



HTML 5 + JS + jQuery

Multimedia Engineering II WS 13/14

Sven Spielvogel





jQuery

- Selector-Engine → Basis
- Zahlreiche Hilfsfunktionalitäten (z.B. each)
- AJAX
- Effekte und Animationen
- DOM-Manipulation
- Umfassendes Event-Handling
- PlugIn-Schnittstelle



jQuery

- Nutzung von jQuery
 - Aufruf Option 1:
→ `jQuery(selector);`
 - Aufruf Option 2:
→ `$(selector);`

`function jQuery(parameter) { .. }`



äquivalent

`function $(parameter) { .. }`

jQuery - Einbinden

- Einfügen im Head:

```
<head>  
  <script src="jquery-1.10.2.min.js"></script>  
</head>
```

- Quellen:

- direkt – Download und ins Projekt einfügen
- jQuery-hosted - <http://code.jquery.com/jquery-1.10.2.min.js>
- Google-hosted - <http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js>

jQuery - Einbinden

- Versionen

- Sicher ist sicher – feste Version

`jquery-1.10.2.min.js`

- Immer neuster Stand – latest

`jquery-latest.min.js`

<http://code.jquery.com/jquery-latest.min.js> <1.10.2

<http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js>



jQuery - Einbinden

- Versionen
 - minified – komprimiert
`jquery-1.10.2.min.js`
 - developer – unkomprimiert
`jquery-1.10.2.js`





jQuery – Eine kurze Diskussion

- Warum/Wann latest?

- ..

- Warum developer?

- ..





jQuery – Ein cooles Feature

- Google JS-API ermöglicht eine dynamische Einbindung

```
<script type="text/javascript"
src="http://www.google.com/jsapi"></script>
```

```
<script type="text/javascript">
// jQuery laden
google.load("jquery", "1.10.2");
</script>
```

<https://developers.google.com/speed/libraries/devguide?hl=de-DE&csw=1>

- Welcher Mehrwert könnte sich daraus ergeben?





Funktionen

(ein kleiner Überblick)



jQuery – API

- API Documentation - <http://api.jquery.com/>
- W3schools.com - <http://www.w3schools.com/jquery/default.asp>
- jQuery-Syntax
 - `$ (selector) .action ()`
 - `$` ermöglicht den Zugriff auf die jQuery-Bibliothek
 - `selector` wählt ein oder mehrere HTML-Elemente
 - `action` führt die entsprechende Funktion aus



jQuery – API

- Wir schauen uns in den folgenden Kapiteln diese jQuery-Funktionalitäten genauer an
 - Events
 - DOM-Manipulation
 - CSS-Manipulation/Effekte
 - Zusatzfunktionen
 - AJAX
 - PlugIn





Events

jQuery – Events

- jQuery bietet eine breite Palette an möglichen Events, um die Event-Getriebene Entwicklung erheblich zu vereinfachen

- `$(document).ready();` → **Event**



```
$(document).ready(function() {  
    // do something now  
});
```

 → auszuführende Funktion

- Das `$(document).ready()`-Event wird erst „gefeuert“, wenn der komplette DOM geladen wurde. → JavaScript darf in den `<head>!`

jQuery – Events

- Fast alle Events nach dem gleichen Schema

```
$("#myButton").click(function(event) {  
    console.log(event); //jQuery-event  
});
```

Im Folgenden wollen wir uns typische Events genauer ansehen:

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	resize
dblclick	keydown	change	scroll
mouseenter	keyup	focus	
mouseleave		blur	



jQuery – Events

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	resize
dblclick	keydown	change	scroll
mouseenter	keyup	focus	
mouseleave		blur	



jQuery – Mouse Events

```
$("#myButton").click(function(event) {  
    console.log(event); //jQuery-event  
});
```

