

Webbasierte Anwendungen

jQuery

Prof. Dr. Ludger Martin

Gliederung

- ★ Einführung
- ★ Selektoren
- ★ Eventhandling
- ★ DOM Manipulation
- ★ AJAX

Einleitung

★ jQuery

- ★ Objektorientierte Bibliothek
- ★ Zentral ist das `jQuery`-Objekt
- ★ Vereinfacht die JavaScript-Programmierung
- ★ Einfach nutzbare AJAX-Klassen
- ★ Kapselt die Unterschiede in den einzelnen Browsern
- ★ <http://jquery.com/>

Einleitung

- ★ Einbinden (komprimierte Variante):

`<script src="jquery.js"></script>`

- ★ Factory-Funktion/Objekt:

- ★ `$ ()` oder `jQuery ()`

- ★ vergleichbar mit `getElementBy...`

- ★ aber liefert ein `jQuery`-Objekt zurück

- ★ kann auch eine Knotenliste enthalten, wenn Selektor auf mehrere Elemente zutrifft

Selektoren

★ Beispiel:

```
<div id="div1"></div>
```

```
<div id="div2"></div>
```

★ `let div1 = $("#div1");`

`div1` ist ein `jQuery`-Objekt, mit dem das `div`-Element mit ID `div1` modifiziert werden kann

★ `let div = $("div");`

`div` ist ein `jQuery`-Objekt, mit dem alle `div`-Elemente gleichzeitig modifiziert werden können (es ist kein Array wie bei `getElementsByName`)

Selektoren

★ Mögliche Selektoren für `$ ()`

- ★ `$ (element)` : Selektiert übergebenes DOM-Element oder ein Array von DOM-Elementen
- ★ `$ (jquery)` : Selektiert jquery-Element
- ★ `$ (htmlString)` : Erzeugt ein DOM-Element aus String
- ★ `$ (css)` : Selektiert anhand von beliebigen CSS-Selektoren (ID, Klassen, Pseudo-Klassen)

Selektoren

★ Mögliche Selektoren für \$ () (Fortsetzung)

★ Basisselektoren

Selektor	Typ	Erklärung
*	Alle	Selektiert typunabhängig alle Elemente
E	Typ	Selektiert alle Elemente mit dem Bezeichner E
.class	Klasse	Selektiert alle Elemente mit genannter CSS-Klasse
#id	ID-Sel.	Selektiert alle Elemente mit der übergebenen ID

★ Mehrfachselektoren

Selektor	Typ	Erklärung
.class1.class2	Klasse	Selektiert Elemente, die alle genannten CSS-Klassen enthalten

Selektoren

★ Mögliche Selektoren für `$ ()` (Fortsetzung)

★ Gruppen- und Kontextselektoren

Selektor	Erklärung
<code>sel1, sel2, ...</code>	Selektiert die Kombination aus allen durch die Selektionen ausgewählten Knoten.
<code>E F</code>	Selektiert alle Elemente F, die Nachfahren eines Elements E sind
<code>E > F</code>	Selektiert alle Elemente F, die Kindknoten eines Elements E sind
<code>E + F</code>	Selektiert alle unmittelbar auf ein Element vom Typ E folgenden Elemente vom Typ F
<code>E ~ F</code>	Selektiert alle folgenden Geschwisterknoten eines Elements E, die den Typ F besitzen

Selektoren

★ Beispiel:

```
<div id="div1">  
  <p id="p1">Text 1</p>  
  <p id="p2">Text 2  
      
  </p>  
</div>  
<div id="div2"></div>
```

```
$("#div1 *")      → #p1, #p2, #img1  
$("#div1 > *")   → #p1, #p2  
$("#div1 ~ *")   → #div2
```

Selektoren

★ Mögliche Selektoren für `$ ()` (Fortsetzung)

★ Filter

Selektor	Erklärung
<code>:first/:last</code>	Selektiert das erste/letzte Element
<code>:even/:odd</code>	Selektiert alle Elemente mit geradem/ungeradem Index innerhalb der aktuellen Kollektion
<code>:not(sel)</code>	Selektiert alle Elemente, auf die der Selektor <code>sel</code> nicht zutrifft
<code>:contains(t)</code>	Selektiert alle Elemente, die den übergebenen Text <code>t</code> enthalten
<code>:empty</code>	Selektiert alle Elemente ohne Inhalt
<code>:hidden</code>	Selektiert alle Elemente mit <code>display="none"</code> , <code>Höhe/Breite = 0</code> oder <code><input type="hidden"/></code>
<code>:visible</code>	Gegenstück zu <code>:hidden</code>

