Web Engineering + Design 1

JQUERY

Markus Stolze

Folien: Michael Gfeller & Markus Stolze

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Inhaltsverzeichnis

- jQuery Einleitung
- Ready vs. Load
- IIFE & noConflict
- jQuery
  - DOM Anfragen
  - DOM Manipulation
  - Style Manipulation
  - DOM Event Handling
  - Pitfalls

jQuery Einleitung

- **JQuery ist eine weit verbreitete JS Library (Open Source, MIT Lizenz) zur DOM Abfrage und Manipulation**
- **jQuery hat 2 aktuelle Versionen**
  - Version 1.*
    - Support für IE 6/7/8
    - Gleiche Funktionalität wie die 2er Version
  - Version 2.*
    - Kein (!) Support für IE 6/7/8
    - kleinere Grösse
    - Diese Version nutzen ausser es gibt einen guten Grund

jQuery Einleitung

- **jQuery definiert die globale Funktionen jQuery und den Alias $ sowie Methoden auf jQuery Result-Sets**
- **jQuery bringt ein alternatives DOM-API mit folgenden Vorteilen**
  - Kurze Schreibweise für DOM Manipulationen

```
//jQuery
$("em").hide();

//JavaScript
var emElmLst = document.querySelectorAll("em");
for (var i=0; i < emElmLst.length; i++){
    emElmLst[i].hidden=true;
}
```

  - Funktion-Chaining und Navigatoren
  - Einfach zu lernende (konsistente) Schreibweise
  - Cross-Browser Kompatibilität
    - Nicht mehr so relevant wie früher.
- **Und einige andere Features (z.B. AJAX)**

Wann nicht: jQuery

- **Falls nur ein sehr kleinen Teil von jQuery genutzt wird ist reines JavaScript sinnvoller**
  - Falls nur wenige DOM-Manipulationen getätigt werden
  - Falls nur eine Animation benötigt wird.
  - Falls nur der AJAX Befehl genutzt wird. => Nächste Woche
- **Beispiele für jQuery vs. JavaScript**
  - http://youmightnotneedjquery.com/

jQuery einbinden

- **Download unter: https://jquery.com/download/**
- **CND: https://jquery.com/download/#using-jquery-with-a-cdn**
- **Einbinden als lokales File**

```
<script src="jquery-2.2.3.min.js"></script>
```

- **Einbinden von einem CDN**

```
<script src="https://code.jquery.com/jquery-2.2.3.min.js"
        integrity="sha256-a23g1Nt4dtEYOj7bR+vTu7+T8VP13humZFBJNIYoEJo="></script>
```

Startpunkt

- **Einsatz von jQuery macht in den meisten Fälle erst Sinn nachdem das DOM bereit ist…**
- **… deshalb auf das DOM-Ready Event warten**
- **ready wird ausgeführt falls DOMContentLoaded ausgelöst wurde.**
  - Falls DOMContentLoaded schon gefeuert wurde – wird diese Funktion sofort aufgerufen!
    - Vorteil da normalerweise diese Funktion ignoriert werden würde.

```
$(document).on('ready', (function () {

    // code ...
    $("#container").text("Hello world");
});

//oder kürzer

$(function () {

    // code ...
     $("#container").text("Hello world");
});
```

addEventListener(…) vs. ready(…)

```
document.addEventListener("DOMContentLoaded", function(){
    console.log("DOMContentLoaded");
});
$(document).on('ready', function () {
    console.log("ready");
});

setTimeout(function(){
    document.addEventListener("DOMContentLoaded", function(){
        console.log("setTimeout - DOMContentLoaded");
    });
    $(document).ready(function () {
        console.log("setTimeout - ready");
    });
}, 10);
```

Output:

```
ready
DOMContentLoaded
setTimeout - ready
>
```

EINSCHUB: READY VS. LOAD

$.ready vs. DOMContentLoaded vs. onload vs. load

- **Load Events werden erst ausgelöst, wenn der gesammte Content geladen ist**
  - Bilder
  - Flash
  - Vorteil: Es ist ALLES geladen
  - Nachteil: Es muss ALLES geladen sein.