Click

Elements Resources Network Sources Timeline Profiles Audits Console

```
▼ b.Event {originalEvent: MouseEvent, type: "click", isDefaultPrevented: function, timeStamp: 1382181134020, jQuery19101697680598590523: true...} 1  
  altKey: false  
  bubbles: true  
  button: 0  
  buttons: undefined  
  cancelable: true  
  clientX: 22  
  clientY: 22  
  ctrlKey: false  
  ▶ currentTarget: button#myButton  
  data: null  
  ▶ delegateTarget: button#myButton  
  eventPhase: 2  
  fromElement: null  
  ▶ handleObj: Object  
  ▶ isDefaultPrevented: function ot(){return!1}  
  jQuery19101697680598590523: true  
  metaKey: false  
  offsetX: 12  
  offsetY: 12  
  ▶ originalEvent: MouseEvent  
    pageX: 22  
    pageY: 22  
    relatedTarget: null  
    screenX: 22  
    screenY: 107  
    shiftKey: false  
  ▶ target: button#myButton  
    timeStamp: 1382181134020  
  ▶ toElement: button#myButton  
    type: "click"  
  ▶ view: Window  
    which: 1  
  ▶ proto : Object
```

siehe „events-mouse-click.htm“



jQuery – Events

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	resize
dblclick	keydown	change	scroll
mouseenter	keyup	focus	
mouseleave		blur	



jQuery – Keyboard Events

```
$ (“#myInput”).keypress (function (event) {  
    console.log (event); //jQuery-event  
} ) ;
```

Elements Resources Network Sources Timeline Profiles Audits Console

```
▼ b.Event {originalEvent: KeyboardEvent, type: "keypress", isDefaultPrevented: function, timeStamp: 1382181605933, jQuery19106987918340601027: true...} 1  
  altKey: false  
  bubbles: true  
  cancelable: true  
  char: undefined  
  charCode: 115  
  ctrlKey: false  
  ▶ currentTarget: input#myInput  
  data: null  
  ▶ delegateTarget: input#myInput  
  eventPhase: 2  
  ▶ handleObj: Object  
  ▶ isDefaultPrevented: function ot(){return!1}  
  jQuery19106987918340601027: true  
  key: undefined  
  keyCode: 115  
  metaKey: false  
  ▶ originalEvent: KeyboardEvent  
  relatedTarget: undefined  
  shiftKey: false  
  ▶ target: input#myInput  
  timeStamp: 1382181605933  
  type: "keypress"  
  ▶ view: Window  
  which: 115  
  ▶ __proto__: Object
```

siehe „events-keyboard-keypress.htm“



jQuery – Events

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	resize
dblclick	keydown	change	scroll
mouseenter	keyup	focus	
mouseleave		blur	



jQuery – Form Events

```
$( "#myButton" ).submit (function (event) {  
    console.log (event); //jQuery-event  
} ) ;
```

Name:

Elements Resources Network Sources Timeline Profiles Audits Console

```
▼ b.Event {originalEvent: Event, type: "submit", isDefaultPrevented: function, timeStamp: 1382182449718, jQuery191009529726766049862: true...} ⓘ  
  altKey: undefined  
  bubbles: true  
  cancelable: true  
  ctrlKey: undefined  
  ▶ currentTarget: form  
  data: null  
  ▶ delegateTarget: form  
  eventPhase: 2  
  ▶ handleObj: Object  
  ▶ isDefaultPrevented: function it(){return!0}  
  jQuery191009529726766049862: true  
  metaKey: false  
  ▶ originalEvent: Event  
  relatedTarget: undefined  
  shiftKey: undefined  
  ▶ target: form  
  timeStamp: 1382182449718  
  type: "submit"  
  view: undefined  
  which: undefined  
  ▶ proto : Object  
> |
```

siehe „events-form-submit.htm“





jQuery – Events

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	resize
dblclick	keydown	change	scroll
mouseenter	keyup	focus	
mouseleave		blur	



jQuery – Document/Window Events

```
$(window).resize(function(event) {  
    console.log(event); //jQuery-event  
});
```

- **ACHTUNG:** Selektoren auf window/document ohne Anführungszeichen



jQuery – eventPhase

- Weiterreichen des Events - Propagation

```
3 <body>      2
    <div>      1
        <button id="myButton" value="mme2">Click</button>
    </div>
</body>
```

