. Die gesamte Bibliothek entwickelte sich im Wesentlichen aus der Erweiterung des MDN document.querySelectorAll .

2. Setzen Sie den `text()` innerhalb des ausgewählten Elements auf `Hello, World!` .

```
#    ↓ - Pass a `selector` to `$` jQuery, returns our element
$('#hello').text('Hello, World!');
#             ↑ - Set the Text on the element
```

Weitere Informationen finden Sie auf der Seite jQuery - Dokumentation .

## Script-Tag im Kopf der HTML-Seite einfügen

Um **jQuery** vom offiziellen CDN zu laden, **rufen Sie** die jQuery- Website auf . Sie sehen eine Liste verschiedener Versionen und Formate.


Kopieren Sie nun die Quelle der jQuery-Version, die Sie laden möchten. Angenommen, Sie möchten **jQuery 2.X** laden, klicken Sie auf **unkomprimiertes** oder **vermindertes** Tag, das Ihnen etwa **Folgendes anzeigt** :


Kopieren Sie den vollständigen Code (oder klicken Sie auf das Kopiersymbol) und fügen Sie ihn in den `<head>` oder `<body>` Ihrer HTML-Datei ein.

Es empfiehlt sich, externe JavaScript-Bibliotheken mit dem Attribut `async` am head-Tag zu laden. Hier ist eine Demonstration:

```
<!DOCTYPE html>
    <html>
      <head>
         <title>Loading jquery-2.2.4</title>
         <script src="https://code.jquery.com/jquery-2.2.4.min.js" async></script>
      </head>
      <body>
          <p>This page is loaded with jquery.</p>
      </body>
   </html>
```

Wenn Sie ein `async` Attribut verwenden, sollten Sie sich bewusst sein, da die Javascript-Bibliotheken dann asynchron geladen und ausgeführt werden, sobald sie verfügbar sind. Wenn zwei Bibliotheken enthalten sind, in denen die zweite Bibliothek von der ersten Bibliothek abhängt, wird in diesem Fall die zweite Bibliothek geladen und vor der ersten Bibliothek ausgeführt, und es kann ein Fehler ausgegeben werden, und die Anwendung kann beschädigt werden.

## Namensraumkollisionen vermeiden

Andere Bibliotheken als jQuery können auch `$` als Alias verwenden. Dies kann zu Interferenzen zwischen diesen Bibliotheken und jQuery führen.

So geben Sie `$` zur Verwendung mit anderen Bibliotheken frei:

```
jQuery.noConflict();
```

Nach dem Aufruf dieser Funktion ist `$` kein Alias für `jQuery` . Sie können jedoch weiterhin die Variable `jQuery` selbst verwenden, um auf die Funktionen von `jQuery` zuzugreifen:

```
jQuery('#hello').text('Hello, World!');
```

Optional können Sie eine andere Variable als Alias für jQuery zuweisen:

```
var jqy = jQuery.noConflict();
jqy('#hello').text('Hello, World!');
```

Um zu verhindern, dass andere Bibliotheken mit jQuery interferieren, können Sie den jQuery-Code in einen sofort aufgerufenen Funktionsausdruck (IIFE) einschließen und `jQuery` als Argument übergeben:

```
(function($) {
    $(document).ready(function() {
        $('#hello').text('Hello, World!');
    });
})(jQuery);
```

In diesem IIFE ist `$` nur ein Alias für jQuery.

Eine weitere einfache Möglichkeit, **um den `$` Alias von jQuery** zu **sichern und sicherzustellen, dass DOM bereit ist** :

```
jQuery(function( $ ) { // DOM is ready
   // You're now free to use $ alias
   $('#hello').text('Hello, World!');
});
```

Zusammenfassen,

- `jQuery.noConflict()` : `$` bezieht sich nicht mehr auf jQuery, während die Variable `jQuery` tut.
- `var jQuery2 = jQuery.noConflict()` - `$` bezieht sich nicht mehr auf jQuery, während die Variable `jQuery` und die Variable `jQuery2` dies tut.

Nun gibt es ein drittes Szenario - Was ist, wenn wir möchten, dass jQuery **nur in `jQuery2`** verfügbar ist? Benutzen,

```
var jQuery2 = jQuery.noConflict(true)
```

Dies führt dazu, dass sich weder `$` noch `jQuery` auf `jQuery` beziehen.

Dies ist nützlich, wenn mehrere Versionen von jQuery auf dieselbe Seite geladen werden sollen.

```
<script src='https://code.jquery.com/jquery-1.12.4.min.js'></script>
```

```
<script>
    var jQuery1 = jQuery.noConflict(true);
</script>
<script src='https://code.jquery.com/jquery-3.1.0.min.js'></script>
<script>
    // Here, jQuery1 refers to jQuery 1.12.4 while, $ and jQuery refers to jQuery 3.1.0.
</script>
```

https://learn.jquery.com/using-jquery-core/avoid-conflicts-other-libraries/

### Laden von jQuery über die Konsole auf einer Seite, auf der es nicht vorhanden ist.

Manchmal muss man mit Seiten arbeiten, die `jQuery` nicht verwenden, während die meisten Entwickler `jQuery` Hand haben.

In solchen Situationen können Sie die `Chrome Developer Tools` Konsole ( `F12` ) verwenden, um `jQuery` manuell auf einer geladenen Seite hinzuzufügen, indem Sie Folgendes `jQuery` :

```
var j = document.createElement('script');
j.onload = function(){ jQuery.noConflict(); };
j.src = "https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js";
document.getElementsByTagName('head')[0].appendChild(j);
```

Die gewünschte Version kann von der obigen Version ( `1.12.4` ) abweichen. Den Link für die `1.12.4` Version finden Sie hier .

### Das jQuery-Objekt

Bei jedem Aufruf von jQuery wird mithilfe von `$()` oder `jQuery()` intern eine `new` Instanz von `jQuery` . Dies ist der Quellcode, der die neue Instanz zeigt:

```
// Define a local copy of jQuery
jQuery = function( selector, context ) {

    // The jQuery object is actually just the init constructor 'enhanced'
    // Need init if jQuery is called (just allow error to be thrown if not included)
    return new jQuery.fn.init( selector, context );
}
```

Intern bezeichnet jQuery seinen Prototyp als `.fn` , und der hier verwendete Stil der internen Instantiierung eines jQuery-Objekts ermöglicht es, diesen Prototyp `.fn` , ohne dass der Aufrufer ausdrücklich etwas `new` .

Neben dem Einrichten einer Instanz (wie die jQuery-API, z. B. `.each` , `children` , `filter` usw., `.each` wird), erstellt jQuery intern auch eine `.each` Struktur, die mit dem Ergebnis des Selektors übereinstimmt (vorausgesetzt, dies ist `.each` ) etwas anderes als nichts, `undefined` , `null` oder ähnliches wurde als Argument übergeben. Bei einem einzelnen Element enthält diese Array-ähnliche Struktur nur dieses Element.

Eine einfache Demonstration wäre, ein Element mit einer ID zu finden und dann auf das jQuery-

Objekt zuzugreifen, um das zugrunde liegende DOM-Element zurückzugeben (dies funktioniert auch, wenn mehrere Elemente übereinstimmen oder vorhanden sind).

```
var $div = $("#myDiv");//populate the jQuery object with the result of the id selector
var div = $div[0];//access array-like structure of jQuery object to get the DOM Element
```

## Laden von Namespace-Plugins

Stellen Sie beim Laden von Plugins normalerweise sicher, dass das Plugin immer *nach* jQuery eingefügt wird.

```
<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
<script src="some-plugin.min.js"></script>
```

Wenn Sie mehrere Versionen von jQuery verwenden müssen, *müssen* Sie die Plugins *nach* der erforderlichen Version von jQuery laden, gefolgt von Code, um `jQuery.noConflict(true)` . Laden Sie dann die nächste Version von jQuery und die zugehörigen Plugins:

```
<script src="https://code.jquery.com/jquery-1.7.0.min.js"></script>
<script src="plugin-needs-1.7.min.js"></script>
<script>
// save reference to jQuery v1.7.0
var $oldjq = jQuery.noConflict(true);
</script>
<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
<script src="newer-plugin.min.js"></script>
```

Wenn Sie nun die Plugins initialisieren, müssen Sie die zugehörige jQuery-Version verwenden

```
<script>
// newer jQuery document ready
jQuery(function($){
  // "$" refers to the newer version of jQuery
  // inside of this function

  // initialize newer plugin
  $('#new').newerPlugin();
});

// older jQuery document ready
$oldjq(function($){
  // "$" refers to the older version of jQuery
  // inside of this function

  // initialize plugin needing older jQuery
  $('#old').olderPlugin();
});
</script>
```

Es ist möglich, nur eine Document Ready-Funktion zu verwenden, um beide Plugins zu initialisieren. Um jedoch Verwirrung und Probleme mit zusätzlichem jQuery-Code in der Document Ready-Funktion zu vermeiden, sollten die Verweise besser getrennt bleiben.

Erste Schritte mit jQuery online lesen: https://riptutorial.com/de/jquery/topic/211/erste-schritte-mit-jquery

# Kapitel 2: Ajax

## Syntax

- var jqXHR = $ .ajax (URL [, Einstellungen])
- var jqXHR = $ .ajax ([Einstellungen])
- jqXHR.done (Funktion (data, textStatus, jqXHR) {});
- jqXHR.fail (Funktion (jqXHR, textStatus, errorThrown) {});
- jqXHR.always (Funktion (jqXHR) {});

## Parameter

| Parameter | Einzelheiten |
|---|---|
| URL | Gibt die URL an, an die die Anfrage gesendet wird |
| die Einstellungen | ein Objekt, das zahlreiche Werte enthält, die das Verhalten der Anforderung beeinflussen |
| Art | Die für die Anforderung zu verwendende HTTP-Methode |
| Daten | Daten, die von der Anfrage gesendet werden sollen |
| Erfolg | Eine Rückruffunktion, die aufgerufen wird, wenn die Anforderung erfolgreich ist |
| Error | Ein Rückruf zur Fehlerbehandlung |
| Statuscode | Ein Objekt mit numerischen HTTP-Codes und Funktionen, die aufgerufen werden sollen, wenn die Antwort den entsprechenden Code enthält |
| Datentyp | Der Datentyp, den Sie vom Server erwarten |
| Inhaltstyp | Inhaltstyp der Daten, die an den Server gesendet werden sollen. Standard ist "application / x-www-form-urlencoded; charset = UTF-8". |
| Kontext | Gibt den Kontext an, der in Callbacks verwendet werden soll. `this` bezieht sich normalerweise auf das aktuelle Ziel. |

## Bemerkungen

AJAX steht für **einen** synchronen **J** avascript nd **X** ML. AJAX ermöglicht es einer Webseite, eine asynchrone HTTP (AJAX) -Anforderung an den Server auszuführen und eine Antwort zu erhalten, ohne die gesamte Seite neu laden zu müssen.

# Examples

## Behandlung von HTTP-Antwortcodes mit $ .ajax ()

Zusätzlich zu `.done` , `.fail` und `.always` versprechen Callbacks, die `.done` , ob die Anforderung erfolgreich war oder nicht, ausgelöst werden, die Option, eine Funktion auszulösen, wenn ein bestimmter HTTP-Statuscode vom Server zurückgegeben wird. Dies kann mit dem Parameter `statusCode` .

```
$.ajax({
    type: {POST or GET or PUT etc.},
    url:  {server.url},
    data: {someData: true},
    statusCode: {
        404: function(responseObject, textStatus, jqXHR) {
            // No content found (404)
            // This code will be executed if the server returns a 404 response
        },
        503: function(responseObject, textStatus, errorThrown) {
            // Service Unavailable (503)
            // This code will be executed if the server returns a 503 response
        }
    }
})
.done(function(data){
    alert(data);
})
.fail(function(jqXHR, textStatus){
    alert('Something went wrong: ' + textStatus);
})
.always(function(jqXHR, textStatus) {
   alert('Ajax request was finished')
});
```

In der offiziellen jQuery-Dokumentation heißt es:

> Wenn die Anforderung erfolgreich ist, übernehmen die Statuscodefunktionen dieselben Parameter wie der Erfolgsrückruf. Wenn dies zu einem Fehler führt (einschließlich 3xx-Weiterleitung), übernehmen sie dieselben Parameter wie der `error` Callback.

## Ajax zum Senden eines Formulars verwenden

Manchmal haben Sie möglicherweise ein Formular und möchten es mit ajax absenden.

Angenommen, Sie haben diese einfache Form -

```
<form id="ajax_form" action="form_action.php">
  <label for="name">Name :</label>
  <input name="name" id="name" type="text" />
  <label for="name">Email :</label>
  <input name="email" id="email" type="text" />
  <input type="submit" value="Submit" />
</form>
```

Der folgende jQuery-Code kann verwendet werden (innerhalb eines `$(document).ready` Aufrufs):

```
$('#ajax_form').submit(function(event){
  event.preventDefault();
  var $form = $(this);

  $.ajax({
      type: 'POST',
      url: $form.attr('action'),
      data: $form.serialize(),
      success: function(data) {
        // Do something with the response
      },
      error: function(error) {
        // Do something with the error
      }
  });
});
```

**Erläuterung**

- `var $form = $(this)` - das Formular, das zur Wiederverwendung zwischengespeichert wird
- `$('#ajax_form').submit(function(event){` - Wenn das Formular mit der ID "ajax_form" gesendet wird, führen Sie diese Funktion aus und übergeben Sie das Ereignis als Parameter.
- `event.preventDefault();` - Verhindern Sie, dass das Formular normal übermittelt wird (Alternativ können Sie nach der `ajax({});` Anweisung `ajax({}); return false` , was die gleiche Wirkung hat).
- `url: $form.attr('action'),` - `url: $form.attr('action'),` den Wert des "action" -Attributs des Formulars ab und verwendet ihn für die "url" -Eigenschaft.
- `data: $form.serialize(),` - Konvertiert die Eingaben innerhalb des Formulars in eine Zeichenfolge, die zum Senden an den Server geeignet ist. In diesem Fall wird etwas wie "name=Bob&email=bob@bobsemailaddress.com" zurückgegeben.

## Senden von JSON-Daten

Mit jQuery ist die Handhabung von JSON- *Antworten* problemlos. Es ist jedoch etwas mehr Arbeit erforderlich, wenn eine bestimmte Anforderung das *Senden von* Daten im JSON-Format erfordert:

```
$.ajax("/json-consuming-route", {
     data: JSON.stringify({author: {name: "Bullwinkle J. Moose",
                                    email: "bullwinkle@example.com"} }),
     method: "POST",
     contentType: "application/json"
  });
```

Beachten Sie, dass wir den korrekten `contentType` für die Daten `contentType` die wir senden. Dies ist im Allgemeinen eine bewährte Methode, die möglicherweise von der API verlangt wird, für die Sie eine Veröffentlichung durchführen. Dies hat jedoch *auch* den Nebeneffekt, dass jQuery angewiesen wird, die Standardkonvertierung von `%20` in `+` nicht durchzuführen, was bei einem `contentType` wäre auf dem Standardwert von `application/x-www-form-urlencoded` . Wenn Sie aus irgendeinem Grund contentType auf den Standardwert setzen müssen, stellen Sie sicher, dass

`processData` auf false gesetzt ist, um dies zu verhindern.