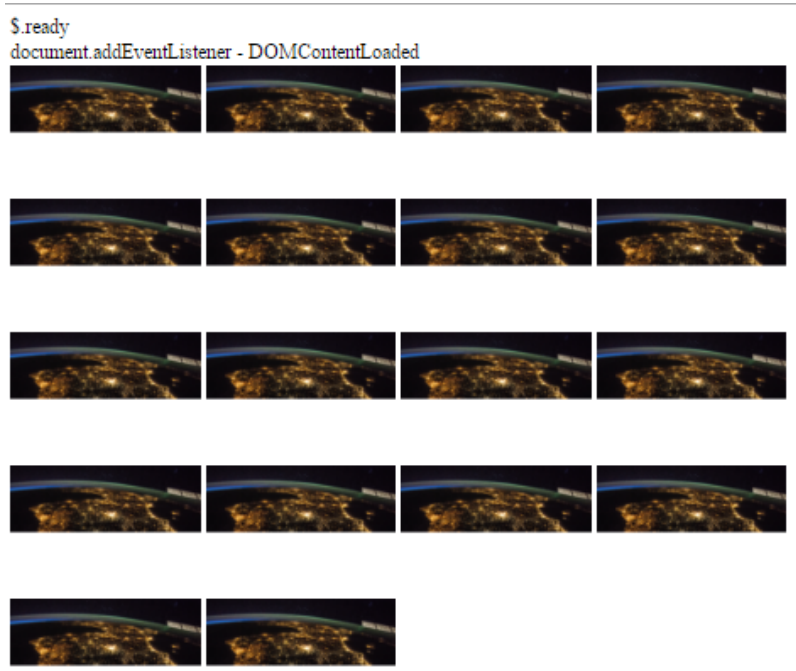
- **Ready Events**
  - Keine Garantie für das Bilder schon vorhanden sind.
  - Frühester Zeitpunkt welcher DOM Manipulationen möglich sind.
  - In den meisten Fällen die bessere Wahl.