1 – eventPhase → 2

2+3 – eventPhase → 3

- Aufhalten des Bubbling-Verhalten durch
`event.stopPropagation();`

jQuery – Manuelles Event-Handling

- Auslösen eines Events
 - `trigger`
- Lauschen auf ein Event
 - `on/off`
 - `bind/unbind` seit 1.7 nur `on`
 - `delegate/undelegate`
 - `one` → einmaliges Lauschen

jQuery – Manuelles Event-Handling

siehe „events-manual.htm“

Sat Oct 19 2013 17:47:31 GMT+0200 (Mittleuropäische Sommerzeit)

```
Elements Resources Network Sources Timeline Profiles Audits Console x
▶ b.Event {type: "click", timeStamp: 1382197648261, jQuery19101365068145096302: true,
  isTrigger: true, namespace: ""...}
                                                                    events-manual.htm:21
▶ b.Event {originalEvent: MouseEvent, type: "click", isDefaultPrevented: function,
  timeStamp: 1382197651645, jQuery19101365068145096302: true...}
                                                                    events-manual.htm:21
>
```

jQuery – Events - Zusammenfassung

- Fast alle Events nach dem gleichen Schema

```
$("#myButton").click(function(event){});
```

```
$("#myButton").on("click",function(event){});
```

- Das im Kontext übergebene Event wird durch jQuery erzeugt, beinhaltet aber das originale Event
- Im jQuery-Event stecken zusätzliche Informationen
 - <http://api.jquery.com/category/events/event-object/>
- Aufhalten des Default-Verhalten durch `event.preventDefault()`;
 - Bsp: Überprüfung eines Formulars



DOM-Manipulation





jQuery – DOM Manipulation

- jQuery ermöglicht es, leicht und umfassend Anpassungen am DOM zur Laufzeit vorzunehmen
 - Klassen
 - Attribute/Werte
 - HTML



jQuery – DOM Manipulation

■ Klassen

- `addClass()` fügt ein oder mehrere Klassen hinzu

```
$("#myDiv").addClass("red");
```

- `removeClass()` entfernt ein oder mehrere Klassen

```
$("#myDiv").removeClass("red");
```

- `hasClass()` gibt zurück, ob mindestens ein Element diese Klasse besitzt

```
$("#myDiv").hasClass("red");
```



jQuery – DOM Manipulation

- Klassen

Dies ist ein DIV

```
Elements Resources Network Sources Timeline Pr
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <button value="red">Red</button>
    <button value="green">Green</button>
    <button value="clear">Clear</button>
    <div id="myDiv" class="red">
      Dies ist ein DIV
    </div>
    <script>...</script>
  </body>
</html>
```

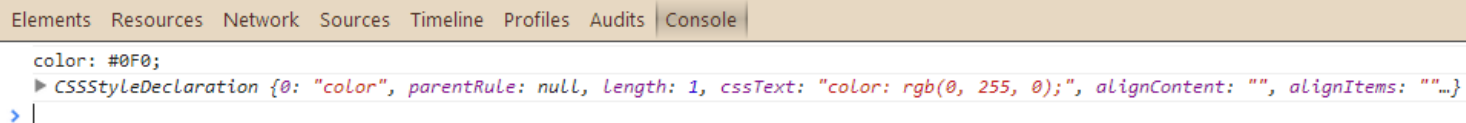
siehe „dom-class.htm“



jQuery – DOM Manipulation

- Attribute/Werte
 - `attr()` – HTML-Wert
 - `prop()` – DOM-Objekt-Wert
 - `val()` – äquivalent zu `prop("value")`

Output



The screenshot shows a web browser's developer console with the 'Console' tab selected. It displays a log entry for a `CSSStyleDeclaration` object. The object has a property `color` with the value `#0F0`. The console output is as follows:

```
color: #0F0;  
▶ CSSStyleDeclaration {0: "color", parentRule: null, length: 1, cssText: "color: rgb(0, 255, 0);", alignContent: "", alignItems: ""...}  
> |
```