Der Aufruf von `JSON.stringify` könnte hier vermieden werden, aber durch die Verwendung von `JSON.stringify` können die Daten in Form eines JavaScript-Objekts `JSON.stringify` werden (um peinliche JSON-Syntaxfehler zu vermeiden, z. B. das Angeben von Eigenschaftennamen).

## Alles in einem Beispielen

**Ajax Get:**

**Lösung 1:**

```
$.get('url.html', function(data){
    $('#update-box').html(data);
});
```

**Lösung 2:**

```
 $.ajax({
     type: 'GET',
     url: 'url.php',
 }).done(function(data){
     $('#update-box').html(data);
 }).fail(function(jqXHR, textStatus){
     alert('Error occured: ' + textStatus);
 });
```

**Ajax Load:** Eine weitere **Ajax-** Get-Methode, die zur Vereinfachung erstellt wurde

```
$('#update-box').load('url.html');
```

.load kann auch mit zusätzlichen Daten aufgerufen werden. Der Datenteil kann als String oder Objekt bereitgestellt werden.

```
$('#update-box').load('url.php', {data: "something"});
$('#update-box').load('url.php', "data=something");
```

Wenn .load mit einer Callback-Methode aufgerufen wird, wird die Anforderung an den Server als Post ausgegeben

```
$('#update-box').load('url.php', {data: "something"}, function(resolve){
    //do something
});
```

**Ajax Post:**

**Lösung 1:**

```
$.post('url.php',
    {date1Name: data1Value, date2Name: data2Value},  //data to be posted
```

```
    function(data){
        $('#update-box').html(data);
    }
);
```

### Lösung 2:

```
$.ajax({
    type: 'Post',
    url: 'url.php',
    data: {date1Name: data1Value, date2Name: data2Value}  //data to be posted
}).done(function(data){
    $('#update-box').html(data);
}).fail(function(jqXHR, textStatus){
    alert('Error occured: ' + textStatus);
});
```

### Ajax Post JSON:

```
var postData = {
    Name: name,
    Address: address,
    Phone: phone
};

$.ajax({
     type: "POST",
     url: "url.php",
     dataType: "json",
     data: JSON.stringfy(postData),
     success: function (data) {
         //here variable data is in JSON format
     }
 });
```

### Ajax Get JSON:

### Lösung 1:

```
$.getJSON('url.php', function(data){
    //here variable data is in JSON format
});
```

### Lösung 2:

```
 $.ajax({
      type: "Get",
      url: "url.php",
      dataType: "json",
      data: JSON.stringfy(postData),
      success: function (data) {
          //here variable data is in JSON format
      },
      error: function(jqXHR, textStatus){
          alert('Error occured: ' + textStatus);
      }
```

```
  });
```

### Ajax Abbruch eines Anrufs oder einer Anfrage

```
var xhr = $.ajax({
    type: "POST",
    url: "some.php",
    data: "name=John&location=Boston",
    success: function(msg){
       alert( "Data Saved: " + msg );
    }
});
```

// töte die Anfrage

```
xhr.abort()
```

### Ajax-Datei-Uploads

# 1. Ein einfaches vollständiges Beispiel

Wir können diesen Beispielcode verwenden, um die vom Benutzer ausgewählten Dateien jedes Mal hochzuladen, wenn eine neue Dateiauswahl getroffen wird.

```
<input type="file" id="file-input" multiple>
```

```
var files;
var fdata = new FormData();
$("#file-input").on("change", function (e) {
    files = this.files;

    $.each(files, function (i, file) {
        fdata.append("file" + i, file);
    });

    fdata.append("FullName", "John Doe");
    fdata.append("Gender", "Male");
    fdata.append("Age", "24");

    $.ajax({
        url: "/Test/Url",
        type: "post",
        data: fdata, //add the FormData object to the data parameter
        processData: false, //tell jquery not to process data
        contentType: false, //tell jquery not to set content-type
        success: function (response, status, jqxhr) {
            //handle success
        },
        error: function (jqxhr, status, errorMessage) {
            //handle error
        }
    });
```

```
});
```

Lassen Sie uns das nun aufschlüsseln und Teil für Teil untersuchen.

# 2. Arbeiten mit Dateieingaben

Dieses MDN-Dokument (Verwenden von Dateien aus Webanwendungen) enthält Informationen zu verschiedenen Methoden zum Umgang mit Dateieingaben. Einige dieser Methoden werden auch in diesem Beispiel verwendet.

Bevor wir Dateien hochladen können, müssen wir dem Benutzer zunächst die Möglichkeit geben, die Dateien auszuwählen, die er hochladen möchte. Zu diesem Zweck verwenden wir eine `file input` . Die `multiple` Eigenschaft ermöglicht das Auswählen mehrerer Dateien. Sie können sie entfernen, wenn Sie möchten, dass der Benutzer jeweils eine Datei auswählt.

```
<input type="file" id="file-input" multiple>
```

Wir werden das `change event` von input verwenden, um die Dateien zu erfassen.

```
var files;
$("#file-input").on("change", function(e){
    files = this.files;
});
```

Innerhalb der Handlerfunktion greifen wir über die Eigenschaft files unserer Eingabe auf die Dateien zu. Dies gibt uns eine FileList , ein Array-ähnliches Objekt.

# 3. Erstellen und Füllen der FormData

Um Dateien mit Ajax hochzuladen, verwenden wir FormData .

```
var fdata = new FormData();
```

Die Dateiliste, die wir im vorherigen Schritt erhalten haben, ist ein Array-ähnliches Objekt, das mit verschiedenen Methoden einschließlich für Schleife , für ... der Schleife und jQuery.each wiederholt werden kann . In diesem Beispiel bleiben wir bei der jQuery.

```
$.each(files, function(i, file) {
  //...
});
```

Wir werden die Append-Methode von FormData verwenden, um die Dateien unserem formdata-Objekt hinzuzufügen.

```
$.each(files, function(i, file) {
  fdata.append("file" + i, file);
```

```
});
```

Wir können auch andere Daten hinzufügen, die wir auf dieselbe Weise senden möchten. Angenommen, wir möchten einige persönliche Informationen, die wir vom Benutzer erhalten haben, zusammen mit den Dateien senden. Wir könnten diese Informationen in unser formdata-Objekt einfügen.

```
fdata.append("FullName", "John Doe");
fdata.append("Gender", "Male");
fdata.append("Age", "24");
//...
```

# 4. Senden der Dateien mit Ajax

```
$.ajax({
    url: "/Test/Url",
    type: "post",
    data: fdata, //add the FormData object to the data parameter
    processData: false, //tell jquery not to process data
    contentType: false, //tell jquery not to set content-type
    success: function (response, status, jqxhr) {
        //handle success
    },
    error: function (jqxhr, status, errorMessage) {
        //handle error
    }
});
```

Die Eigenschaften von `processData` und `contentType` setzen wir auf `false` . Dies geschieht, damit die Dateien an den Server gesendet und vom Server korrekt verarbeitet werden können.

Ajax online lesen: https://riptutorial.com/de/jquery/topic/316/ajax

# Kapitel 3: Anhängen

## Syntax

1. $ (Selektor) .append (Inhalt)
2. $ (content) .appendTo (Selektor)

## Parameter

| Parameter | Einzelheiten |
|---|---|
| Inhalt | Mögliche Typen: Element, HTML-String, Text, Array, Objekt oder sogar eine Funktion, die einen String zurückgibt. |

## Bemerkungen

- .append () & .after () kann möglicherweise Code ausführen. Dies kann durch das Einfügen von Skript-Tags oder die Verwendung von HTML-Attributen geschehen, die Code ausführen (z. B.). Verwenden Sie diese Methoden nicht zum Einfügen von Zeichenfolgen aus nicht vertrauenswürdigen Quellen wie URL-Abfrageparametern, Cookies oder Formulareingaben. Dadurch können Cross-Site-Scripting-Schwachstellen (XSS) entstehen. Entfernen Sie die Benutzereingaben, bevor Sie Inhalte zum Dokument hinzufügen.
- jQuery unterstützt SVG nicht offiziell. Verwenden von jQuery-Methoden für SVG Wenn Dokumente nicht explizit für diese Methode dokumentiert sind, kann dies zu unerwartetem Verhalten führen. Beispiele für Methoden, die SVG ab unterstützen jQuery 3.0 sind addClass und removeClass.

## Examples

### Ein Element an einen Container anhängen

**Lösung 1:**

```
$('#parent').append($('#child'));
```

**Lösung 2:**

```
$('#child').appendTo($('#parent'));
```

Beide Lösungen hängen das Element `#child` (das am Ende hinzugefügt wird) an das Element `#parent` .

Vor:

```
<div id="parent">
  <span>other content</span>
</div>
<div id="child">

</div>
```

Nach dem:

```
<div id="parent">
  <span>other content</span>
  <div id="child">

  </div>
</div>
```

---

**Hinweis:** Wenn Sie Inhalte anhängen, die bereits im Dokument vorhanden sind, wird dieser Inhalt aus dem ursprünglichen übergeordneten Container entfernt und an den neuen übergeordneten Container angehängt. Sie können also nicht `.append()` oder `.appendTo()` , um ein Element zu klonen. Wenn Sie einen Klon benötigen, verwenden Sie `.clone()` -> [ http://api.jquery.com/clone/([1]

### Effiziente fortlaufende Verwendung von .append ()

Beginnend:

# HTML

```
<table id='my-table' width='960' height='500'></table>
```

# JS

```
var data = [
    { type: "Name", content: "John Doe" },
    { type: "Birthdate", content: "01/01/1970" },
    { type: "Salary", content: "$40,000,000" },
    // ...300 more rows...
    { type: "Favorite Flavour", content: "Sour" }
];
```

---

**Anhängen innerhalb einer Schleife**

Sie haben gerade eine große Anzahl von Daten erhalten. Jetzt ist es an der Zeit, es durchzublättern und auf der Seite darzustellen.

Ihr erster Gedanke könnte sein, so etwas zu tun:

```
var i;                          // <- the current item number
var count = data.length;        // <- the total
var row;                        // <- for holding a reference to our row object

// Loop over the array
for ( i = 0; i < count; ++i ) {
    row = data[ i ];

    // Put the whole row into your table
    $('#my-table').append(
        $('<tr></tr>').append(
            $('<td></td>').html(row.type),
            $('<td></td>').html(row.content)
        )
    );
}
```

Dies ist *vollkommen gültig* und wird genau das wiedergeben, was Sie erwarten würden, aber ...

# Mach das nicht.

Erinnern Sie sich an diese **300 Datenzeilen** ?

**Jeder** wird den Browser zwingen, die Breite, Höhe und Positionierungswerte jedes Elements zusammen mit allen anderen Stilen neu zu berechnen - es sei denn, sie sind durch eine Layoutbegrenzung getrennt, die leider für dieses Beispiel (da sie Nachkommen eines `<table>` - Elements sind ), Sie können nicht.

Bei geringen Beträgen und wenigen Kolumnen ist dieser Leistungsnachteil sicherlich vernachlässigbar. Aber wir wollen, dass jede Millisekunde zählt.

---

**Bessere möglichkeiten**

## 1. Fügen Sie zu einem separaten Array hinzu, und fügen Sie die Schleife nach Abschluss an

```
/**
 * Repeated DOM traversal (following the tree of elements down until you reach
 * what you're looking for - like our <table>) should also be avoided wherever possible.
 */

// Keep the table cached in a variable then use it until you think it's been removed
var $myTable = $('#my-table');

// To hold our new <tr> jQuery objects
var rowElements = [];

var count = data.length;
var i;
```

```
var row;

// Loop over the array
for ( i = 0; i < count; ++i ) {
    rowElements.push(
        $('<tr></tr>').append(
            $('<td></td>').html(row.type),
            $('<td></td>').html(row.content)
        )
    );
}

// Finally, insert ALL rows at once
$myTable.append(rowElements);
```

Von diesen Optionen stützt sich diese Option am meisten auf jQuery.

---

## 2. Verwenden moderner Array. * -Methoden

```
var $myTable = $('#my-table');

// Looping with the .map() method
//  - This will give us a brand new array based on the result of our callback function
var rowElements = data.map(function ( row ) {

    // Create a row
    var $row = $('<tr></tr>');

    // Create the columns
    var $type = $('<td></td>').html(row.type);
    var $content = $('<td></td>').html(row.content);

    // Add the columns to the row
    $row.append($type, $content);

    // Add to the newly-generated array
    return $row;
});

// Finally, put ALL of the rows into your table
$myTable.append(rowElements);
```

Funktionell gleichwertig wie das davor, nur lesbarer.

---

## 3. Verwenden von HTML-Zeichenfolgen (anstelle der integrierten Methoden von jQuery)

```
// ...
var rowElements = data.map(function ( row ) {
    var rowHTML = '<tr><td>';
    rowHTML += row.type;
    rowHTML += '</td><td>';
```

```
    rowHTML += row.content;
    rowHTML += '</td></tr>';
    return rowHTML;
});

// Using .join('') here combines all the separate strings into one
$myTable.append(rowElements.join(''));
```

*Perfekt* aber wieder, **nicht empfehlenswert** . Dies zwingt jQuery, eine sehr große Textmenge auf einmal zu analysieren und ist nicht erforderlich. jQuery ist sehr gut bei der richtigen Verwendung.

---

## 4. Elemente manuell erstellen, an Fragment des Dokuments anhängen

```
var $myTable = $(document.getElementById('my-table'));

/**
 * Create a document fragment to hold our columns
 * - after appending this to each row, it empties itself
 *   so we can re-use it in the next iteration.
 */
var colFragment = document.createDocumentFragment();

/**
 * Loop over the array using .reduce() this time.
 * We get a nice, tidy output without any side-effects.
 *  - In this example, the result will be a
 *    document fragment holding all the <tr> elements.
 */
var rowFragment = data.reduce(function ( fragment, row ) {

    // Create a row
    var rowEl = document.createElement('tr');

    // Create the columns and the inner text nodes
    var typeEl = document.createElement('td');
    var typeText = document.createTextNode(row.type);
    typeEl.appendChild(typeText);

    var contentEl = document.createElement('td');
    var contentText = document.createTextNode(row.content);
    contentEl.appendChild(contentText);

    // Add the columns to the column fragment
    // - this would be useful if columns were iterated over separately
    //   but in this example it's just for show and tell.
    colFragment.appendChild(typeEl);
    colFragment.appendChild(contentEl);

    rowEl.appendChild(colFragment);

    // Add rowEl to fragment - this acts as a temporary buffer to
    // accumulate multiple DOM nodes before bulk insertion
    fragment.appendChild(rowEl);

    return fragment;
```

```
}, document.createDocumentFragment());

// Now dump the whole fragment into your table
$myTable.append(rowFragment);
```

**Mein persönlicher Favorit** . Dies veranschaulicht eine allgemeine Vorstellung davon, was jQuery auf einer niedrigeren Ebene tut.