Selektoren

★ Beispiel:

```
<div id="div1">
  <p id="p1">Text 1</p>
  <p id="p2">Text 2
    
  </p>
</div>
<div id="div2"></div>
```

<code>\$ ("p:first")</code>	→ <code>#p1</code>
<code>\$ ("p:contains(Text 1)")</code>	→ <code>#p1</code>
<code>\$ ("p:not(:contains(Text 1))")</code>	→ <code>#p2</code>
<code>\$ (":empty")</code>	→ <code>#img1, #div2</code>

Selektoren

★ Mögliche Selektoren für `$ ()` (Fortsetzung)

★ Attributfilter

Selektor	Erklärung
<code>[name]</code>	Selektiert die Elemente beliebigen Typs, die über ein Attribut <code>name</code> verfügen
<code>E[name]</code>	Selektiert alle Elemente vom Typ <code>E</code> , die über ein Attribut <code>name</code> verfügen
<code>[name=wert]</code>	Selektiert alle Elemente, wenn der Wert des Attributs dem übergebenen String entspricht
<code>[name*=wert]</code>	Selektiert alle Elemente, wenn der Wert des Attributs dem übergebenen String als Substring enthält



Nur eine Auswahl!

Selektoren

★ Beispiel:

```
<div id="div1">  
  <input id="i1" type="number" />  
  <input id="i2" type="checkbox" />  
</div>  
<div></div>
```

```
$("[id]")           → #div1, #i1, #i2  
$("input[id]")      → #i1, #i2  
$("[type=checkbox]")   → #i2
```

Eventhandling

★ Eventhandling

★ Event-Objekt

Eigenschaft	Erklärung
<code>e.target</code>	Das DOM-Element, an dem das Ereignis stattfand
<code>e.currentTarget</code>	Das DOM-Element, das während der Capturing-/ Bubbling-Phase Ziel des Ereignisses ist
<code>e.clientX/Y</code>	Mausposition relativ zum Browserfenster
<code>e.timeStamp</code>	Zeitstempel des Ereignisses (s. <code>Date()</code>)
<code>e.type</code>	Die Art des Ereignisses
<code>e.keyCode</code>	Die Keyboard-Taste, die betätigt wurde
<code>e.data</code>	Die Daten, die für das aktuelle Ereignis übergeben werden, sofern vorhanden
...	...

A yellow starburst shape containing the text "jQuery".

Eventhandling

★ Eventhandling – Methoden von `$()`



vor Version 1.7
`bind/unbind`

★ `.on(typ, fn)`

★ Bindet einen Handler `fn` an den Event `typ`

★ `fn` bekommt ein Event-Objekt übergeben

★ `typ` können auch mehrere Events sein

★ Ist `$()` eine Kollektion, wird allen der Handler zugewiesen

★ `.on(typ, [data], fn)`

★ Optional Daten, die im Event-Objekt gespeichert werden

★ `.off(typ)`

★ Entfernt alle Eventbindungen vom Typ `typ`

Eventhandling

★ Beispiel:

```
<p id="p1">Text 1</p>
```

```
<p id="p2">Text 2</p>
```

```
$("#p1").on("click", function (e) {  
    window.alert("angeklickt");  
});
```

```
$("p").on("click", function (e) {  
    window.alert("angeklickt");  
});
```


Eventhandling

★ Eventhandling – Abbruch

- ★ `e.stopPropagation()` – verhindert Weitergabe des Ereignisses an folgende Observer
- ★ `e.preventDefault()` – verhindert die Defaultaktion, die mit dem Ereignis verbunden ist

★ Eventhandling – Shortcuts – Methoden von `$()`

★ `.ready(fn(e))` Sobald HTML-Struktur bereitsteht (vor `load`)

★ `.click(fn(e))` Bindet einen Handler `fn` an das `click`-Event

★ ...