siehe „dom-difference.htm“

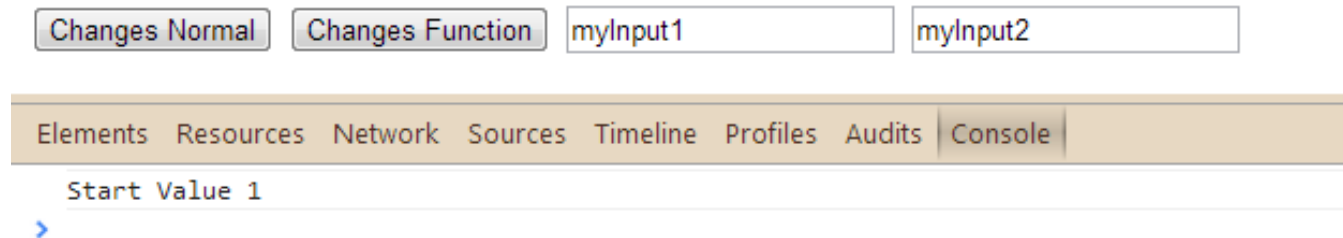


jQuery – DOM Manipulation

- Attribute können über jQuery ausgelesen und gesetzt werden. Ferner kann auch zur Verarbeitung eine Funktion verwendet werden
 - Lesen
 - `attr(attributeName)`
 - Setzen
 - `attr(attributeName, value)`
 - `attr(attributesObject)`
 - `attr(attributeName, function)`
- Vorteil: Cross-Browser-Kompabilität – Der Zugriff auf die Attribute funktioniert so in den meisten Browsern



jQuery – DOM Manipulation



siehe „dom-attr.htm“





jQuery – DOM Manipulation

■ HTML

- `html()` liest oder schreibt den HTML-Inhalt
- `append()` schreibt HTML-Inhalte innerhalb des DOM-Elements an das Ende
- `prepend()` schreibt HTML-Inhalte innerhalb des DOM-Elements an den Anfang
- `after()` schreibt HTML-Inhalte nach dem DOM-Element
- `before()` schreibt HTML-Inhalte vor dem DOM-Element
- `remove()` entfernt das DOM-Element
- `empty()` leert das DOM-Element



jQuery – DOM Manipulation

- HTML

`html()` `empty()` `append()` `prepend()` `before()` `after()` `remove()`

Before!

Prepend!

Hello World

Append!

After!

siehe „dom-html.htm“





CSS-Manipulation Effekte



jQuery – CSS Manipulation/Effekte

■ CSS

- `css()` liest oder schreibt CSS-Inhalte
- `$("p").css("background-color")` → lesen
- `$("p").css("background-color", "#000")` → schreiben
- Wird immer direkt ausgeführt!

■ Effekte – Fest definiert

- `show/hide/toggle`
- `fadeIn/fadeOut/fadeToggle/fadeTo`
- `slideUp/slideDown/slideToggle`

jQuery – CSS Manipulation/Effekte

- Aufruf: z.B. `.slide(speed, easing, callback)`

Parameter	Description
<i>speed</i>	Optional. Specifies the speed of the slide effectPossible values: <ul style="list-style-type: none">•milliseconds•"slow"•"fast"
<i>easing</i>	Optional. Specifies the speed of the element in different points of the animation. Default value is "swing" Possible values: <ul style="list-style-type: none">•"swing" - moves slower at the beginning/end, but faster in the middle•"linear" - moves in a constant speed
<i>callback</i>	Optional. A function to be executed after the method is completed

- Weitere Auflistung von Parametern:

<http://api.jquery.com/category/effects/>



jQuery – CSS Manipulation/Effekte

- Effekt-Funktionen
 - `delay()` Pause in Millisekunden
 - `finish()` Stoppt und beendet sofort alle Animationen in der Warteschlange
 - `stop()` Stoppt die aktuelle Animation
 - `queue()` Gibt die aktuelle Warteschlange zurück
 - `dequeue()` Holt das erste Element aus der Warteschlange und führt es dann aus
 - `clearQueue()` entfernt alle weiteren Elemente der Warteschlange. Laufende Animationen werden dabei nicht unterbrochen → `stop()`





jQuery – CSS Manipulation/Effekte

- Chaining: Als Chaining bezeichnet man das aneinanderreihen mehrerer Effekte
 - `$ ("div")`
 - `.css ("color", "red")`
 - `.delay (2000)`
 - `.toggle (2000)`
 - `.delay (1000)`
 - `.toggle (2000) ;`