---

# Tauchen Sie tiefer ein

- jQuery Source Viewer
- Array.prototype.join ()
- Array.prototype.map ()
- Array.prototype.reduce ()
- document.createDocumentFragment ()
- document.createTextNode ()
- Google Web Fundamentals - Leistung

## jQuery anhängen

HTML

```
<p>This is a nice </p>
<p>I like </p>

<ul>
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
</ul>

<button id="btn-1">Append text</button>
<button id="btn-2">Append list item</button>
```

Skript

```
$("#btn-1").click(function(){
    $("p").append(" <b>Book</b>.");
});
$("#btn-2").click(function(){
    $("ul").append("<li>Appended list item</li>");
 });
});
```

Anhängen online lesen: https://riptutorial.com/de/jquery/topic/1910/anhangen

# Kapitel 4: Attribute

## Bemerkungen

Die jQuery-Funktion `.attr()` den Wert eines Attributs für das **erste** Element in der Gruppe der übereinstimmenden Elemente ab oder `.attr()` ein oder mehrere Attribute für **jedes** übereinstimmende Element fest.

Es sei darauf hingewiesen, dass der Wert eines Attributs nur vom ersten Element `$("input").attr("type");` , das mit dem Selektor übereinstimmt (dh `$("input").attr("type");` würde nur den Typ der ersten Eingabe erhalten , wenn es mehr als eine gibt)

Beim Festlegen eines Attributs wird es jedoch auf alle übereinstimmenden Elemente angewendet.

## Examples

### Ruft den Attributwert eines HTML-Elements ab

Wenn ein einzelner Parameter an die `.attr()` Funktion übergeben wird, gibt er den Wert des übergebenen Attributs für das ausgewählte Element zurück.

Syntax:

```
$([selector]).attr([attribute name]);
```

Beispiel:

HTML:

```
<a href="/home">Home</a>
```

jQuery:

```
$('a').attr('href');
```

**Abrufen** von **`data`**

jQuery bietet die Funktion `.data()` , um mit `.data()` umzugehen. `.data` Funktion `.data` gibt den Wert des `.data` für das ausgewählte Element zurück.

Syntax:

```
$([selector]).data([attribute name]);
```

Beispiel:

Html:

```
<article data-column="3"></article>
```

jQuery:

```
$("article").data("column")
```

**Hinweis:**

> Mit der data () -Methode von jQuery können Sie auf data- * -Attribute zugreifen, ABER, sie blockiert den Fall des Attributnamens. Referenz

## Einstellungswert des HTML-Attributs

Wenn Sie einem Element ein Attribut hinzufügen möchten, können Sie die Funktion `attr(attributeName, attributeValue)` verwenden. Zum Beispiel:

```
$('a').attr('title', 'Click me');
```

In diesem Beispiel wird der Mouseover-Text `"Click me"` zu allen Links auf der Seite hinzugefügt.

Dieselbe Funktion wird zum Ändern von Attributwerten verwendet.

## Attribut entfernen

Um ein Attribut aus einem Element zu entfernen, können Sie die Funktion `.removeAttr(attributeName)` . Zum Beispiel:

```
$('#home').removeAttr('title');
```

Dadurch wird das `title` Attribut aus dem Element mit der ID `home` .

## Differenz zwischen attr () und prop ()

`attr()` das HTML-Attribut mit den DOM-Funktionen `getAttribute()` und `setAttribute()` . `prop()` wird die DOM-Eigenschaft festgelegt, ohne das Attribut zu ändern. In vielen Fällen sind die beiden austauschbar, gelegentlich wird jedoch eine über der anderen benötigt.

So setzen Sie ein Kontrollkästchen als aktiviert:

```
$('#tosAccept').prop('checked', true); // using attr() won't work properly here
```

Um eine Eigenschaft zu entfernen, können Sie die `removeProp()` -Methode verwenden. In `removeAttr()` Weise entfernt `removeAttr()` Attribute.

Attribute online lesen: https://riptutorial.com/de/jquery/topic/4429/attribute

# Kapitel 5: Breite und Höhe eines Elements ermitteln und einstellen

## Examples

### Breite und Höhe abrufen und einstellen (Rand ignorieren)

Breite und Höhe erhalten:

```
var width = $('#target-element').width();
var height = $('#target-element').height();
```

Breite und Höhe einstellen:

```
$('#target-element').width(50);
$('#target-element').height(100);
```

### InnerWidth und innerHeight abrufen und einstellen

Breite und Höhe erhalten:

```
var width = $('#target-element').innerWidth();
var height = $('#target-element').innerHeight();
```

Breite und Höhe einstellen:

```
$('#target-element').innerWidth(50);
$('#target-element').innerHeight(100);
```

### OuterWidth und outerHeight abrufen und einstellen

Breite und Höhe (ohne Rand) abrufen:

```
var width = $('#target-element').outerWidth();
var height = $('#target-element').outerHeight();
```

Breite und Höhe (einschließlich Rand) erhalten:

```
var width = $('#target-element').outerWidth(true);
var height = $('#target-element').outerHeight(true);
```

Breite und Höhe einstellen:

```
$('#target-element').outerWidth(50);
$('#target-element').outerHeight(100);
```

Breite und Höhe eines Elements ermitteln und einstellen online lesen: https://riptutorial.com/de/jquery/topic/2167/breite-und-hohe-eines-elements-ermitteln-und-einstellen

# Kapitel 6: CSS-Manipulation

## Syntax

- .css (cssProperty) // Liefert den gerenderten CSS-Eigenschaftswert
- .css ([cssProperty, ...]) // Werte aus einem Array von cssProperties abrufen
- .css (cssProperty, value) // Wert setzen
- .css ({cssProperty: value, ...}) // Eigenschaften und Werte setzen
- .css (cssProperty, function) // Machen Sie die cssProperty für eine Rückruffunktion verfügbar

## Bemerkungen

### Gerenderte Werte

Wenn eine responsive Unit verwendet wird (wie `"auto"` , `"%"` , `"vw"` usw.), gibt `.css()` den tatsächlich gerenderten Wert in `px`

```
.myElement{ width: 20%; }
```

```
var width = $(".myElement").css("width"); // "123px"
```

### Eigenschaften und Werte formatieren

**Eigenschaften** können mit der **Standard-CSS-Formatierung als String** oder mit **camelCase definiert werden**

```
"margin-bottom"
marginBottom
```

**Die Werte** sollten in String angegeben werden. Numerische Werte werden von jQuery intern als `px` Einheiten behandelt

```
.css(fontSize: "1em")
.css(fontSize: "16px")
.css(fontSize: 16)      // px will be used
```

### Ab jQuery 3 vermeiden Sie die Verwendung von `.show()` und `.hide()`

Gemäß diesem Blogeintrag in jQuery sollten Sie aufgrund von Overhead- und Leistungsproblemen nicht länger `.show()` oder `.hide()` .

> Wenn in einem Stylesheet Elemente festgelegt sind, die `display: none` , wird die Methode `.show()` nicht mehr überschrieben. Die wichtigste Regel für den Wechsel zu jQuery 3.0 lautet daher: Verwenden Sie kein Stylesheet, um den Standard für die `display: none`

`.show()` `display: none` und versuchen Sie dann, `.show()` - oder eine beliebige Methode, die Elemente wie `.slideDown()` und `.fadeIn()` - um es sichtbar zu machen. Wenn ein Element standardmäßig ausgeblendet werden soll, fügen Sie dem Element am besten einen Klassennamen wie „hidden“ hinzu und definieren Sie die `display: none` Klasse `display: none` in einem Stylesheet. Anschließend können Sie diese Klasse mit den `.addClass()` und `.removeClass()` von jQuery hinzufügen oder entfernen, um die Sichtbarkeit zu steuern. Alternativ können Sie einen `.ready()` Handler-Aufruf `.hide()` für die Elemente aufrufen, bevor sie auf der Seite angezeigt werden. Wenn Sie den Stylesheet-Standard wirklich beibehalten müssen, können Sie `.css("display", "block")` (oder den entsprechenden Anzeigewert) verwenden, um das Stylesheet zu überschreiben.

# Examples

## Legen Sie die CSS-Eigenschaft fest

Nur einen Stil festlegen:

```
$('#target-element').css('color', '#000000');
```

Mehrere Stile gleichzeitig einstellen:

```
$('#target-element').css({
    'color': '#000000',
    'font-size': '12pt',
    'float': 'left',
});
```

## CSS-Eigenschaft abrufen

Um die CSS-Eigenschaft eines Elements abzurufen, können Sie die Methode `.css(propertyName)` verwenden:

```
var color    = $('#element').css('color');
var fontSize = $('#element').css('font-size');
```

## Numerische Eigenschaften für Inkrement / Dekrement

Numerische CSS-Eigenschaften können mit der Syntax `+=` und `-=` mit der Methode `.css()` erhöht bzw. dekrementiert werden:

```
  // Increment using the += syntax
  $("#target-element").css("font-size", "+=10");

  // You can also specify the unit to increment by
  $("#target-element").css("width", "+=100pt");
  $("#target-element").css("top", "+=30px");
  $("#target-element").css("left", "+=3em");
```

```
// Decrementing is done by using the -= syntax
$("#target-element").css("height", "-=50pt");
```

### CSS - Getter und Setter

## CSS-Getter

Die `.css()` **Getter** - Funktion kann auf der Seite wie folgt zu jedem DOM - Elemente angewandt werden:

```
// Rendered width in px as a string. ex: `150px`
// Notice the `as a string` designation - if you require a true integer,
// refer to `$.width()` method
$("body").css("width");
```

Diese Zeile gibt die **berechnete Breite** des angegebenen Elements zurück. Jede CSS-Eigenschaft, die Sie in Klammern angeben, liefert den Wert der Eigenschaft für dieses `$("selector")` DOM-Element, wenn Sie nach einem CSS-Attribut fragen, das Sie nicht enthalten wird als Antwort `undefined` .

Sie können den **CSS-Getter auch** mit einem Array von Attributen aufrufen:

```
$("body").css(["animation","width"]);
```

Dadurch wird ein Objekt aller Attribute mit ihren Werten zurückgegeben:

```
Object {animation: "none 0s ease 0s 1 normal none running", width: "529px"}
```

## CSS Setter

Die **Setter-** Methode `.css()` kann auch auf jedes DOM-Element auf der Seite angewendet werden.

```
$("selector").css("width", 500);
```

Diese Anweisung setzt die `width` von `$("selector")` auf `500px` und gibt das jQuery-Objekt zurück, damit Sie mehr Methoden an den angegebenen Selector ketten können.

Der `.css()` **Setter** kann auch verwendet werden, indem ein Object mit CSS-Eigenschaften und Werten übergeben wird, wie:

```
$("body").css({"height": "100px", width:100, "padding-top":40, paddingBottom:"2em"});
```

Alle Änderungen der Setter gemacht angehängt werden , auf das DOM - Element - `style` Eigenschaft somit die Elemente Stile beeinflussen (es sei denn , dass Stil - Wert bereits definiert als `!important` woanders in Stile).

CSS-Manipulation online lesen: https://riptutorial.com/de/jquery/topic/2732/css-manipulation

# Kapitel 7: dokumentbereites Ereignis

## Examples

### Was ist dokumentenbereit und wie soll ich es verwenden?

jQuery-Code wird häufig in `jQuery(function($) { ... });` so dass es erst ausgeführt wird, nachdem das Laden des DOM abgeschlossen ist.

```
<script type="text/javascript">
  jQuery(function($) {
    // this will set the div's text to "Hello".
    $("#myDiv").text("Hello");
  });
</script>

<div id="myDiv">Text</div>
```

Dies ist wichtig, da jQuery (und JavaScript im Allgemeinen) kein DOM-Element auswählen können, das nicht für die Seite gerendert wurde.

```
<script type="text/javascript">
  // no element with id="myDiv" exists at this point, so $("#myDiv") is an
  // empty selection, and this will have no effect
  $("#myDiv").text("Hello");
</script>

<div id="myDiv">Text</div>
```

Beachten Sie, dass Sie den jQuery-Namespace `.ready()` indem Sie einen benutzerdefinierten Handler an die `.ready()` -Methode übergeben. Dies ist nützlich, wenn eine andere JS-Bibliothek denselben verkürzten `$` -Alias verwendet wie *jQuery* , wodurch ein Konflikt entsteht. Um diesen Konflikt zu vermeiden, müssen Sie `$.noConflict();` - Dies zwingt Sie, nur den Standard- *jQuery*-Namespace zu verwenden (anstelle des kurzen Alias " `$` ).
Durch Übergeben eines benutzerdefinierten `.ready()` an den `.ready()` Handler können Sie den Aliasnamen für die Verwendung von *jQuery* auswählen.

```
$.noConflict();

jQuery( document ).ready(function( $ ) {
   // Here we can use '$' as jQuery alias without it conflicting with other
   // libraries that use the same namespace
   $('body').append('<div>Hello</div>')
});

jQuery( document ).ready(function( jq ) {
   // Here we use a custom jQuery alias 'jq'
   jq('body').append('<div>Hello</div>')
});
```

Anstatt Ihren jQuery-Code einfach am unteren Rand der Seite zu platzieren, stellt die Funktion `$(document).ready` sicher, dass alle HTML-Elemente gerendert werden und das gesamte Document Object Model (DOM) für die Ausführung von JavaScript-Code bereit ist.

### jQuery 2.2.3 und früher

Diese sind alle gleichwertig. Der Code in den Blöcken wird ausgeführt, wenn das Dokument fertig ist:

```
$(function() {
  // code
});

$().ready(function() {
  // code
});

$(document).ready(function() {
  // code
});
```

Da diese gleichwertig sind, wird zuerst das empfohlene Formular verwendet. Nachfolgend wird eine Version mit `jQuery` Schlüsselwort anstelle von `$` die die gleichen Ergebnisse liefern:

```
jQuery(function() {
  // code
});
```

### jQuery 3.0

## Notation

Ab jQuery 3.0 wird nur dieses Formular empfohlen:

```
jQuery(function($) {
  // Run when document is ready
  // $ (first argument) will be internal reference to jQuery
  // Never rely on $ being a reference to jQuery in the global namespace
});
```

Alle anderen dokumentbereiten Handler werden in jQuery 3.0 nicht mehr unterstützt.