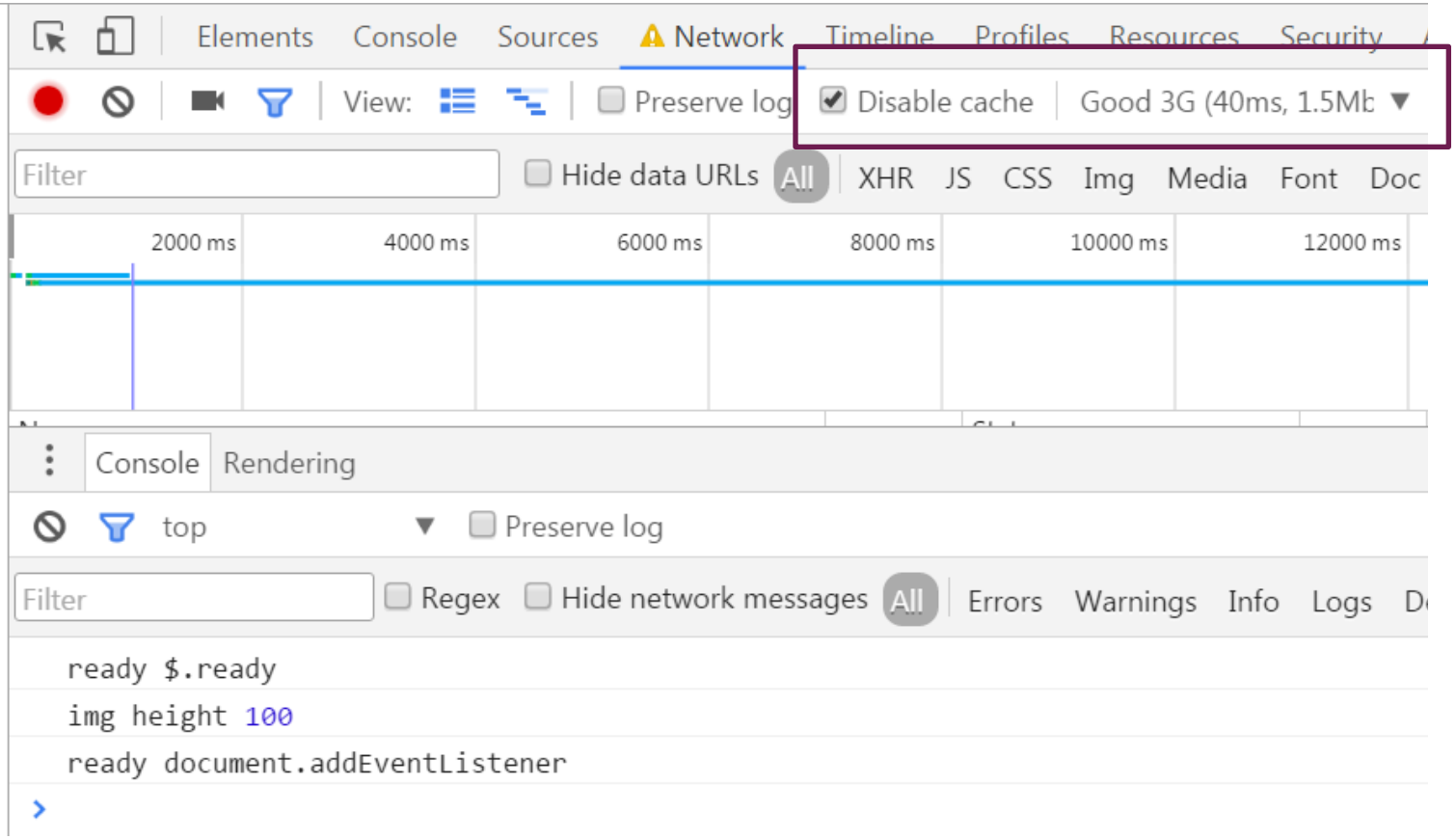
$.ready vs. DOMContentLoaded vs. onload vs. load

```
window.onload = function () {
    console.log("load", "window.onload");
};

$(window).on('load', (function(){
    console.log("load", "$.load");
}));

window.addEventListener('load', function () {
    console.log("load", "window.addEventListener");
    console.log("img height", $("img").first().height());
}, false);

$(function(){
    console.log("ready","$.ready");
    console.log("img height", $("img").first().height());
});

document.addEventListener('DOMContentLoaded', function () {
    console.log("ready", "document.addEventListener");
}, false);
```

```
ready $.ready
ready img height 0
ready document.addEventListener
load window.onload
load $.load
load window.addEventListener
load img height 100
```

$.ready vs. DOMContentLoaded vs. onload vs. load





IIFE & NOCONFLICT

IIFE

- **Immediately-invoked function expression**
  - https://en.wikipedia.org/wiki/Immediately-invoked_function_expression
- **Eine Funktion die sofort ausgeführt wird.**
- **Vorteile:**
  - Es wird ein neuen Scope defininiert
  - Variablen und Funktionen innerhalb eines IIFE sind nicht global.

```
(function() {

    // eigener Scope

})();
```

noConflict

- **Problem: Das $ Zeichen wird auch von anderen Frameworks genutzt.**
- **jQuery kann einen «noConflict» Modus aktivieren.**
  - $ wird wieder auf die alte Referenz gelegt.
  - jQuery ist nur noch über jQuery zugreifbar.
  - Es kann ein anderer Alias vergeben werden.

```
var $j = jQuery.noConflict();
```

- **Lösung mit IIFE**

```
(function($) {

    // $ ist jQuery

})(jQuery);
```

- **Details: https://learn.jquery.com/using-jquery-core/avoid-conflicts-other-libraries**
- **Lösung über JQuery Parameter Injection**

```
jQuery(function($j) {

    // $ ist jQuery
    console.log($("em"));

});
```