jQuery – CSS Manipulation/Effekte

- CSS/Effekte

`css()` `chain` `stop()` `animate()` `Clear Queue and Stop`

Default Text

siehe „css.htm“





Zusatzfunktionen





jQuery – Zusatzfunktionen

- Folgende Zusatzfunktionen können immer wieder sehr nützlich sein
 - `each()` iteriert die gefundenen Elemente durch
 - `data()` erlaubt den Zugriff auf das Data-Objekt
 - `removeData()` löscht alle Inhalte im Data-Objekt
 - `get(i)` gibt das Element mit dem gewünschten Index zurück
 - `index()` gibt den Index des Elements in einer Vielzahl wieder (Liste)



jQuery – Zusatzfunktionen

each()

one
two
three
four
five

siehe „util-each.htm“

```
Elements Resources Network Sources Timeline Profiles Audits Console x
<div id="one">one</div> util-each.htm:18
<div id="two">two</div> util-each.htm:18
<div id="three">three</div> util-each.htm:18
<div id="four">four</div> util-each.htm:18
<div id="five">five</div> util-each.htm:18
> |
```

jQuery – Zusatzfunktionen

- Es gibt auch nützliche Funktionen, die ohne Selektor funktionieren
 - `$.each(array, function)` iteriert ein Array durch – Aber Achtung!! <http://jsperf.com/jquery-each-vs-foreach>
 - `$.grep(array, function)` filtert ein Array
 - `$.isArray(arg)` überprüft, ob Parameter ein Array ist
 - `$.now()` Unix-Timestamp → `new Date().getTime()`
 - `$.merge(arr1, arr2)` kombiniert mehrere Arrays
 - ...
 - <http://api.jquery.com/category/utilities/>



AJAX

AJAX

- Was ist AJAX?





AJAX

- Was ist AJAX?
- Asynchronous JavaScript And XML
 - Asynchroner Austausch von Daten zwischen Browser und Server
 - jQuery vereinfacht nur die „Schreibarbeit“



AJAX

■ Klassisch:

```
// compatible with IE7+, Firefox, Chrome, Opera, Safari
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function(){
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200){
        alert(xmlhttp.responseText);
    }
}
xmlhttp.open("GET", "test.php", true); // 3. Parameter → Async
xmlhttp.send();
```

AJAX

- jQuery:

```
$.ajax({  
    url: "test.php",  
    success: function(response) {  
        alert(response);  
    }  
});
```

- Noch kürzer:

```
$.get("test.php", function(response) {  
    alert(response);  
});
```



jQuery – AJAX - Methoden

- AJAX-Calls können auf verschiedene Arten mit jQuery durchgeführt werden:
 - `$.ajax` Klassischer AJAX-Call mit allen möglichen Parametern
 - `$.get` reines GET – verkürzt AJAX siehe „ajax/ajax.htm“
 - `$.post` reines POST – verkürzt AJAX
 - `$.getJSON` reines GET – verkürzt AJAX + Objekt zurück
 - `$.getScript` lädt JS und führt den Code aus
- <http://api.jquery.com/category/ajax/>



jQuery – AJAX - Methoden

- Neben den Parametern `success` und `error` gibt es auch noch weitere Methoden, die zur Verarbeitung der Ergebnisse helfen können:
 - `$.done` wird aufgerufen, wenn der Call erfolgreich war
 - `$.fail` wird aufgerufen, wenn der Call nicht erfolgreich war
 - `$.always` wird nach dem Call immer aufgerufen
- z.B. `success` hat folgende Parameter:
 - Ergebnis
 - Status
 - XHR-Objekt





Name	Value/Description
async	A Boolean value indicating whether the request should be handled asynchronous or not. Default is true
beforeSend(<i>xhr</i>)	A function to run before the request is sent
cache	A Boolean value indicating whether the browser should cache the requested pages. Default is true
complete(<i>xhr, status</i>)	A function to run when the request is finished (after success and error functions)
contentType	The content type used when sending data to the server. Default is: "application/x-www-form-urlencoded"
context	Specifies the "this" value for all AJAX related callback functions
data	Specifies data to be sent to the server
dataFilter(<i>data, type</i>)	A function used to handle the raw response data of the XMLHttpRequest
dataType	The data type expected of the server response.
error(<i>xhr, status, error</i>)	A function to run if the request fails.
global	A Boolean value specifying whether or not to trigger global AJAX event handles for the request. Default is true