## Asynchron

Ab jQuery 3.0 wird der Ready-Handler immer asynchron aufgerufen . Das bedeutet, dass im folgenden Code immer das Protokoll "Outside Handler" angezeigt wird, unabhängig davon, ob das Dokument zum Zeitpunkt der Ausführung bereit war.

```
$(function() {
```

```
  console.log("inside handler");
});
console.log("outside handler");
```

> Äußerer Handler
> im Handler

## Unterschied zwischen $ (document) .ready () und $ (window) .load ()

`$(window).load()` wurde **in jQuery Version 1.8 nicht mehr unterstützt (und vollständig von jQuery 3.0 entfernt )** und sollte daher nicht mehr verwendet werden. Die Gründe für die Abwertung werden auf der jQuery-Seite zu diesem Ereignis angegeben

> Vorbehalte des Ladeereignisses bei Verwendung mit Bildern
>
> Eine häufige Herausforderung, die Entwickler mithilfe der `.load()` Verknüpfung lösen `.load()` ist das Ausführen einer Funktion, wenn ein Bild (oder eine Bildersammlung) vollständig geladen wurde. Es gibt einige bekannte Vorbehalte, die beachtet werden sollten. Diese sind:
>
> - Es funktioniert nicht konsistent oder zuverlässig über Browser
> - Es wird nicht richtig in WebKit ausgelöst, wenn die Image- `src` auf dieselbe `src` wie zuvor eingestellt ist
> - Der DOM-Baum wird nicht richtig angezeigt
> - Kann für Bilder, die sich bereits im Cache des Browsers befinden, nicht mehr ausgelöst werden

Wenn Sie noch `load()` möchten, wird dies nachfolgend dokumentiert:

---

`$(document).ready()` wartet, bis das vollständige DOM verfügbar ist - alle Elemente im HTML-Code wurden analysiert und befinden sich im Dokument. Ressourcen wie Bilder wurden an diesem Punkt jedoch möglicherweise nicht vollständig geladen. Wenn es wichtig ist, zu warten, bis alle Ressourcen geladen sind, `$(window).load()` **und Sie sich der erheblichen Einschränkungen dieses Ereignisses bewusst** sind, können Sie stattdessen Folgendes verwenden:

```
$(document).ready(function() {
  console.log($("#my_large_image").height()); // may be 0 because the image isn't available
});

$(window).load(function() {
  console.log($("#my_large_image").height()); // will be correct
});
```

## Ereignisse anfügen und das DOM in ready () bearbeiten

Beispielanwendungen von `$(document).ready()` :

1. **Event-Handler anhängen**
   Fügen Sie jQuery-Ereignishandler hinzu

```
$(document).ready(function() {
  $("button").click(function() {
    // Code for the click function
  });
});
```

2. **Führen Sie jQuery-Code aus, nachdem die Seitenstruktur erstellt wurde**

```
jQuery(function($) {
// set the value of an element.
   $("#myElement").val("Hello");
});
```

3. **Bearbeiten Sie die geladene DOM-Struktur**
   Zum Beispiel: Ein `div` ausblenden, wenn die Seite zum ersten Mal `div` wird, und es beim Klickereignis einer Schaltfläche anzeigen

```
$(document).ready(function() {
  $("#toggleDiv").hide();
  $("button").click(function() {
    $("#toggleDiv").show();
  });
});
```

## Unterschied zwischen jQuery (fn) und vorherigem Ausführen Ihres Codes

Die Verwendung des dokumentbereiten Ereignisses kann kleine Leistungsnachteile mit einer verzögerten Ausführung von bis zu ~ 300 ms haben. Dasselbe Verhalten kann manchmal durch die Ausführung von Code unmittelbar vor dem schließenden `</body>` -Tag erreicht werden:

```
<body>
  <span id="greeting"></span> world!
  <script>
    $("#greeting").text("Hello");
  </script>
</body>
```

führt zu einem ähnlichen Verhalten, führt jedoch früher aus, als es nicht auf den Ereignisauslöser für das Dokument bereit wartet, wie in:

```
<head>
  <script>
    jQuery(function($) {
      $("#greeting").text("Hello");
    });
  </script>
</head>
<body>
  <span id="greeting"></span> world!
</body>
```

Betonen Sie die Tatsache, dass das erste Beispiel auf Ihrem Wissen über Ihre Seite und der

Platzierung des Skripts unmittelbar vor dem schließenden `</body>` -Tag und insbesondere nach dem `span` Tag beruht.

dokumentbereites Ereignis online lesen:
https://riptutorial.com/de/jquery/topic/500/dokumentbereites-ereignis

# Kapitel 8: DOM Traversing

## Examples

### Wählen Sie die untergeordneten Elemente des Elements aus

Um die untergeordneten Elemente eines Elements auszuwählen, können Sie die `children()` -Methode verwenden.

```
<div class="parent">
    <h2>A headline</h2>
    <p>Lorem ipsum dolor sit amet...</p>
    <p>Praesent quis dolor turpis...</p>
</div>
```

Ändern Sie die Farbe *aller* `.parent` Elemente des Elements `.parent` :

```
$('.parent').children().css("color", "green");
```

Die Methode akzeptiert ein optionales `selector` Argument, mit dem die zurückgegebenen Elemente gefiltert werden können.

```
// Only get "p" children
$('.parent').children("p").css("color", "green");
```

### Iteration über eine Liste von jQuery-Elementen

Wenn Sie die Liste der jQuery-Elemente durchlaufen müssen.

Betrachten Sie diese DOM-Struktur:

```
<div class="container">
    <div class="red one">RED 1 Info</div>
    <div class="red two">RED 2 Info</div>
    <div class="red three">RED 3 Info</div>
</div>
```

So drucken Sie den in allen `div` Elementen vorhandenen Text mit einer `red` Klasse:

```
$(".red").each(function(key, ele){
    var text = $(ele).text();
    console.log(text);
});
```

*Tipp: Der* `key` ist der Index des `div.red` -Elements, das wir gerade innerhalb seines übergeordneten Elements `div.red` . `ele` ist das HTML-Element, so dass wir daraus ein jQuery-Objekt mit `$()` oder `jQuery()` erstellen können: `$(ele)` . Danach können wir jede jQuery-Methode für das Objekt

aufrufen, wie `css()` oder `hide()` usw. In diesem Beispiel ziehen wir nur den Text des Objekts.

## Geschwister auswählen

Zum Auswählen von Geschwistern eines Elements können Sie die `.siblings()` -Methode verwenden.

Ein typisches Beispiel, in dem Sie die Geschwister eines Elements ändern möchten, befindet sich in einem Menü:

```
<ul class="menu">
    <li class="selected">Home</li>
    <li>Blog</li>
    <li>About</li>
</ul>
```

Wenn der Benutzer auf einem Menüpunkt klickt die `selected` sollte Klasse angeklickten Element hinzugefügt werden und von seinen *Geschwistern* entfernt:

```
$(".menu").on("click", "li", function () {
    $(this).addClass("selected");
    $(this).siblings().removeClass("selected");
});
```

Die Methode verwendet ein optionales `selector` , das verwendet werden kann, wenn Sie die Arten von Geschwistern einschränken möchten, die Sie auswählen möchten:

```
$(this).siblings("li").removeClass("selected");
```

## nächste () Methode

Gibt das erste Element zurück, das mit dem Selektor übereinstimmt, beginnend am Element und durch den DOM-Baum.

HTML

```
<div id="abc" class="row">
    <div id="xyz" class="row">
    </div>
    <p id="origin">
      Hello
    </p>
</div>
```

jQuery

```
var target = $('#origin').closest('.row');
console.log("Closest row:", target.attr('id') );

var target2 = $('#origin').closest('p');
console.log("Closest p:", target2.attr('id') );
```

AUSGABE

```
"Closest row: abc"
"Closest p: origin"
```

**first () -Methode:** Die erste Methode gibt das erste Element aus der übereinstimmenden Elementmenge zurück.

HTML

```
<div class='.firstExample'>
   <p>This is first paragraph in a div.</p>
   <p>This is second paragraph in a div.</p>
   <p>This is third paragraph in a div.</p>
   <p>This is fourth paragraph in a div.</p>
   <p>This is fifth paragraph in a div.</p>
</div>
```

JQuery

```
var firstParagraph = $("div p").first();
console.log("First paragraph:", firstParagraph.text());
```

Ausgabe:

```
First paragraph: This is first paragraph in a div.
```

## Holen Sie sich das nächste Element

Um das nächste Element zu erhalten, können Sie die `.next()` -Methode verwenden.

```
<ul>
    <li>Mark</li>
    <li class="anna">Anna</li>
    <li>Paul</li>
</ul>
```

Wenn Sie auf dem Element "Anna" stehen und das nächste Element "Paul" erhalten `.next()` , können Sie dies mit der `.next()` -Methode tun.

```
// "Paul" now has green text
$(".anna").next().css("color", "green");
```

Die Methode verwendet ein optionales `selector` , das verwendet werden kann, wenn das nächste Element eine bestimmte Art von Element sein muss.

```
// Next element is a "li", "Paul" now has green text
$(".anna").next("li").css("color", "green");
```

Wenn das nächste Element nicht vom `selector` ist, wird eine leere Menge zurückgegeben, und die

Änderungen führen zu nichts.

```
// Next element is not a ".mark", nothing will be done in this case
$(".anna").next(".mark").css("color", "green");
```

### Vorheriges Element abrufen

Um das vorherige Element zu erhalten, können Sie die `.prev()` -Methode verwenden.

```
<ul>
    <li>Mark</li>
    <li class="anna">Anna</li>
    <li>Paul</li>
</ul>
```

Wenn Sie auf dem Element "Anna" stehen und das vorherige Element "Mark" erhalten `.prev()` , können Sie dies mit der Methode `.prev()` tun.

```
// "Mark" now has green text
$(".anna").prev().css("color", "green");
```

Die Methode verwendet ein optionales `selector` , das verwendet werden kann, wenn das vorherige Element eine bestimmte Art von Element sein muss.

```
// Previous element is a "li", "Mark" now has green text
$(".anna").prev("li").css("color", "green");
```

Wenn das vorherige Element nicht vom `selector` ist, wird eine leere Menge zurückgegeben, und die Änderungen führen zu nichts.

```
// Previous element is not a ".paul", nothing will be done in this case
$(".anna").prev(".paul").css("color", "green");
```

### Auswahl filtern

Um eine Auswahl zu filtern, können Sie die `.filter()` -Methode verwenden.

Die Methode wird bei einer Auswahl aufgerufen und gibt eine neue Auswahl zurück. Wenn der Filter mit einem Element übereinstimmt, wird es der zurückgegebenen Auswahl hinzugefügt. Andernfalls wird es ignoriert. Wenn kein Element übereinstimmt, wird eine leere Auswahl zurückgegeben.

## Das HTML

Dies ist der HTML-Code, den wir verwenden werden.

```
<ul>
    <li class="zero">Zero</li>
```

```
    <li class="one">One</li>
    <li class="two">Two</li>
    <li class="three">Three</li>
</ul>
```

## Wähler

Das Filtern mit Selektoren ist eine der einfacheren Möglichkeiten, eine Auswahl zu filtern.

```
$("li").filter(":even").css("color", "green"); // Color even elements green
$("li").filter(".one").css("font-weight", "bold"); // Make ".one" bold
```

## Funktion

Das Filtern einer Auswahl mithilfe einer Funktion ist hilfreich, wenn die Verwendung von Selektoren nicht möglich ist.

Die Funktion wird für jedes Element in der Auswahl aufgerufen. Wenn ein `true` Wert zurückgegeben wird, wird das Element der zurückgegebenen Auswahl hinzugefügt.

```
var selection = $("li").filter(function (index, element) {
    // "index" is the position of the element
    // "element" is the same as "this"
    return $(this).hasClass("two");
});
selection.css("color", "green"); // ".two" will be colored green
```

## Elemente

Sie können nach DOM-Elementen filtern. Wenn sich die DOM-Elemente in der Auswahl befinden, werden sie in die zurückgegebene Auswahl aufgenommen.

```
var three = document.getElementsByClassName("three");
$("li").filter(three).css("color", "green");
```

## Auswahl

Sie können eine Auswahl auch nach einer anderen Auswahl filtern. Wenn sich ein Element in beiden Auswahlen befindet, wird es in die zurückgegebene Auswahl aufgenommen.

```
var elems = $(".one, .three");
$("li").filter(elems).css("color", "green");
```

### find () -Methode

Mit der .find () -Methode können wir die Nachkommen dieser Elemente in der DOM-Struktur durchsuchen und aus den übereinstimmenden Elementen ein neues jQuery-Objekt erstellen.

HTML

```
<div class="parent">
   <div class="children" name="first">
       <ul>
           <li>A1</li>
           <li>A2</li>
           <li>A3</li>
       </ul>
   </div>
   <div class="children" name="second">
       <ul>
           <li>B1</li>
           <li>B2</li>
           <li>B3</li>
       </ul>
   </div>
  </div>
```

jQuery

```
$('.parent').find('.children[name="second"] ul li').css('font-weight','bold');
```

Ausgabe

- A1
- A2
- A3

- **B1**
- **B2**
- **B3**

DOM Traversing online lesen: https://riptutorial.com/de/jquery/topic/1189/dom-traversing

# Kapitel 9: DOM-Manipulation

## Examples

### Erstellen von DOM-Elementen

Die `jQuery` Funktion (normalerweise als `$` ) kann sowohl zum Auswählen von Elementen als auch zum Erstellen neuer Elemente verwendet werden.

```
var myLink = $('<a href="http://stackexchange.com"></a>');
```

Sie können optional ein zweites Argument mit Elementattributen übergeben:

```
var myLink = $('<a>', { 'href': 'http://stackexchange.com' });
```

`'<a>'` -> Das erste Argument gibt den Typ des DOM-Elements an, das Sie erstellen möchten. In diesem Beispiel handelt es sich um einen Anker , es könnte sich jedoch um alles auf dieser Liste handeln . Siehe die Spezifikation für eine Referenz des `a` Elements.

`{ 'href': 'http://stackexchange.com' }` -> Das zweite Argument ist ein JavaScript-Objekt mit Attributnamen / Wert-Paaren.