DOM ANFRAGEN

DOM Anfragen

- **jQuery erlaubt es wie bei CSS Selektoren zu definieren**
  - Hinweis: jQuery nutzt wenn möglich die native Funktion document.querySelectorAll(…)
- **Beispiele**

```
$("p > em");
$("#container");
$(".alert");
```

- Es ist auch möglich die Suche einzuschränken
  - Beispiel: Alle Elemente mit der Klasse die unter dem angegebenen jQuery Objekt liegen:

```
var container = $("#container");
$(".alert", container);
```

- Das Resultat aller Anfragen ist eine Liste von gefunden Elementen (JQuery Result-List)
  - Jedes Element ist ein jQuery Objekt mit allen jQuery Funktionen
    - Das native DOM Objekt erhält man mit `var domElm = $(".alert").get( index )` oder `$(".alert")[index]`
    - $(*domElm*) erzeugt aus einem nativen Element ein jQuery Objekt
  - .length gibt die Anzahl von gefunden Elementen zurück
  - Aktionen die auf der Result-List ausgeführt werden, werden auf alle Elemente in der Liste angewendet.
- Diese Resultat kann verwendet werden um weiter zu navigieren z.B. zum Parent / zu den Kindern.

Navigation

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>WED1 JQuery Demo</title>
    <script src="jquery-2.2.3.min.js"></script>
    <script>
        jQuery(function($) {
            console.log(1, $("p"));
            console.log(2, $("p").parent());
            console.log(3, $("p").prev());
            console.log(4, $("p").next());
            console.log(5, $("p").parent());
            console.log(6, $("p").children());
        })
    </script>
</head>
<body>
    <h1>DOM Navigation</h1>
    <p>
        <span>Hello</span>
        <span>World</span>
    </p>
    <img src="..." style="width:150px">
</body>
</html>
```

```
1 ▶ [p, prevObject: n.fn.init[1], context: document, selector: "p"]
2 ▶ [body, prevObject: n.fn.init[1], context: document]
3 ▶ [h1, prevObject: n.fn.init[1], context: document]
4 ▶ [img, prevObject: n.fn.init[1], context: document]
5 ▶ [body, prevObject: n.fn.init[1], context: document]
6 ▶ [span, span, prevObject: n.fn.init[1], context: document]
```

Navigation

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>WED1 JQuery Demo</title>
    <script src="jquery-2.2.3.min.js"></script>
    <script>
        jQuery(function($) {
            console.log(1, $("p"));
            console.log(2, $("p").parent());
            console.log(3, $("p").prev());
            console.log(4, $("p").next());
            console.log(5, $("p").parent());
            console.log(6, $("p").children());
        })
    </script>
</head>
<body>
    <h1>DOM Navigation</h1>
    <p>
        <span>Hello</span>
        <span>World</span>
    </p>
    <p>Hello World</p>
    <img src="..." style="width:150px">
</body>
</html>
```

```
1 ▶ [p, p, prevObject: n.fn.init[1], context: document, selector: "p"]
2 ▶ [body, prevObject: n.fn.init[2], context: document]
3 ▶ [h1, p, prevObject: n.fn.init[2], context: document]
4 ▶ [p, img, prevObject: n.fn.init[2], context: document]
5 ▶ [body, prevObject: n.fn.init[2], context: document]
6 ▶ [span, span, prevObject: n.fn.init[2], context: document]
```

- **jQuery führt für jedes Element der Result-List die Navigation aus.**
- **Erzeugt ein neues Liste mit allen (!) Resultaten**
- **Beispiel $("p").prev()**
  - Suche für alle p das vorhergehende Element.