Name	Value/Description
ifModified	A Boolean value specifying whether a request is only successful if the response has changed since the last request. Default is: false.
jsonp	A string overriding the callback function in a jsonp request
jsonpCallback	Specifies a name for the callback function in a jsonp request
password	Specifies a password to be used in an HTTP access authentication request.
processData	A Boolean value specifying whether or not data sent with the request should be transformed into a query string. Default is true
scriptCharset	Specifies the charset for the request
success(<i>result,status,xhr</i>)	A function to be run when the request succeeds
timeout	The local timeout (in milliseconds) for the request
traditional	A Boolean value specifying whether or not to use the traditional style of param serialization
type	Specifies the type of request. (GET or POST)
url	Specifies the URL to send the request to. Default is the current page
username	Specifies a username to be used in an HTTP access authentication request
xhr	A function used for creating the XMLHttpRequest object

Plug-In



Plug-In

- Plug-Ins in jQuery erweitern die Standard-Bibliothek um jede gewünschte Funktionalität
- Große Plug-Ins:
 - jQueryUI - <http://jqueryui.com/>
 - jQuery File Upload - <http://plugins.jquery.com/blueimp-file-upload/>
 - ...
 - <http://plugins.jquery.com/> ← Große Sammlung

Plug-In



[Demos](#) [Download](#) [API Documentation](#) [Themes](#) [Development](#) [Support](#) [Blog](#) [About](#)

Search

Interactions

- [Draggable](#)
- [Droppable](#)
- [Resizable](#)
- [Selectable](#)
- [Sortable](#)

Widgets

- [Accordion](#)
- [Autocomplete](#)
- [Button](#)
- [Datepicker](#)
- [Dialog](#)
- [Menu](#)
- [Progressbar](#)
- [Slider](#)
- [Spinner](#)
- [Tabs](#)
- [Tooltip](#)

Effects

jQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library. Whether you're building highly interactive web applications or you just need to add a date picker to a form control, jQuery UI is the perfect choice.

Download jQuery UI 1.10.3

Custom Download

Quick Downloads:

Stable

v1.10.3
jQuery 1.6+

Legacy

v1.9.2
jQuery 1.6+

What's New in jQuery UI 1.10?

jQuery UI 1.10 includes dozens of bug fixes and improved accessibility. In addition, the dialog and progressbar widgets have undergone [API redesigns](#), making them easier to use and creating more consistency across plugins.

Interested in the full details of what changed? Check out the [1.10 upgrade guide](#), [1.10.0 changelog](#), [1.10.1 changelog](#), and [1.10.2 changelog](#), and [1.10.3 changelog](#).

Dive In!

jQuery UI is built for designers and developers alike. We've designed all of our plugins to get you up and running quickly while being flexible enough to evolve with your needs and solve a plethora of use cases. If you're new to jQuery UI, check out our [getting started guide](#) and [other tutorials](#). Play around with the [demos](#) and read through the [API documentation](#) to get an idea of what's possible.

Stay informed about what's going on with jQuery UI by subscribing to our [blog](#) and following us on [Twitter](#).

Developer Links

[Source Code \(GitHub\)](#)
[jQuery UI Git \(WIP Build\)](#)
[Theme \(WIP Build\)](#)
[Bug Tracker](#)
[Submit a New Bug Report](#)
[Discussion Forum](#)
[Using jQuery UI](#)
[Developing jQuery UI](#)
[Development Planning Wiki](#)
[Roadmap](#)
[Previous Releases](#)
[Changelogs](#)
[Upgrade Guides](#)



Plug-In - Entwicklung

- Warum generell Plug-Ins entwickeln?
 - Was sind die Voraussetzungen?
- Warum könnte jQuery als Basis für ein Plug-In richtig sein?