Die 'name': 'value' -Paare werden zwischen den `< >` des ersten Arguments `<a name:value>` , zum Beispiel `<a name:value>` wäre in unserem Beispiel `<a href="http://stackexchange.com"></a>`

### Elementklassen bearbeiten

Angenommen, die Seite enthält ein HTML-Element wie:

```
<p class="small-paragraph">
  This is a small <a href="https://en.wikipedia.org/wiki/Paragraph">paragraph</a>
  with a <a class="trusted" href="http://stackexchange.com">link</a> inside.
</p>
```

jQuery bietet nützliche Funktionen zum Bearbeiten von DOM-Klassen, insbesondere `hasClass()` , `addClass()` , `removeClass()` und `toggleClass()` . Diese Funktionen ändern direkt das `class` der übereinstimmenden Elemente.

```
$('p').hasClass('small-paragraph'); // true
$('p').hasClass('large-paragraph'); // false

// Add a class to all links within paragraphs
$('p a').addClass('untrusted-link-in-paragraph');

// Remove the class from a.trusted
$('a.trusted.untrusted-link-in-paragraph')
.removeClass('untrusted-link-in-paragraph')
.addClass('trusted-link-in-paragraph');
```

### Klasse umschalten

Mit dem Beispielmarkup können wir eine Klasse mit unserer ersten `.toggleClass()` hinzufügen:

```
$(".small-paragraph").toggleClass("pretty");
```

Jetzt würde dies `true` : `$(".small-paragraph").hasClass("pretty")`

`toggleClass` bietet den gleichen Effekt mit weniger Code als:

```
if($(".small-paragraph").hasClass("pretty")){
   $(".small-paragraph").removeClass("pretty");}
else {
   $(".small-paragraph").addClass("pretty"); }
```

zwei Klassen umschalten:

```
$(".small-paragraph").toggleClass("pretty cool");
```

Boolean zum Hinzufügen / Entfernen von Klassen:

```
$(".small-paragraph").toggleClass("pretty",true); //cannot be truthy/falsey

$(".small-paragraph").toggleClass("pretty",false);
```

Funktion für Klassenumschaltung (siehe Beispiel weiter unten, um ein Problem zu vermeiden)

```
$( "div.surface" ).toggleClass(function() {
  if ( $( this ).parent().is( ".water" ) ) {
    return "wet";
  } else {
    return "dry";
  }
});
```

In Beispielen verwendet:

```
// functions to use in examples
function stringContains(myString, mySubString) {
  return myString.indexOf(mySubString) !== -1;
}
function isOdd(num) { return num % 2;}
var showClass = true; //we want to add the class
```

Beispiele:

Verwenden Sie den Elementindex, um die Klassen ungerade / gerade umzuschalten

```
$( "div.gridrow" ).toggleClass(function(index,oldClasses, false), showClass ) {
  showClass
  if ( isOdd(index) ) {
    return "wet";
```

```
  } else {
    return "dry";
  }
});
```

### Komplexeres `toggleClass` Beispiel bei einer einfachen Rastermarkierung

```
<div class="grid">
  <div class="gridrow">row</div>
  <div class="gridrow">row</div>
  <div class="gridrow">row</div>
  <div class="gridrow">row</div>
  <div class="gridrow">row</div>
  <div class="gridrow gridfooter">row but I am footer!</div>
</div>
```

Einfache Funktionen für unsere Beispiele:

```
function isOdd(num) {
  return num % 2;
}

function stringContains(myString, mySubString) {
  return myString.indexOf(mySubString) !== -1;
}
var showClass = true; //we want to add the class
```

Fügen Sie Elementen mit einer `gridrow` Klasse eine ungerade / gerade Klasse `gridrow`

```
$("div.gridrow").toggleClass(function(index, oldClasses, showThisClass) {
  if (isOdd(index)) {
    return "odd";
  } else {
    return "even";
  }
  return oldClasses;
}, showClass);
```

Wenn die Zeile eine `gridfooter` Klasse hat, entfernen Sie die ungeraden / geraden Klassen. `gridfooter` den Rest bei.

```
$("div.gridrow").toggleClass(function(index, oldClasses, showThisClass) {
  var isFooter = stringContains(oldClasses, "gridfooter");
  if (isFooter) {
    oldClasses = oldClasses.replace('even', ' ').replace('odd', ' ');
    $(this).toggleClass("even odd", false);
  }
  return oldClasses;
}, showClass);
```

Die Klassen, die zurückgegeben werden, sind das, was bewirkt wird. Wenn ein Element keinen `gridfooter` , fügen Sie hier eine Klasse für gerade / ungerade hinzu. Dieses Beispiel veranschaulicht die Rückgabe der ALD-Klassenliste. Wenn dies `else return oldClasses;` wird entfernt, nur die neuen Klassen werden hinzugefügt, daher würden in der Zeile mit einer

`gridfooter` Klasse alle Klassen entfernt, wenn wir die alten nicht zurückgegeben hätten - sie wären sonst umgeschaltet (entfernt).

```
$("div.gridrow").toggleClass(function(index, oldClasses, showThisClass) {
  var isFooter = stringContains(oldClasses, "gridfooter");
  if (!isFooter) {
    if (isOdd(index)) {
      return "oddLight";
    } else {
      return "evenLight";
    }
  } else return oldClasses;
}, showClass);
```

## Andere API-Methoden

jQuery bietet eine Vielzahl von Methoden, die zur DOM-Manipulation verwendet werden können.

Die erste ist die .empty () -Methode.

Stellen Sie sich folgendes Markup vor:

```
<div id="content">
  <div>Some text</div>
</div>
```

Durch Aufruf von `$('#content').empty();` Das innere Div würde entfernt werden. Dies kann auch mit `$('#content').html('');` .

Eine weitere praktische Funktion ist die .closest () - Funktion:

```
<tr id="row_1">
  <td><button type="button" class="delete">Delete</button>
</tr>
```

Wenn Sie die nächste Zeile einer Schaltfläche suchen möchten, die in einer der Zeilenzellen angeklickt wurde, können Sie Folgendes tun:

```
$('.delete').click(function() {
  $(this).closest('tr');
});
```

Da es wahrscheinlich mehrere Zeilen sein, jede mit ihren eigenen `delete` verwenden wir `$(this)` innerhalb der .click () Funktion den Bereich auf die Schaltfläche begrenzen wir tatsächlich angeklickt.

Wenn Sie die `id` der Zeile mit der Schaltfläche " `Delete` möchten, auf die Sie geklickt haben, können Sie etwa `Delete` eingeben:

```
$('.delete').click(function() {
  var $row = $(this).closest('tr');
```

```
  var id = $row.attr('id');
});
```

In der Regel wird empfohlen, Variablen, die jQuery-Objekte enthalten, ein `$` (Dollarzeichen) voranzustellen, um die Variablen eindeutig darzustellen.

Eine Alternative zu `.closest()` ist die Methode .parents () :

```
$('.delete').click(function() {
  var $row = $(this).parents('tr');
  var id = $row.attr('id');
});
```

und es gibt auch eine .parent () - Funktion:

```
$('.delete').click(function() {
  var $row = $(this).parent().parent();
  var id = $row.attr('id');
});
```

`.parent()` geht nur eine Ebene des DOM-Baums nach oben. `.parent()` ist es ziemlich unflexibel. Wenn Sie die Schaltfläche "Löschen" so ändern, dass sie zum Beispiel innerhalb eines `span` , wird der jQuery-Selektor beschädigt.

## .html ()

Sie können diese Methode verwenden, um den gesamten HTML-Code innerhalb des Selektors zu ersetzen. Angenommen, Sie haben ein HTML-Element wie dieses

```
<div class="row">
    <div class="col-md-12">
        <div id="information">
            <p>Old message</p>
        </div>
    </div>
</div>
```

Sie könnten `.html()` . Entfernen und Hinzufügen eines Warn- oder Informationstextes, um Benutzer mit einer Zeile zu warnen.

```
$("#information").html("<p>This is the new alert!</p>");
```

## Elemente sortieren

Um Elemente effizient zu sortieren (alle gleichzeitig und mit minimaler DOM-Unterbrechung), müssen wir:

1. **Finde** die Elemente
2. **Sortiere** basierend auf einer festgelegten Bedingung
3. **Legen** Sie in dem DOM zurück

```
<ul id='my-color-list'>
    <li class="disabled">Red</li>
    <li>Green</li>
    <li class="disabled">Purple</li>
    <li>Orange</li>
</ul>
```

1. Finde sie - `.children()` oder `.find()`

   Dies gibt uns ein Array-ähnliches Objekt, mit dem wir spielen können.

   ```
   var $myColorList = $('#my-color-list');

   // Elements one layer deep get .children(), any deeper go with .find()
   var $colors = $myColorList.children('li');
   ```

2. Ordnen Sie sie neu an - `Array.prototype.sort()`

   Dies ist derzeit so eingestellt, dass die Elemente in aufsteigender Reihenfolge basierend auf dem HTML-Inhalt (auch bekannt als deren Farben) zurückgegeben werden.

   ```
   /**
    * Bind $colors to the sort method so we don't have to travel up
    * all these properties more than once.
    */
   var sortList = Array.prototype.sort.bind($colors);

   sortList(function ( a, b ) {

       // Cache inner content from the first element (a) and the next sibling (b)
       var aText = a.innerHTML;
       var bText = b.innerHTML;

       // Returning -1 will place element `a` before element `b`
       if ( aText < bText ) {
           return -1;
       }

       // Returning 1 will do the opposite
       if ( aText > bText ) {
           return 1;
       }

       // Returning 0 leaves them as-is
       return 0;
   });
   ```

3. Fügen Sie sie ein - `.append()`

   *Beachten Sie, dass wir die Elemente nicht zuerst `append()` - `append()` wird Elemente verschieben, die bereits im DOM vorhanden sind, sodass keine zusätzlichen Kopien vorhanden sind*

   ```
   // Put it right back where we got it
   $myColorList.append($colors);
   ```

# Mach es niedlich

## Fügen Sie eine Sortierschaltfläche hinzu

```
<!-- previous HTML above -->
<button type='button' id='btn-sort'>
    Sort
</button>
```

## Legen Sie den Anfangswert der Sortierrichtung fest

```
var ascending = true;
```

## Zwischenspeichern Sie unsere DOM-Elemente und `sortList()` hier, um die DOM-Verarbeitung zu minimieren

```
var $myColorList = $('#my-color-list');
var $colors = $myColorList.children('li');
var sortList = Array.prototype.sort.bind($colors);
```

## Wickeln Sie alles in eine `doSort()` Funktion

```
// Put the sortList() and detach/append calls in this portable little thing
var doSort = function ( ascending ) {

    sortList(function ( a, b ) {
        // ...
    });

    $myColorList.append($colors);
};
```

## Klick-Handler für `$('#btn-sort')`

```
$('#btn-sort').on('click', function () {
    // Run the sort and pass in the direction value
    doSort(ascending);

    // Toggle direction and save
    ascending = !ascending;
});
```

# Jetzt alle zusammen

```
var ascending = true;

var $myColorList = $('#my-color-list');
var $colors = $myColorList.children('li');
var sortList = Array.prototype.sort.bind($colors);

var doSort = function ( ascending ) {

    sortList(function ( a, b ) {

        var aText = a.innerHTML;
        var bText = b.innerHTML;

        if ( aText < bText ) {
            return ascending ? -1 : 1;
        }

        if ( aText > bText ) {
            return ascending ? 1 : -1;
        }

        return 0;
    });

    $myColorList.append($colors);

};

$('#btn-sort').on('click', function () {
    doSort(ascending);
    ascending = !ascending;
});
```

---

**Bonus**

# Mehrstufige Sortierung (Gruppierung sortierter Elemente)

```
// ...

var doSort = function ( ascending ) {

    sortList(function ( a, b ) {

        // ...initial sorting...

    }).sort(function ( a, b ) {

        // We take the returned items and sort them once more
        var aClass = a.className;
        var bClass = b.className;

        // Let's group the disabled items together and put them at the end

        /**
         * If the two elements being compared have the same class
         * then there's no need to move anything.
         */
        if ( aClass !== bClass ) {
```

```
            return aClass === 'disabled' ? 1 : -1;
        }
        return 0;
    });

    // And finalize with re-insert
    $myColorList.append($colors);
};

// ...
```

**Kannst du noch einen Schritt weiter gehen?**

## Fügen Sie eine weitere Schaltfläche hinzu, um die deaktivierte Gruppensortierung umzuschalten

MDN - Array.prototype.sort ()

DOM-Manipulation online lesen: https://riptutorial.com/de/jquery/topic/512/dom-manipulation

# Kapitel 10: Element Sichtbarkeit

## Parameter

| Parameter | Einzelheiten |
|---|---|
| Dauer | Wenn bestanden, werden die Effekte von `.hide()` , `.show()` und `.toggle()` animiert; Die Elemente werden allmählich ein- oder ausgeblendet. |

## Examples

### Überblick

```
$(element).hide()            // sets display: none
$(element).show()            // sets display to original value
$(element).toggle()          // toggles between the two
$(element).is(':visible')    // returns true or false
$('element:visible')         // matches all elements that are visible
$('element:hidden')          // matches all elements that are hidden

$('element').fadeIn();          // display the element
$('element').fadeOut();          // hide the element

$('element').fadeIn(1000);          // display the element using timer
$('element').fadeOut(1000);          // hide the element using timer

// display the element using timer and a callback function
$('element').fadeIn(1000, function(){
 // code to execute
});

// hide the element using timer and a callback function
$('element').fadeOut(1000, function(){
   // code to execute
});
```

### Möglichkeiten umschalten

#### Einfacher `toggle()` Fall

```
function toggleBasic() {
  $(".target1").toggle();
}
```

#### Mit bestimmter *Dauer*

```
function toggleDuration() {
  $(".target2").toggle("slow"); // A millisecond duration value is also acceptable
}
```

### ... und *Rückruf*

```
function toggleCallback() {
  $(".target3").toggle("slow",function(){alert('now do something');});
}
```

### ... oder mit *Lockerung* und Rückruf.

```
function toggleEasingAndCallback() {
  // You may use jQueryUI as the core only supports linear and swing easings
  $(".target4").toggle("slow","linear",function(){alert('now do something');});
}
```

### ... oder mit einer Vielzahl von *Optionen* .

```
function toggleWithOptions() {
  $(".target5").toggle(
    { // See all possible options in: api.jquery.com/toggle/#toggle-options
      duration:1000, // milliseconds
      easing:"linear",
      done:function(){
        alert('now do something');
      }
    }
  );
}
```

### Es ist auch möglich, eine *Folie* als Animation mit `slideToggle()`

```
function toggleSlide() {
  $(".target6").slideToggle(); // Animates from top to bottom, instead of top corner
}
```

### ... oder durch `fadeToggle()` Deckkraft mit `fadeToggle()` / `fadeToggle()`

```
function toggleFading() {
  $( ".target7" ).fadeToggle("slow")
}
```

### ... oder eine Klasse mit `toggleClass()`

```
function toggleClass() {
  $(".target8").toggleClass('active');
}
```

### Ein häufiger Fall ist die Verwendung von `toggle()` , um ein Element anzuzeigen, während das andere ausgeblendet wird (dieselbe Klasse)

```
function toggleX() {
  $(".targetX").toggle("slow");
}
```

Alle obigen Beispiele finden Sie hier

Element Sichtbarkeit online lesen: https://riptutorial.com/de/jquery/topic/1298/element-sichtbarkeit

# Kapitel 11: Jede Funktion

## Examples

### Grundlegende Verwendung

```
// array
var arr = [
   'one',
   'two',
   'three',
   'four'
];
$.each(arr, function (index, value) {
  console.log(value);

  // Will stop running after "three"
  return (value !== 'three');
});
// Outputs: one two three
```

### jQuery jede Funktion

HTML:

```
<ul>
  <li>Mango</li>
  <li>Book</li>
</ul>
```

Skript:

```
$( "li" ).each(function( index ) {
  console.log( index + ": " + $( this ).text() );
});
```

Eine Nachricht wird somit für jeden Eintrag in der Liste protokolliert:

0: Mango

1: Buchen

Jede Funktion online lesen: https://riptutorial.com/de/jquery/topic/10853/jede-funktion

# Kapitel 12: jQuery .animate () - Methode

## Syntax

1. (Selektor) .animate ({Stile}, {Optionen})

## Parameter

| Parameter | Einzelheiten |
|---|---|
| Eigenschaften | Ein Objekt mit CSS-Eigenschaften und Werten, zu dem die Animation verschoben wird |
| Dauer | (Standard: 400) Eine Zeichenfolge oder Nummer, die bestimmt, wie lange die Animation ausgeführt wird |
| Lockerung | (Standardeinstellung: swing) Eine Zeichenfolge, die angibt, welche Beschleunigungsfunktion für den Übergang verwendet werden soll |
| Komplett | Eine Funktion, die aufgerufen wird, sobald die Animation abgeschlossen ist. Sie wird einmal pro übereinstimmendem Element aufgerufen. |
| Start | Gibt eine Funktion an, die ausgeführt werden soll, wenn die Animation beginnt. |
| Schritt | gibt eine Funktion an, die für jeden Schritt in der Animation ausgeführt werden soll. |
| Warteschlange | ein boolescher Wert, der angibt, ob die Animation in der Effektwarteschlange platziert werden soll oder nicht. |
| Fortschritt | gibt eine Funktion an, die nach jedem Schritt der Animation ausgeführt werden soll. |
| erledigt | Gibt eine Funktion an, die ausgeführt werden soll, wenn die Animation endet. |
| Scheitern | Gibt eine Funktion an, die ausgeführt werden soll, wenn die Animation nicht abgeschlossen wird. |
| specialEasing | Eine Karte mit einer oder mehreren CSS-Eigenschaften aus dem styles-Parameter und den entsprechenden Beschleunigungsfunktionen. |
| immer | Gibt eine Funktion an, die ausgeführt werden soll, wenn die Animation ohne Abschluss beendet wird. |

# Examples

## Animation mit Rückruf

Manchmal müssen wir die Wortposition von einem Ort zum anderen ändern oder die Größe der Wörter verkleinern und die Farbe der Wörter automatisch ändern, um die Attraktivität unserer Website oder Web-Apps zu verbessern. JQuery hilft mit diesem Konzept sehr viel mit `fadeIn(), hide(), slideDown()` aber seine Funktionalität ist begrenzt und es wurde nur die spezifische Aufgabe erledigt, die ihm zugewiesen wurde.