Eventhandling

★ Event feuern – Methoden von `$()`

★ `.trigger(typ, arrPm)` – Triggert ein Event `typ` und bekommt einen Array mit weiteren Parametern

★ Beispiel:

```
<input type="checkbox" id="i2"/>  
$("#i2").trigger('click');
```

DOM Manipulation

★ Inhalt setzen – Methoden von `$()`

★ `.html()`

`.html(htmlString)`

`.append(inhalt)`

Holt bzw. ersetzt (hängt an) den HTML-Inhalt durch den übergebenen HTML-String

★ `.text()`

`.text(txt)`

Holt bzw. setzt den Textinhalt auf den übergebenen Textstring txt

★ `.empty()`

Entfernt den Inhalt (alle Kindknoten) aus den Elementen

DOM Manipulation

★ Beispiel

```
<div id="div1"></div>  
<div id="div2"></div>
```

```
$("#div1")  
    .html("<p>Dies ist HTML-Inhalt.</p>");  
$("#div2")  
    .text("Dies ist Text ohne <tags>.");
```

DOM Manipulation

★ Attribute setzen – Methoden von \$ ()

- ★ `.attr(name)`
`.attr(name, val)`
`.removeAttr(name)`

Holt bzw. setzt den Wert eines Attributs `name`

- ★ `.val()`
`.val(wert)`

Holt bzw. setzt den Wert eines Input-Elements

DOM Manipulation

★ Beispiel

```
<input type="checkbox" id="i2"/>
```

```
★ let i2 = $("#i2");  
  i2.attr("type", "text");  
  i2.val("Text");
```

oder

```
★ $("#i2").attr("type", "text")  
  .val("Text");
```

DOM Manipulation

★ Elemente un-/sichtbar – Methoden von \$()

★ `.hide()`

Versteckt HTML-Elemente

★ `.show()`

Zeigt HTML-Elemente

★ Beispiel

```
<input type="checkbox" id="i2"/>  
$("#i2").hide();
```

AJAX

★ Inhaltsmethoden von `$()`

- ★ Methode `.ajax(config)` stellt eine AJAX-Anfrage
- ★ Die Konfiguration geschieht mittels JSON
 - ★ `type`: `get` oder `post`
(andere nicht von allen Browsern unterstützt)
 - ★ `url`: Adresse und get-Parameter der Anfrage
 - ★ `dataType`: vom Server übermittelter Type (`xml`, `json`)
 - ★ `data`: Daten der Anfrage als Querystring oder JSON
 - ★ `success`: `function(data) {}`: Funktion, die ausgeführt werden soll, wenn Anfrage erfolgreich
 - ★ `error`: `function(...)`: Funktion, die ausgeführt werden soll, wenn Fehler aufgetreten

AJAX

★ Fortune-Beispiel:

```
<body>
  <div id="fortune" >
    <p><b>Spruch:</b></p>
    <div id="fortuneMessage"></div>
    <p><input type="button" value="Neuer Spruch"
      onclick="
        $.ajax( { url: 'fortune.php',
          success: function (data) {
            $('#fortuneMessage').html(data);
          } } );">
    </p>
  </div>
  <script src="jquery.min.js"></script>
</body>
```

AJAX

★ AJAX Convenience-Methoden

- ★ `.post()` Setzt einen post-Request ab
- ★ `.get()` Setzt einen get-Request ab
- ★ `.getJSON()` Setzt einen get-Request ab und erwartet JSON-encodierte Daten
- ★ `.getScript()` Setzt einen get-Request ab und führt das zurückgegebene JavaScript aus
- ★ `.load(url)` Lädt die mit `url` bezeichnete Ressource und fügt diese als HTML in das durch die Collection bestimmte Zielelement ein (*Achtung:* bis V1.7 kann `.load()` auch für Events genutzt werden)

AJAX

★ Fortune-Beispiel:

```
<body>
  <div id="fortune" >
    <p><b>Spruch:</b></p>
    <div id="fortuneMessage"></div>
    <p><input type="button" value="Neuer Spruch"
      onclick="
        $(' #fortuneMessage').load('fortune.php');
      ">
    </p>
  </div>
  <script src="jquery.min.js"></script>
</body>
```

AJAX

★ Serialisierung von Formulardaten

- ★ `.serialize()` - Serialisiert die Werte aus Formularelementen als URL-encodierten Textstring
- ★ `.serializeArray()` - Serialisiert die Werte aus Formularelementen als JavaScriptArray (für JSON)
- ★ Als Parameter für AJAX nutzbar