Suchen und Filtern

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>WED1 JQuery Demo</title>
    <script src="jquery-2.2.3.min.js"></script>
    <script>
        jQuery(function($) {
            var items = $("#item-list > li");

            //color sollten in den CSS Klassen stehen!
            console.log("filter:\t.alert\t", items.filter(".alert").css("color", "green").length);
            console.log("filter:\t.info\t", items.filter(".info").css("color", "blue").length);

            var alertItems =  $("#item-list").find(".alert");
            console.log("find:\t.alert\t", alertItems.length);
        })
    </script>
</head>
<body>
    <h1>DOM Navigation</h1>
    <ul id="item-list">
        <li class="alert">Item 1</li>
        <li class="alert">Item 2</li>
        <li class="info">Item 3</li>
        <li class="alert">Item 4</li>
    </ul>
</body>
</html>
```

```
filter:  .alert   3
filter:  .info    1
find:    .alert   3
```

- **Listen können gefilteret werden**
  - filter()
  - not()
- **Unterbäume durchsucht werden**
  - find()

DOM MANIPULATIONEN

DOM Manipulationen

- **jQuery bietet einen einfachen Syntax zur Abfrage und Manipulation von DOM –Element Attributen**
  - `val()`
    - Abfragen und Setzen von Werten von Form-Elementen
    - Abfrage: Wert des ersten Elements in der Result-List; Setzen: Werte aller Elemente in der Result-List auf den Wert setzen.
  - `text()`
    - Abfragen und Setzen von Werten von nicht-Form-Elementen
    - Abfrage: Konkatenierte Texte aller Elemente in der Result-List (inkl. Sub-Elemente);
      Setzen: Texte aller Elemente in der Result-List auf den Wert setzen.
  - `html()`
    - Abfrage: HTML-Inhalt des ersten Elements; Setzen: HTML Inhalt jedes Elements in der Liste wird gesetzt.
  - `attr()`
    - Abfrage: Attribut-Wert des ersten Elements; Setzen: Attribut-Werte jedes Elements in der Liste wird gesetzt.
  - Andere nützliche Funktionen
    - **`hide(), remove(), append(), toogle()`**
- $ (“html“) -> neue Element Struktur
  - Beispiel: $(“<ul>”) => neues ul-element
  - Funktioniert auch mit ganzen HTML-Strukturen
    - $("
myItem
")

DOM Manipulation Beispiel

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>WED1 JQuery Demo</title>
    <script src="jquery-2.2.3.min.js"></script>
    <script>
        jQuery(function($) {
            var menuData = [{title : "Start", url : "/" },
                            {title : "Shop", url : "/shop"}];
            var menu = $("<ul/>", {"class" : "menu"});
            for( let item of menuData)
            {
                var link =  $( "<a/>", {
                    text: item.title ,
                    "class": "menu-content",
                    href: item.url
                });
                menu.append($("<li/>", {"class" : "menu-item"}).append(link));
            }
            // nur eine DOM manipulation notwendig!
            $("header").append(menu);
        })
    </script>
</head>
<body>
    <header></header>
</body>
</html>
```

```
▼<header>
  ▼<ul class="menu">
    ▼<li class="menu-item">
        <a class="menu-content" href="/">Start</a>
      </li>
    ▼<li class="menu-item">
        <a class="menu-content" href="/shop">Shop</a>
      </li>
    </ul>
  </header>
```

DOM Manipulationen: Function Chaining

- **Jede Abfrage von jQuery gibt eine Result-Liste zurück und alle Befehle werden meist auf alle Objekte angewendet. Diese Befehle geben in den meisten Fällen wieder eine Liste von jQuery Objekten zurück. Dies ermöglicht es beliebig viele Befehle aneinander zu reihen (Function Chaining)**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>WED1 JQuery Demo</title>
    <script src="jquery-2.2.3.min.js"></script>
    <script>
        jQuery(function($) {
            var list = $("#item-list");
            list.children().text("Hello World");
            list.find(".done").hide();
        });
    </script>
</head>
<body>
    <ul id="item-list">
        <li class="done">Item 1</li>
        <li class="done">Item 2</li>
        <li class="">Item 3</li>
        <li class="">Item 4</li>
    </ul>
</body>
</html>
```