Jquery behebt dieses Problem, indem es eine erstaunliche und flexible Methode namens `.animate()` . Mit dieser Methode können Sie benutzerdefinierte Animationen festlegen, die css-Eigenschaften verwenden, die die Berechtigung zum Überfliegen von Grenzen geben. Wenn wir zum Beispiel die Eigenschaft css style als `width:200;` angeben `width:200;` und die aktuelle Position des DOM-Elements ist 50, animiere Methode den aktuellen Positionswert von einem gegebenen CSS-Wert und animiere dieses Element auf 150.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
    $("#btn1").click(function(){
        $("#box").animate({width: "200px"});
    });
</script>

<button id="btn1">Animate Width</button>
<div id="box" style="background:#98bf21;height:100px;width:100px;margin:6px;"></div>
```

### Liste der CSS-Style-Eigenschaften, die die `.animate()` -Methode zulassen.

```
backgroundPositionX, backgroundPositionY, borderWidth, borderBottomWidth, borderLeftWidth,
borderRightWidth, borderTopWidth, borderSpacing, margin, marginBottom, marginLeft,
marginRight, marginTop, outlineWidth, padding, paddingBottom, paddingLeft, paddingRight,
paddingTop, height, width, maxHeight, maxWidth, minHeight, minWidth, fontSize, bottom, left,
right, top, letterSpacing,  wordSpacing, lineHeight, textIndent,
```

### In der `.animate()` -Methode angegebene `.animate()` .

```
milliseconds (Ex: 100, 1000, 5000, etc.),
"slow",
"fast"
```

### In `.animate()` -Methode angegebene `.animate()` .

| |
|---|
| "Swing" |
| "linear" |

Hier einige Beispiele mit komplexen Animationsoptionen.

ZB 1:

```
$( "#book" ).animate({
  width: [ "toggle", "swing" ],
  height: [ "toggle", "swing" ],
  opacity: "toggle"
 }, 5000, "linear", function() {
    $( this ).after( "<div>Animation complete.</div>" );
});
```

ZB 2:

```
  $("#box").animate({
     height: "300px",
     width: "300px"
     }, {
     duration: 5000,
     easing: "linear",
     complete: function(){
        $(this).after("<p>Animation is complete!</p>");
     }
  });
```

jQuery .animate () - Methode online lesen: https://riptutorial.com/de/jquery/topic/5064/jquery--animate------methode

# Kapitel 13: jQuery Verschobene Objekte und Versprechen

## Einführung

jQuery-Versprechen sind eine clevere Art, asynchrone Vorgänge auf Bausteinweise miteinander zu verketten. Dies ersetzt das Verschachteln von Rückrufen aus der alten Schule, die nicht so einfach zu reorganisieren sind.

## Examples

### Grundversprechen schaffen

Hier ist ein sehr einfaches Beispiel für eine Funktion, die " *verspricht,* nach Ablauf einer bestimmten Zeit fortzufahren". Dies geschieht durch Erstellen eines neuen `Deferred` Objekts, das später aufgelöst wird und das Versprechen des `Deferred` Objekts zurückgibt:

```
function waitPromise(milliseconds){

   // Create a new Deferred object using the jQuery static method
   var def = $.Deferred();

   // Do some asynchronous work - in this case a simple timer
   setTimeout(function(){

       // Work completed... resolve the deferred, so it's promise will proceed
       def.resolve();
   }, milliseconds);

   // Immediately return a "promise to proceed when the wait time ends"
   return def.promise();
}
```

Und wie folgt verwenden:

```
waitPromise(2000).then(function(){
     console.log("I have waited long enough");
});
```

### Asynchrone Versprechen Verkettung

Wenn Sie mehrere asynchrone Tasks haben, die nacheinander ausgeführt werden müssen, müssen Sie ihre Versprechungsobjekte miteinander verketten. Hier ist ein einfaches Beispiel:

```
function First() {
    console.log("Calling Function First");
    return $.get("/ajax/GetFunction/First");
}
```

```
function Second() {
    console.log("Calling Function Second");
    return $.get("/ajax/GetFunction/Second");
}

function Third() {
    console.log("Calling Function Third");
    return $.get("/ajax/GetFunction/Third");
}

function log(results){
    console.log("Result from previous AJAX call: " + results.data);
}

First().done(log)
       .then(Second).done(log)
       .then(Third).done(log);
```

## jQuery ajax () Erfolg, Fehler VS .done (), .fail ()

**Erfolg und Fehler:** Ein **erfolgreicher** Rückruf, der nach erfolgreichem Abschluss einer Ajax-Anforderung aufgerufen wird.

Ein **Fehler-** Callback, der aufgerufen wird, falls während der Anforderung ein Fehler auftritt.

**Beispiel:**

```
 $.ajax({
        url: 'URL',
        type: 'POST',
        data: yourData,
        datatype: 'json',
        success: function (data) { successFunction(data); },
        error: function (jqXHR, textStatus, errorThrown) { errorFunction(); }
    });
```

**.done () und .fail ():**

.ajax (). done (Funktion (Daten, TextStatus, JqXHR) {}); Ersetzt die in jQuery 1.8 veraltete Methode .success (). Dies ist ein alternatives Konstrukt für die oben beschriebene Erfolgsrückruffunktion.

.ajax (). fail (Funktion (jqXHR, textStatus, errorThrown) {}); Ersetzt die in jQuery 1.8 veraltete Methode .error (). Dies ist ein alternatives Konstrukt für die vollständige Callback-Funktion oben.

**Beispiel:**

```
$.ajax({
    url: 'URL',
    type: 'POST',
    data: yourData,
    datatype: 'json'
})
.done(function (data) { successFunction(data); })
```

```
.fail(function (jqXHR, textStatus, errorThrown) { serrorFunction(); });
```

## Holen Sie sich den aktuellen Stand eines Versprechens

Standardmäßig steht der Status eines Versprechens bei der Erstellung aus. Der Status eines Versprechens wird geändert, wenn das zurückgestellte Objekt, das das Versprechen erstellt hat, es entweder löst oder ablehnt.

```
var deferred = new $.Deferred();
var d1= deferred.promise({
    prop: "value"
});
var d2= $("div").promise();
var d3= $("div").hide(1000).promise();

console.log(d1.state()); // "pending"
console.log(d2.state()); // "resolved"
console.log(d3.state()); // "pending"
```

jQuery Verschobene Objekte und Versprechen online lesen:
https://riptutorial.com/de/jquery/topic/8308/jquery-verschobene-objekte-und-versprechen

# Kapitel 14: Kontrollkästchen Alle auswählen, wenn andere Kontrollkästchen geändert werden sollen

## Einführung

Ich habe verschiedene Stackoverflow-Beispiele und -Antworten verwendet, um zu diesem wirklich einfachen Beispiel zu gelangen, wie das Kontrollkästchen "Alle auswählen" in Verbindung mit einem automatischen Aktivieren / Deaktivieren des Status des Gruppen-Kontrollkästchens geändert wird. Einschränkung: Die "Alle auswählen" -ID muss mit den Eingabenamen übereinstimmen, um die Gruppe "Alle auswählen" zu erstellen. Im Beispiel lautet die Eingabe select all ID cbGroup1. Die Eingabenamen sind auch cbGroup1

Der Code ist sehr kurz, nicht viel if-Anweisung (zeit- und ressourcenintensiv).

## Examples

### 2 Markieren Sie alle Kontrollkästchen mit den entsprechenden Kontrollkästchen

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

<p>
<input id="cbGroup1" type="checkbox">Select all
<input name="cbGroup1" type="checkbox" value="value1_1">Group1 value 1
<input name="cbGroup1" type="checkbox" value="value1_2">Group1 value 2
<input name="cbGroup1" type="checkbox" value="value1_3">Group1 value 3
</p>
<p>
<input id="cbGroup2" type="checkbox">Select all
<input name="cbGroup2" type="checkbox" value="value2_1">Group2 value 1
<input name="cbGroup2" type="checkbox" value="value2_2">Group2 value 2
<input name="cbGroup2" type="checkbox" value="value2_3">Group2 value 3
</p>

<script type="text/javascript" language="javascript">
    $("input").change(function() {
      $('input[name=\''+this.id+'\']').not(this).prop('checked', this.checked);
      $('#'+this.name).prop('checked', $('input[name=\''+this.name+'\']').length ===
$('input[name=\''+this.name+'\']').filter(':checked').length);
    });
</script>
```

Kontrollkästchen Alle auswählen, wenn andere Kontrollkästchen geändert werden sollen online lesen: https://riptutorial.com/de/jquery/topic/10076/kontrollkastchen-alle-auswahlen--wenn-andere-kontrollkastchen-geandert-werden-sollen

# Kapitel 15: Plugins

## Examples

### Plugins - Erste Schritte

Die jQuery-API kann durch Hinzufügen zu ihrem Prototyp erweitert werden. Zum Beispiel stehen für die vorhandene API bereits viele Funktionen zur Verfügung, wie `.hide()` , `.fadeIn()` , `.hasClass()` usw.

Der jQuery-Prototyp wird über `$.fn` , der Quellcode enthält die Zeile

```
jQuery.fn = jQuery.prototype
```

Durch das Hinzufügen von Funktionen zu diesem Prototyp können diese Funktionen von jedem erstellten jQuery-Objekt aufgerufen werden (dies wird implizit bei jedem Aufruf von jQuery oder bei jedem Aufruf von $ ausgeführt, wenn Sie dies vorziehen).

Ein erstelltes jQuery-Objekt enthält ein internes Array mit Elementen, die auf dem an ihn übergebenen Selektor basieren. `$('.active')` zum Beispiel ein jQuery-Objekt, das zum Zeitpunkt des Aufrufs Elemente mit der aktiven Klasse enthält (wie in, dies ist keine Live-Gruppe von Elementen).

`this` Wert innerhalb der Plugin-Funktion bezieht sich auf das erstellte jQuery-Objekt. Als Ergebnis wird `this` verwendet, um die übereinstimmende Menge darzustellen.

**Grundlegendes Plugin** :

```
$.fn.highlight = function() {
    this.css({ background: "yellow" });
};

// Use example:
$("span").highlight();
```

jsFiddle-Beispiel

#### Kettenfähigkeit & Wiederverwendbarkeit

**Im Gegensatz zum obigen Beispiel** wird erwartet, dass jQuery-Plugins **verkettbar sind** .
Dies bedeutet die Möglichkeit, mehrere Methoden zu einer gleichen Elementgruppe wie `$(".warn").append("WARNING! ").css({color:"red"})` (wie wir das verwendet haben) `.css()` -Methode nach dem `.append()` , beide Methoden gelten für dieselbe `.warn` Sammlung.)

Die Möglichkeit, dasselbe Plugin für verschiedene Collections zu verwenden, und die Weitergabe verschiedener Anpassungsoptionen spielt eine wichtige Rolle bei der **Anpassung /**

### Wiederverwendbarkeit

```
(function($) {
  $.fn.highlight = function( custom ) {

    // Default settings
    var settings = $.extend({
        color : "",              // Default to current text color
        background : "yellow"    // Default to yellow background
    }, custom);

    return this.css({     // `return this` maintains method chainability
        color : settings.color,
        backgroundColor : settings.background
    });

  };
}( jQuery ));


// Use Default settings
$("span").highlight();    // you can chain other methods

// Use Custom settings
$("span").highlight({
    background: "#f00",
    color: "white"
});
```

jsFiddle Demo

### Freiheit

Die obigen Beispiele dienen zum Verständnis der grundlegenden Plugin-Erstellung. Denken Sie daran, einen Benutzer nicht auf eine begrenzte Anzahl von Anpassungsoptionen zu beschränken.

Nehmen wir zum Beispiel Sie wollen einen bauen `.highlight()` Plugin , wo Sie einen gewünschten **Text** String übergeben kann , die hervorgehoben werden und ermöglichen maximale Freiheit in Bezug auf Stile:

```
//...
// Default settings
var settings = $.extend({
   text : "",          // text to highlight
   class : "highlight" // reference to CSS class
}, custom);

return this.each(function() {
   // your word highlighting logic here
});
//...
```

Der Benutzer kann nun einen gewünschten **Text übergeben** und die hinzugefügten Stile mithilfe einer benutzerdefinierten CSS-Klasse vollständig steuern:

```
$("#content").highlight({
```

```
    text : "hello",
    class : "makeYellowBig"
});
```

jsFiddle-Beispiel

## jQuery.fn.extend () -Methode

Diese Methode erweitert das jQuery-Prototypobjekt ($ .fn), um neue benutzerdefinierte Methoden bereitzustellen, die mit der jQuery () - Funktion verkettet werden können.

Zum Beispiel:

```
<div>Hello</div>
<div>World!</div>

<script>
jQuery.fn.extend({
  // returns combination of div texts as a result
  getMessage: function() {
    var result;
    // this keyword corresponds result array of jquery selection
    this.each(function() {
        // $(this) corresponds each div element in this loop
        result = result + " " + $(this).val();
    });
    return result;
  }
});

// newly created .getMessage() method
var message = $("div").getMessage();

// message = Hello World!
console.log(message);
</script>
```

Plugins online lesen: https://riptutorial.com/de/jquery/topic/1805/plugins

# Kapitel 16: Selektoren

## Einführung

Ein jQuery-Selektor wählt oder sucht ein DOM-Element (Document Object Model) in einem HTML-Dokument. Es wird verwendet, um HTML-Elemente basierend auf ID, Name, Typen, Attributen, Klasse usw. auszuwählen. Sie basiert auf vorhandenen CSS-Selektoren.