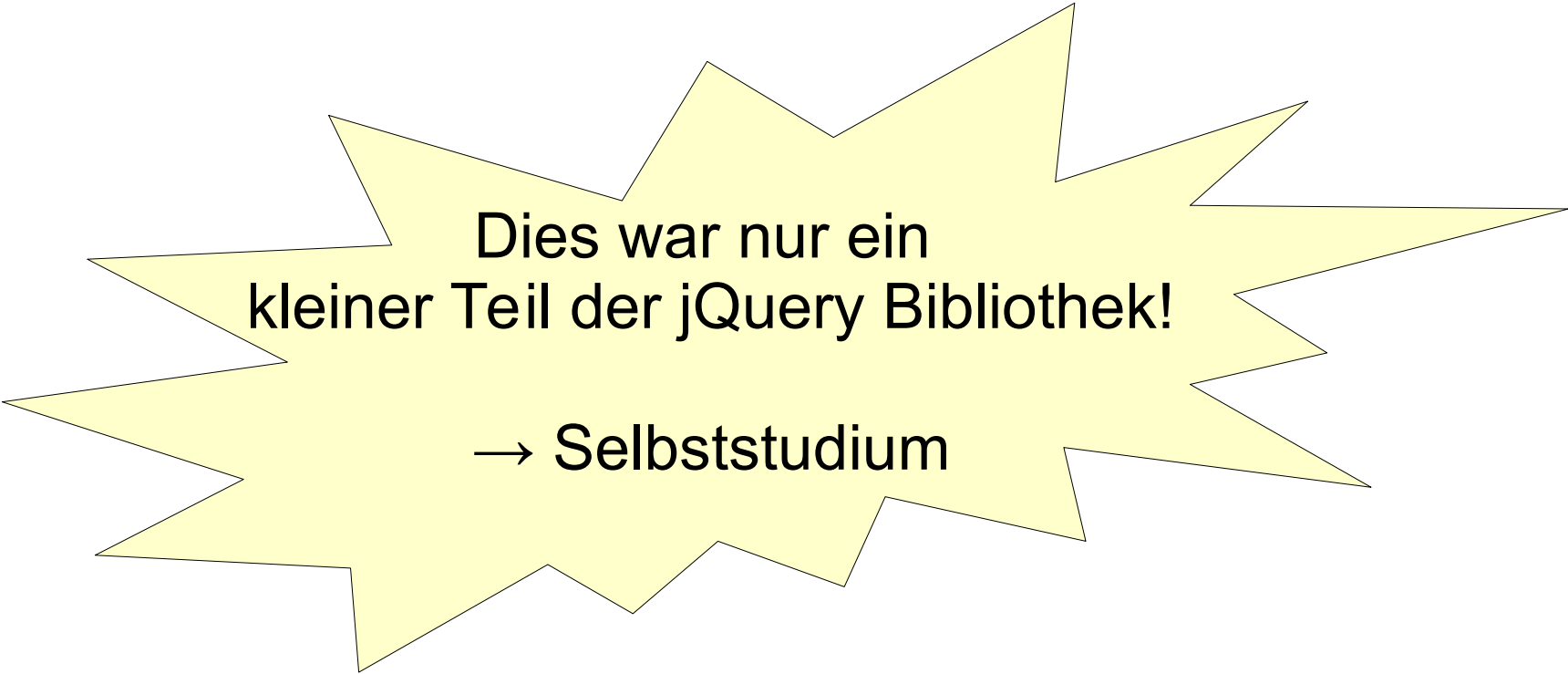
AJAX

★ Beispiel:

```
<div id="div2">
  <input type="checkbox" id="i1" name="i1"
        value="true" checked/>
  <label for="i1">I1</label>
  <input type="checkbox" id="i2" name="i2"
        value="true"/>
  <label for="i2">I1</label>
  <input type="text" id="i3" name="i3" value="Text"/>
</div>
```

```
console.log($('#div2 > *').serialize());
console.log($('#div2 > *').serializeArray());
```

AJAX



Dies war nur ein
kleiner Teil der jQuery Bibliothek!

→ Selbststudium

Literatur

- ★ Vallendorf, M. und Bongers, F.: jQuery – Das Praxisbuch, Galileo Computing, 2010
- ★ jQuery Project: jQuery, <http://jquery.com>
- ★ Flanagan, D.: JavaScript - The Definitive Guide, Auflage 5, O'Reilly, 2006
- ★ Wösten, André: jQuery – Das Training für interaktive und animierte Websites! Galileo Computing, 2011