```
▼<ul id="item-list">
    <li class="done" style="display: none;">Hello World</li>
    <li class="done" style="display: none;">Hello World</li>
    <li class>Hello World</li>
    <li class>Hello World</li>
  </ul>
```

STYLE MANIPULATIONEN

Style Manipulation

- **JQuery bietet einen einfachen Syntax zur Styling Manipulation und Animation**
  - Classen
    - addClass (name)
    - removeClass (name)
  - CSS Eigenschaften ändern
    - css(property-name, value)
    - Definiert diese Werte als Inline-Style
  - CSS-Animationen
    - fadeIn() / fadeOut() / fadeToggle()
    - slideUp() / slideDown
    - …

Beispiel

```
$("p").slideUp("slow").
       addClass("cool").slideDown("slow")
```

EVENT HANDLING

Event Handling

- jQuery stellt Funktionen zu Verfügung um typische Events zu registrieren; Diese Funktion sind Kurzschreibweisen der .on() Funktion
  - .click() oder besser .on(“click“, …)
  - .hover() oder besser .on(“hover“, …)
  - .focus() oder besser .on(“focus“, …)
- Die .on() Funktion ermöglicht es Event zu registrieren und verschiedene Optionen zu nutzen (z.B. Event-Bubbling zu nutzen)
  - $(body).on(eventsStr [, selector] [, data], handlerFn)
    - eventsStr: String mit Liste der Events (mit einem Leerzeichen getrennt)
    - selectorStr: cssSelectrString: nur für Elemente auf die der Selektor zutrifft wird die handlerFn aufgerufen
    - data: optionals (json) Object welches zum Zeitpunt des Event-Handliings in der handlerFn unter even.data genutzt werden kann (z.B. für index etc)
    - Handler: HandlerFn die aufgerufen wird – wenn derEvent ausgelösst wird.
  - Hier wird die handlerFn direkt auf dem Body Element registriert.
  - Alle Events welche auf den den Kinder-Elementen ausgelösst werden gelangen zum Parent-Element (Event-Bubbling)
    - Vorteil: Falls ein neues Child hinzugefügt wird ist die handlerFn auch für dieses Element aktiv! Selektion mit selectorStr
    - Wichtig für Template Engines => Nächste Woche
  - http://api.jquery.com/on/
- Die .off() Funktion kann die Registration wieder rückgängig machen (Referenz zur handlerFn ist notwendig)
  - http://api.jquery.com/off/

Event Handling

```
$("body").click(function(event){
    console.log("body - click " + event.target.nodeName);
    event.preventDefault();
});

$("body").on("click", "a", function(event){
    console.log("body with filter - click " + event.target.nodeName);
    event.preventDefault();
});

$("a").click(function(event){
    console.log("a - click " + event.target.nodeName);
    event.preventDefault();
});

var menuData = [{title : "Start", url : "/" }, {title : "Shop", url : "/shop" }];
for( let item of menuData)
{
    var link =  $( "<a/>", { text: item.title , href: item.url });
    var btn =  $( "<button/>", { text: item.title});
    $("header").append(link);
    $("header").append(btn);
}
```

- Was wird ausgegeben, falls auf einem Link geklickt wird?
- Was wird ausgegeben, falls auf ein Button geklickt wird?

JQUERY PITFALLS

Event Handling

- http://www.codeproject.com/Articles/346904/Common-Pitfalls-of-jQuery
- **Wichtige Punkte / Achtung bei**
  - Unnötige jQuery Abfragen statt Zwischenspeichern von Resultaten
  - Ineffziente Selektoren, z.B. #elmlds nicht genutzt um Suche einschränken
  - Shortcuts (wie hide, show, toggle, empty) nicht genutzt -> ineffizient & unleserlich
  - Repetitive Selektors

QUELLEN

Zum Nachschlagen

- https://learn.jquery.com/
- https://api.jquery.com/