## Syntax

- Tag: Keine Markierung, verwenden Sie das Tag direkt
- Id: `#id`
- Klasse: `.className`
- Attribut: `[attributeName]`
- Attribut mit Wert: `[attributeName ='value']`
- Attribut beginnt mit Wert `^=` : `[attributeName ^= 'value']`
- Attribut endet mit dem Wert `$=` : `[attributeName $='value']`
- Attribut enthält Wert `*=` : `[attributeName *= 'value']`
- Pseudo-Selektor:: `:pseudo-selector`
- Beliebiger Nachkomme: Leerzeichen zwischen Selektoren
- Direkte Kinder: `>` zwischen Selektoren
- Benachbarte Geschwister nach dem ersten: `+`
- Nicht benachbarter Bruder folgt dem ersten: `~`
- Oder: `,` (Komma) zwischen Selektor

## Bemerkungen

Wenn Sie `selectors` für eine `class` oder eine `id` oder ein `attribute` das einige Sonderzeichen enthält, wie

```
! " # $ % & ' ( ) * + , . / : ; < = > ? @ [ \ ] ^ { | } ~
```

Die Zeichen müssen mit zwei umgekehrten Schrägstrichen `\\` maskiert werden.

z.B.

```
<span id="temp.foobar"><span>
```

die Selektoren werden wie gerahmt,

```
$('#temp\\.foobar')
```

## Examples

## Arten von Selektoren

In jQuery können Sie Elemente auf einer Seite mit vielen verschiedenen Eigenschaften des Elements auswählen, darunter:

- Art
- Klasse
- ICH WÜRDE
- Besitz des Attributs
- Attributwert
- Indexierte Auswahl
- Pseudo-Zustand

Wenn Sie CSS-Selektoren kennen, werden Sie feststellen, dass die Selektoren in jQuery identisch sind (mit geringfügigen Ausnahmen).

Nehmen Sie zum Beispiel den folgenden HTML-Code:

```
<a href="index.html"></a>                  <!-- 1 -->
<a id="second-link"></a>                   <!-- 2 -->
<a class="example"></a>                    <!-- 3 -->
<a class="example" href="about.html"></a>  <!-- 4 -->
<span class="example"></span>              <!-- 5 -->
```

### Auswahl nach Typ:

Der folgende jQuery-Selektor wählt alle `<a>` -Elemente aus, einschließlich 1, 2, 3 und 4.

```
$("a")
```

### Auswahl nach Klasse

Die folgende jQuery Selektor werden alle Elemente der Klasse ausgewählt `example` (einschließlich Nicht-A - Elemente), die 3, 4 und 5.

```
$(".example")
```

### Auswahl über ID

Der folgende jQuery-Selektor wählt das Element mit der angegebenen ID (2) aus.

```
$("#second-link")
```

### Auswahl durch Attributbesitz

Der folgende jQuery-Selektor wählt alle Elemente mit einem definierten `href` Attribut aus, einschließlich 1 und 4.

```
$("[href]")
```

### Auswahl nach Attributwert

Der folgende jQuery-Selektor wählt alle Elemente aus, in denen das `href` Attribut vorhanden ist, mit dem Wert `index.html` , der nur 1 ist.

```
$("[href='index.html']")
```

### Auswahl über indizierte Position ( *indizierte* Auswahl)

Der folgende jQuery-Selektor wählt nur 1 aus, der zweite `<a>` dh. die `second-link` , weil Index geliefert wird `1` wie `eq(1)` (Beachten Sie, dass der Index beginnt um `0` daher die zweiten hier wurde ausgewählt!).

```
$("a:eq(1)")
```

### Auswahl mit indiziertem Ausschluss

So schließen Sie ein Element mithilfe seines Index aus `:not(:eq())`

Die folgenden wählt `<a>` Elemente, mit der Ausnahme , dass mit der Klasse `example` , die 1

```
$("a").not(":eq(0)")
```

### Auswahl mit Ausschluss

Um ein Element von einer Auswahl auszuschließen, verwenden Sie `:not()`

Die folgende wählt `<a>` Elemente, mit Ausnahme derjenigen mit der Klasse `example` , sind die 1 und 2.

```
$("a:not(.example)")
```

### Auswahl nach Pseudo-Status

Sie können in jQuery auch Pseudo-Status auswählen, darunter `:first-child` `:last-child` `:first-of-type` `:last-of-type` usw.

Der folgende jQuery-Selektor wählt nur das erste `<a>` -Element aus: number 1.

```
$("a:first-of-type")
```

### JQuery-Selektoren kombinieren

Sie können Ihre Spezifität auch erhöhen, indem Sie mehrere jQuery-Selektoren kombinieren. Sie können beliebig viele oder alle kombinieren. Sie können auch mehrere Klassen, Attribute und Status gleichzeitig auswählen.

```
$("a.class1.class2.class3#someID[attr1][attr2='something'][attr3='something']:first-of-type:first-child")
```

Dies würde ein `<a>` -Element auswählen, das:

- Verfügt über die folgenden Klassen: `class1, class2, and class3`
- Hat die folgende ID: `someID`
- Hat das folgende Attribut: `attr1`
- Hat die folgenden Attribute und Werte: `attr2` mit Wert `something` , `attr3` mit Wert `something`
- Hat die folgenden Zustände: `first-child` und `first-of-type`

Sie können auch verschiedene Selektoren mit einem Komma trennen:

```
$("a, .class1, #someID")
```

Dies würde auswählen:

- Alle `<a>` Elemente
- Alle Elemente, die die Klasse `class1`
- Ein Element mit der ID `#someID`

---

**Auswahl von Kindern und Geschwistern**

jQuery-Selektoren entsprechen im Allgemeinen den gleichen Konventionen wie CSS, sodass Sie Kinder und Geschwister auf dieselbe Weise auswählen können.

- Verwenden Sie ein Leerzeichen, um ein nicht direktes untergeordnetes Element auszuwählen
- Um ein direktes Kind auszuwählen, verwenden Sie ein `>`
- Um ein benachbartes Geschwister nach dem ersten auszuwählen, verwenden Sie ein `+`
- Um ein nicht benachbartes Geschwister nach dem ersten auszuwählen, verwenden Sie ein `~`

---

**Platzhalterauswahl**

Es kann Fälle geben, in denen wir alle Elemente auswählen möchten, es gibt jedoch keine gemeinsame Eigenschaft (Klasse, Attribut usw.) zur Auswahl. In diesem Fall können wir den `*` Selektor verwenden, der einfach alle Elemente auswählt:

```
$('#wrapper *')    // Select all elements inside #wrapper element
```

## Selektoren kombinieren

Betrachten Sie die folgende DOM-Struktur

```
<ul class="parentUl">
    <li> Level 1
        <ul class="childUl">
            <li>Level 1-1 <span> Item - 1 </span></li>
            <li>Level 1-1 <span> Item - 2 </span></li>
        </ul>
    </li>
```

```
    <li> Level 2
        <ul class="childUl">
            <li>Level 2-1 <span> Item - 1 </span></li>
            <li>Level 2-1 <span> Item - 1 </span></li>
        </ul>
    </li>
</ul>
```

---

# Nachkommen- und Kindselektoren

Wenn ein übergeordneter `<ul>` - `parentUl` seine Nachkommen findet ( `<li>` ),

1. Einfach **`$('parent child')`**

   >> `$('ul.parentUl li')`

   Dadurch werden alle passenden Nachkommen des angegebenen Vorfahren auf *allen Ebenen heruntergefahren* .

2. `>` - **`$('parent > child')`**

   >> `$('ul.parentUl > li')`

   Damit werden alle passenden Kinder gefunden ( *nur 1. Ebene nach unten* ).

3. Kontextbasierter Selektor - **`$('child','parent')`**

   >> `$('li','ul.parentUl')`

   Dies funktioniert genauso wie oben.

4. `find()` - **`$('parent').find('child')`**

   >> `$('ul.parentUl').find('li')`

   Dies funktioniert genauso wie oben.

5. `children()` - **`$('parent').find('child')`**

   >> `$('ul.parentUl').children('li')`

   Dies funktioniert genauso wie oben.

---

# Andere Kombinatoren

### Gruppenauswahl: ","

Wählen Sie alle `<ul>` Elemente UND alle `<li>` Elemente UND alle `<span>` Elemente aus:

```
$('ul, li, span')
```

### Mehrfachauswahl: "" (kein Zeichen)

Wählen Sie alle `<ul>` -Elemente mit der Klasse `parentUl` :

```
$('ul.parentUl')
```

### Benachbarte Geschwisterauswahl: "+"

Wählen Sie alle `<li>` -Elemente aus, die unmittelbar nach einem anderen `<li>` -Element platziert werden:

```
$('li + li')
```

### Allgemeine Geschwisterauswahl: "~"

Wählen Sie alle `<li>` Elemente aus, die Geschwister anderer `<li>` Elemente sind:

```
$('li ~ li')
```

### Überblick

Elemente können von jQuery mit jQuery Selectors ausgewählt werden . Die Funktion gibt entweder ein Element oder eine Liste von Elementen zurück.

## Grundlegende Selektoren

```
$("*")                      // All elements
$("div")                    // All <div> elements
$(".blue")                  // All elements with class=blue
$(".blue.red")              // All elements with class=blue AND class=red
$(".blue,.red")             // All elements with class=blue OR class=red
$("#headline")              // The (first) element with id=headline
$("[href]")                 // All elements with an href attribute
$("[href='example.com']")   // All elements with href=example.com
```

## Beziehungsoperatoren

```
$("div span")          // All <span>s that are descendants of a <div>
$("div > span")        // All <span>s that are a direct child of a <div>
$("a ~ span")          // All <span>s that are siblings following an <a>
$("a + span")          // All <span>s that are immediately after an <a>
```

## Caching-Selektoren

Bei jeder Verwendung eines Selektors in jQuery wird im DOM nach Elementen gesucht, die Ihrer Abfrage entsprechen. Wenn Sie dies zu oft oder wiederholt ausführen, wird die Leistung beeinträchtigt. Wenn Sie mehr als einmal auf einen bestimmten Selektor verweisen, sollten Sie ihn zum Cache hinzufügen, indem Sie ihn einer Variablen zuweisen:

```
var nav = $('#navigation');
nav.show();
```

Dies würde ersetzen:

```
$('#navigation').show();
```

Das Zwischenspeichern dieser Auswahl kann hilfreich sein, wenn Ihre Website dieses Element häufig ein- und ausblenden muss. Wenn mehrere Elemente mit demselben Selektor vorhanden sind, wird die Variable zu einem Array dieser Elemente:

```
<div class="parent">
    <div class="child">Child 1</div>
    <div class="child">Child 2</div>
</div>

<script>
    var children = $('.child');
    var firstChildText = children[0].text();
    console.log(firstChildText);

    // output: "Child 1"
</script>
```

**HINWEIS:** Das Element muss zum Zeitpunkt seiner Zuweisung zu einer Variablen im DOM vorhanden sein. Wenn sich im DOM kein Element mit einer Klasse mit dem Namen `child` , wird ein leeres Array in dieser Variablen gespeichert.

```
<div class="parent"></div>

<script>
    var parent   = $('.parent');
    var children = $('.child');
    console.log(children);

    // output: []

    parent.append('<div class="child">Child 1</div>');
    children = $('.child');
    console.log(children[0].text());

    // output: "Child 1"
</script>
```

Denken Sie daran, den Selektor erneut der Variablen zuzuweisen, nachdem Sie Elemente im DOM mit diesem Selektor hinzugefügt oder entfernt haben.

**Hinweis** : Beim Zwischenspeichern von Selektoren beginnen viele Entwickler den Variablennamen mit einem `$` um anzuzeigen, dass die Variable ein jQuery-Objekt ist.

```
var $nav = $('#navigation');
$nav.show();
```

## DOM-Elemente als Selektoren

jQuery akzeptiert eine Vielzahl von Parametern. Einer davon ist ein tatsächliches DOM-Element. Das Übergeben eines DOM-Elements an jQuery bewirkt, dass die zugrunde liegende arrayähnliche Struktur des jQuery-Objekts dieses Element enthält.

jQuery erkennt, dass das Argument ein DOM-Element ist, indem es seinen nodeType untersucht.

Ein DOM-Element wird am häufigsten in Callbacks verwendet, wobei das aktuelle Element an den jQuery-Konstruktor übergeben wird, um Zugriff auf die jQuery-API zu erhalten.

Wie in `each` Callback (Anmerkung: Jede ist eine Iteratorfunktion).

```
$(".elements").each(function(){
    //the current element is bound to `this` internally by jQuery when using each
    var currentElement = this;

    //at this point, currentElement (or this) has access to the Native API

    //construct a jQuery object with the currentElement(this)
    var $currentElement = $(this);

    //now $currentElement has access to the jQuery API
});
```

## HTML-Strings als Selektoren

jQuery akzeptiert eine Vielzahl von Parametern als "Selektoren". Einer davon ist ein HTML-String. Die Übergabe einer HTML-Zeichenfolge an jQuery bewirkt, dass die zugrunde liegende arrayähnliche Struktur des jQuery-Objekts den erstellten HTML-Code enthält.

jQuery verwendet regex, um zu bestimmen, ob die an den Konstruktor übergebene Zeichenfolge eine HTML-Zeichenfolge ist und außerdem mit `<` beginnen *muss* . Dieser Regex ist definiert als `rquickExpr = /^(?:\s*(<[\w\W]+>)[^>]*|#([\w-]*))$/` ( Erklärung bei regex101.com ).

Eine HTML-Zeichenfolge als Selektor wird am häufigsten verwendet, wenn nur DOM-Elemente im Code erstellt werden müssen. Dies wird häufig von Bibliotheken für Modal-Popouts verwendet.

Zum Beispiel eine Funktion, die ein in ein div eingeschlossenes Ankertag als Vorlage zurückgegeben hat

```
function template(href,text){
    return $("<div><a href='" + href + "'>" + text + "</a></div>");
}
```

Würde ein jQuery-Objekt zurückgeben, das hält

```
<div>
    <a href="google.com">Google</a>
</div>
```

wenn als `template("google.com","Google")` aufgerufen `template("google.com","Google")`.

Selektoren online lesen: https://riptutorial.com/de/jquery/topic/389/selektoren

# Kapitel 17: Veranstaltungen

## Bemerkungen

jQuery behandelt Ereignisse intern über die Funktion addEventListener . Das bedeutet, dass es völlig legal ist, mehrere Funktionen für dasselbe DOM-Element an dasselbe Ereignis gebunden zu haben.

## Examples

### Event-Handler anhängen und trennen

# Hängen Sie einen Ereignishandler an

Seit Version 1.7 hat jQuery die Ereignis-API `.on()` . Auf diese Weise können alle Standardereignisse für Javascript oder benutzerdefinierte Ereignisse an das aktuell ausgewählte jQuery-Element gebunden werden. Es gibt Verknüpfungen wie `.click()` , aber `.on()` bietet Ihnen mehr Optionen.

---

## HTML

```
<button id="foo">bar</button>
```

## jQuery

```
$( "#foo" ).on( "click", function() {
  console.log( $( this ).text() ); //bar
});
```

---

# Trennen Sie einen Ereignishandler

Natürlich haben Sie auch die Möglichkeit, Ereignisse von Ihren jQuery-Objekten zu trennen. Verwenden Sie dazu `.off( events [, selector ] [, handler ] )` .

---

## HTML

```
<button id="hello">hello</button>
```

## jQuery

```
$('#hello').on('click', function(){
    console.log('hello world!');
    $(this).off();
});
```

Wenn Sie auf die Schaltfläche klicken, verweist `$(this)` auf das aktuelle jQuery-Objekt und entfernt alle angefügten Ereignishandler daraus. Sie können auch angeben, welcher Ereignishandler entfernt werden soll.