Plug-In - Entwicklung

- Warum generell Plug-Ins entwickeln?
 - Was sind die Voraussetzungen?
 - Kleine immer wieder genutzte Funktionalitäten
 - Einfach zu implementieren in neue/bestehende Systeme
- Warum könnte jQuery als Basis für ein Plug-In richtig sein?
 - Verbreitete Standard-Bibliothek
 - jQuery kann Vorrausgesetzt werden inkl. Minimum Version





Plug-In - Entwicklung

- Warum generell Plug-Ins entwickeln?
 - Was sind die Voraussetzungen?
 - Kleine immer wieder genutzte Funktionalitäten
 - Einfach zu implementieren in neue/bestehende Systeme
- Warum könnte jQuery als Basis für ein Plug-In richtig sein?
 - Verbreitete Standard-Bibliothek
 - jQuery kann Vorrausgesetzt werden inkl. Minimum Version





Plug-In – Entwicklung in jQuery

- Einfachste Variante

```
$.fn.mmeplugin = function() {  
    this.html( "Hallo MME2" );  
};
```

siehe „plugin-basic.htm“

```
$( "#myDiv" ).mmeplugin();
```





Plug-In – Entwicklung in jQuery

- Um Chaining zu ermöglichen, müssen wir den Kontext zurück geben

```
$.fn.mmeplugin = function() {  
    this.html( "Hallo MME2" );  
    return this;  
};
```

```
$( "#myDiv" ).mmeplugin().css("background-  
color", "#000");
```



Plug-In – Entwicklung in jQuery

- Da der `$`-Kontext nicht immer gilt, schaffen wir den für unser Plug-In selber
- Das Semikolon `;` verhindert Fehler bei unsauberer Programmierung

```
;(function ( $ ) {  
    $.fn.mmeplugin = function() {  
        this.html( "Hallo MME2" );  
        return this;  
    };  
})( jQuery );
```

```
$( "#myDiv" ).mmeplugin().css("background-  
color", "#000");
```


Plug-In – Entwicklung in jQuery

- Um Gruppen zu bearbeiten, \$.each verwenden:

```
;(function ( $ ) {  
    $.fn.mmeplugin = function() {  
        return this.each(function() {  
            $(this).html( "Hallo MME2" );  
        });  
    };  
})( jQuery );
```

siehe „plugin-advanced.htm“

```
$( ".myDiv" ).mmeplugin();
```



Plug-In – Entwicklung in jQuery

- Parameter für alles Nötige
 - Auswahl der Aktionen
 - Optionen
 - Callback für Ergebnisse



Plug-In – Entwicklung in jQuery

```
;(function ( $ ) {  
    $.fn.mmeplugin = function(options){  
        options = $.extend( {action:"hallo",text:"Hallo MME2"},  
            options );  
        return this.each(function(){  
            switch(options.action){  
                case: "hallo"  
                    $(this).html(options.text);  
                break;  
                case: "bye"  
                    alert(options.text);  
                break;  
            }  
        });  
    };  
})( jQuery );  
$( ".myDiv" ).mmeplugin();  
$( ".myDiv" ).mmeplugin({action:"bye",text:"Bis bald!"});
```

Option

Aktion

Plug-In – Entwicklung in jQuery

```
;(function ( $ ) {  
    $.fn.mmeplugin = function(options){  
        options = $.extend( {x:1,y:2}, options );  
        return this.each(function(){  
            $(this).html(options.x+options.y);  
            if(typeof(options.fertig) == "function"){  
                options.fertig(this);  
            }  
        });  
    };  
})( jQuery );  
  
$( ".myDiv" ).mmeplugin({  
    x:2,  
    fertig: function(context){  
        console.log(context);  
    }  
});
```

siehe „plugin-callback.htm“



Plug-In – Entwicklung in jQuery

- Zusammenfassung
 - Plug-Ins können kurz und knapp entwickelt werden
 - Parameter erweitern das Plug-In wie benötigt
 - jQuery-Plug-In, wenn man mit dem DOM arbeiten möchte
 - komplexe Algorithmen können so auch verpackt werden, aber man sollte sich nicht von jQuery Abhängig machen, wenn nicht unbedingt nötig
- Guides, Boilerplate und Patterns auf
 - <http://jqueryboilerplate.com/>





Plug-In – Ideen für die Übung

- Boilerplate nutzen
- CSS in extra Dateien
- Möglicher Ablauf:
 - DIV zeichnen
 - Canvas zeichnen
 - DIV und Canvas füllen
 - Add als Parameter-Aktion
 - Move-Event über Callback





Vielen Dank für die
Aufmerksamkeit!