## jQuery

```
$('#hello').on('click', function(){
    console.log('hello world!');
    $(this).off('click');
});

$('#hello').on('mouseenter', function(){
    console.log('you are about to click');
});
```

In diesem Fall funktioniert das `mouseenter` Ereignis nach dem Klicken weiterhin.

### Delegierte Ereignisse

Beginnen wir mit dem Beispiel. Hier ist ein sehr einfaches Beispiel-HTML.

# Beispiel HTML

```
<html>
    <head>
    </head>
    <body>
        <ul>
            <li>
                <a href="some_url/">Link 1</a>
            </li>
            <li>
                <a href="some_url/">Link 2</a>
            </li>
            <li>
                <a href="some_url/">Link 3</a>
            </li>
        </ul>
    </body>
</html>
```

---

# Das Problem

In diesem Beispiel möchten wir nun allen `<a>` Elementen einen Ereignis-Listener hinzufügen. Das Problem ist, dass die Liste in diesem Beispiel dynamisch ist. `<li>` -Elemente werden im Laufe der Zeit hinzugefügt und entfernt. Die Seite wird jedoch zwischen den Änderungen nicht aktualisiert. Dies würde uns ermöglichen, einfache Klickereignis-Listener für die Verknüpfungsobjekte zu verwenden (dh `$('a').click()` ).

Das Problem, das wir haben, ist das Hinzufügen von Ereignissen zu den `<a>` Elementen, die kommen und gehen.

---

# Hintergrundinformationen - Weitergabe von Ereignissen

Delegierte Ereignisse sind nur aufgrund der Weitergabe von Ereignissen möglich (häufig als Ereignisblubbern bezeichnet). Jedes Mal, wenn ein Ereignis ausgelöst wird, sprudelt es bis zum Dokumentstamm nach oben. Sie *delegieren* die Behandlung eines Ereignisses an ein sich nicht änderndes Vorfahrenelement, daher der Name "delegierte" Ereignisse.

Wenn Sie im obigen Beispiel auf `<a>` Element-Link klicken, wird in diesen Elementen in dieser Reihenfolge ein Klick-Ereignis ausgelöst:

- ein
- li
- ul
- Karosserie
- html
- Dokument Root

---

# Lösung

Da wir wissen, was das Aufblühen von Ereignissen bewirkt, können wir eines der gesuchten Ereignisse abfangen, die sich durch unseren HTML-Code ausbreiten.

In diesem Beispiel ist das Element `<ul>` ein guter Ort, um es zu erfassen, da dieses Element nicht dynamisch ist:

```
$('ul').on('click', 'a', function () {
  console.log(this.href); // jQuery binds the event function to the targeted DOM element
                          // this way `this` refers to the anchor and not to the list
  // Whatever you want to do when link is clicked
});
```

In oben:

- Wir haben 'ul', der Empfänger dieses Ereignis-Listeners
- Der erste Parameter ("Klick") definiert, welche Ereignisse wir zu erkennen versuchen.
- Der zweite Parameter ('a') wird verwendet, um anzugeben, woher das Ereignis *stammen muss* (von allen untergeordneten Elementen unter dem Empfänger dieses Ereignis-Listeners, ul).
- Der dritte Parameter ist der Code, der ausgeführt wird, wenn die Anforderungen des ersten und des zweiten Parameters erfüllt sind.

# Im Detail, wie die Lösung funktioniert

1. Der Benutzer klickt auf das `<a>` -Element
2. Dadurch wird das Klickereignis für das `<a>` -Element `<a>` .
3. Das Ereignis sprudelt zum Dokumentstamm.
4. Das Ereignis sprudelt zuerst zum Element `<li>` und dann zum Element `<ul>` .
5. Der Ereignis-Listener wird ausgeführt, wenn das `<ul>` -Element mit dem Ereignis-Listener verbunden ist.
6. Der Ereignis-Listener erkennt zuerst das auslösende Ereignis. Das sprudelnde Ereignis ist "Klick" und der Listener hat "Klick", es ist ein Pass.
7. Die Listener-Checks versuchen, den zweiten Parameter ('a') mit jedem Element in der Blasenkette abzugleichen. Da das letzte Element in der Kette ein 'a' ist, stimmt dies mit dem Filter überein und dies ist auch ein Durchlauf.
8. Der Code im dritten Parameter betrieben , um die angepassten Element mit , wie es `this` . Wenn die Funktion keinen Aufruf von `stopPropagation()` , wird das Ereignis weiter nach oben in Richtung Stamm ( `document` ) propagiert.

Hinweis: Wenn ein passender, nicht wechselnder Vorgänger nicht verfügbar / zweckmäßig ist, sollten Sie ein `document` . Als Gewohnheit verwenden Sie `'body'` aus folgenden Gründen nicht:

- `body` hat einen Fehler, der mit dem Styling zu tun hat, was dazu führen kann, dass Mausereignisse nicht dazu kommen. Dies ist vom Browser abhängig und kann auftreten, wenn die berechnete Körpergröße 0 ist (z. B. wenn alle untergeordneten Elemente absolute Positionen haben). Mausereignisse blasen immer zum `document` .
- `document` *immer* in Ihrem Skript vorhanden, sodass Sie delegierte Handler an ein `document` außerhalb eines *DOM-ready-Handlers anschließen* können, um sicherzustellen, dass sie weiterhin funktionieren.

### Ereignis zum Laden von Dokumenten .load ()

Wenn Sie möchten, dass Ihr Skript wartet, bis eine bestimmte Ressource geladen wurde, z. B. ein Bild oder eine PDF, können Sie `.load()` verwenden. `.load()` ist eine Abkürzung für die Verknüpfung `.on( "load", handler)` .

**HTML**

```
<img src="image.jpeg" alt="image" id="image">
```

### jQuery

```
$( "#image" ).load(function() {
  // run script
});
```

## Ereignisse zum Wiederholen von Elementen ohne Verwendung von IDs

### Problem

Es gibt eine Reihe sich wiederholender Elemente auf der Seite, die Sie wissen müssen, bei welchem Ereignis ein Ereignis aufgetreten ist, um etwas mit dieser bestimmten Instanz zu tun.

### Lösung

- Geben Sie allen gemeinsamen Elementen eine gemeinsame Klasse
- Wenden Sie den Ereignislistener auf eine Klasse an. `this` interne Ereignishandler ist das passende Auswahlelement, auf dem das Ereignis aufgetreten ist
- Traverse zum äußersten wiederholenden Behälter für die Instanz durch beginnend bei `this`
- Verwenden Sie `find()` in diesem Container, um andere für diese Instanz spezifische Elemente zu isolieren

HTML

```
<div class="item-wrapper" data-item_id="346">
   <div class="item"><span class="person">Fred</span></div>
   <div class="item-toolbar">
      <button class="delete">Delete</button>
   </div>
</div>
<div class="item-wrapper" data-item_id="393">
   <div clss="item"><span class="person">Wilma</span></div>
   <div class="item-toolbar">
      <button class="delete">Delete</button>
   </div>
</div>
```

jQuery

```
$(function() {
  $('.delete').on('click', function() {
    // "this" is element event occured on
    var $btn = $(this);
    // traverse to wrapper container
    var $itemWrap = $btn.closest('.item-wrapper');
    // look within wrapper to get person for this button instance
    var person = $itemWrap.find('.person').text();
    // send delete to server and remove from page on success of ajax
    $.post('url/string', { id: $itemWrap.data('item_id')}).done(function(response) {
      $itemWrap.remove()
    }).fail(function() {
```

```
      alert('Ooops, not deleted at server');
    });
  });
});
```

### originalEvent

Manchmal gibt es Eigenschaften, die im jQuery-Ereignis nicht verfügbar sind. Um auf die zugrunde liegenden Eigenschaften `Event.originalEvent` verwenden Sie `Event.originalEvent`

# Scroll-Richtung abrufen

```
$(document).on("wheel",function(e){
    console.log(e.originalEvent.deltaY)
    // Returns a value between -100 and 100 depending on the direction you are scrolling
})
```

### Ein- und Ausschalten bestimmter Ereignisse über jQuery. (Named Listeners)

Manchmal möchten Sie alle zuvor registrierten Listener ausschalten.

```
//Adding a normal click handler
$(document).on("click",function(){
    console.log("Document Clicked 1")
});
//Adding another click handler
$(document).on("click",function(){
    console.log("Document Clicked 2")
});
//Removing all registered handlers.
$(document).off("click")
```

Ein Problem bei dieser Methode ist, dass ALLE Listener, die von anderen Plugins usw. an das `document` gebunden wurden, ebenfalls entfernt werden.

**Meistens möchten wir alle Zuhörer abnehmen, die nur von uns angehängt werden.**

Um dies zu erreichen, können wir benannte Hörer als

```
//Add named event listener.
$(document).on("click.mymodule",function(){
    console.log("Document Clicked 1")
});
$(document).on("click.mymodule",function(){
    console.log("Document Clicked 2")
});

//Remove named event listener.
$(document).off("click.mymodule");
```

Dadurch wird sichergestellt, dass ein anderer Klicklistener nicht versehentlich geändert wird.

Veranstaltungen online lesen: https://riptutorial.com/de/jquery/topic/1321/veranstaltungen

# Kapitel 18: Voranstellen

## Examples

### Ein Element an einen Container übergeben

**Lösung 1:**

```
$('#parent').prepend($('#child'));
```

**Lösung 2:**

```
$('#child').prependTo($('#parent'));
```

Beide Lösungen fügen das Element `#child` (Element am Anfang) dem Element `#parent` .

Vor:

```
<div id="parent">
  <span>other content</span>
</div>
<div id="child">

</div>
```

Nach dem:

```
<div id="parent">
  <div id="child">

  </div>
  <span>other content</span>
</div>
```

### Methode voranstellen

`prepend()` - Fügt den durch den Parameter angegebenen Inhalt an den Anfang jedes Elements in der Menge der übereinstimmenden Elemente ein.

**1.** `prepend( content [, content ] )`

```
// with html string
jQuery('#parent').prepend('<span>child</span>');
// or you can use jQuery object
jQuery('#parent').prepend($('#child'));
// or you can use comma seperated multiple elements to prepend
jQuery('#parent').prepend($('#child1'),$('#child2'));
```

**2.** `prepend(function)`

JQuery- *`version: 1.4`* können Sie die Callback-Funktion als Argument verwenden. Wo können Sie Argumente als Indexposition des Elements in der Menge und den alten HTML-Wert des Elements erhalten. Innerhalb der Funktion bezieht sich `this` auf das aktuelle Element in der Menge.

```
jQuery('#parent').prepend(function(i,oldHTML){
     // return the value to be prepend
     return  '<span>child</span>';
});
```

Voranstellen online lesen: https://riptutorial.com/de/jquery/topic/1909/voranstellen

# Credits

| S. No | Kapitel | Contributors |
|---|---|---|
| 1 | Erste Schritte mit jQuery | A.J, acdcjunior, amflare, Anil, bwegs, Community, DGS, empiric, Fueled By Coffee, hairboat, Hirshy, Iceman, Igor Raush, J F, jkdev, John C, Kevin B, Kevin Katzke, Kevin Montrose, Luca Putzu, Matas Vaitkevicius, mico, Mottie, Neal, ni8mr, Prateek, RamenChef, Rion Williams, Roko C. Buljan, secelite, Shaunak D, Stephen Leppik, Suganya, Sverri M. Olsen, the12, Travis J, user2314737, Velocibadgery, Yosvel Quintero |
| 2 | Ajax | Alon Eitan, amflare, Andrew Brooke, Ashkan Mobayen Khiabani, Athafoud, atilacamurca, Ben H, Cass, Community, csbarnes, Dr. J. Testington, Edathadan Chief aka Arun, empiric, hasan, joe_young, John C, kapantzak, Kiren Siva, Lacrioque, Marimba, Nirav Joshi, Ozan, shaN, Shaunak D, Teo Dragovic, Yosvel Quintero |
| 3 | Anhängen | Ashkan Mobayen Khiabani, bipon, Community, Darshak, Deryck, empiric, Flyer53, J F, JF it, Paul Roub, Pranav C Balan, Proto, Shaunak D |
| 4 | Attribute | A.J, acdcjunior, ban17, ochi, Scimonster |
| 5 | Breite und Höhe eines Elements ermitteln und einstellen | Ashkan Mobayen Khiabani |
| 6 | CSS-Manipulation | abaracedo, Ashkan Mobayen Khiabani, Brandt Solovij, J F, j08691, Jonathan Michalik, Kevin B, Petroff, Roko C. Buljan, ScientiaEtVeritas, Shlomi Haver, Sorangwala Abbasali, Stephen Leppik, Sverri M. Olsen |
| 7 | dokumentbereites Ereignis | Adjit, Alon Eitan, amflare, charlietfl, Emanuel Vintilă, Igor Raush, J F, jkdev, Joram van den Boezem, Liam, Mark Schultheiss, Melanie, Nhan, Nico Westerdale, Scimonster, secelite, TheDeadMedic, the-noob, URoy |
| 8 | DOM Traversing | A.J, charlietfl, Community, dlsso, mark.hch, rmondesilva, SGS Venkatesh, sucil, Sverri M. Olsen, The_Outsider, Zaz |
| 9 | DOM-Manipulation | Angelos Chalaris, Assimilater, Brock Davis, DawnPaladin, DefyGravity, Deryck, Marimba, Mark Schultheiss, martincarlin87, Neal, Paul Roub, Shaunak D, still_learning |

| | | |
|---|---|---|
| 10 | Element Sichtbarkeit | Alex Char, Paul Roub, Rupali Pemare, The_Outsider, Theodore K., user2314737, Zaz |
| 11 | Jede Funktion | bipon, Renier |
| 12 | jQuery .animate () - Methode | RamenChef, Rust in Peace, Simplans, VJS |
| 13 | jQuery Verschobene Objekte und Versprechen | Alex, Ashiquzzaman, Gone Coding |
| 14 | Kontrollkästchen Alle auswählen, wenn andere Kontrollkästchen geändert werden sollen | user1851673 |
| 15 | Plugins | hasan, Roko C. Buljan, Travis J |
| 16 | Selektoren | alepeino, Alon Eitan, Brock Davis, Castro Roy, David, DelightedD0D, devlin carnate, dlsso, hasan, Iceman, James Donnelly, JLF, John Slegers, kapantzak, Kevin B, Keyslinger, Matas Vaitkevicius, Melanie, MikeC, Nux, Petr R., rbashish, Scimonster, Shaunak D, Shekhar Pankaj, Sorangwala Abbasali, ssb, Sverri M. Olsen, Travis J, whales, WOUNDEDStevenJones , Zaz |
| 17 | Veranstaltungen | Adjit, charlietfl, DelightedD0D, doydoy44, empiric, Gone Coding, Horst Jahns, Jatniel Prinsloo, Kevin B, Louis, Luca Putzu, Marimba, NotJustin, SGS Venkatesh, Stephen Leppik, Sunny R Gupta, Washington Guedes, WMios, Zakaria Acharki |
| 18 | Voranstellen | Ashkan Mobayen Khiabani, empiric, J F, Pranav C Balan |