Vorlesung

Webbasierte Informationssysteme

(CS4130)

jQuery, Ajax und Web Components

Professor Dr. rer. nat. habil. Sven Groppe
https://www.ifis.uni-luebeck.de/index.php?id=groppe

Chronologische Übersicht über die Themen

jQuery

- Veröffentlicht im Jahr 2006 von John Resig
- Open Source JavaScript Bibliothek
- Vereinfachung von
  - HTML-Navigation
  - Event-Handling
  - AJAX
  - Animationen

Gründe für den Einsatz von jQuery

- Eine der schnellsten JavaScript Bibliotheken
- Kompabilität
  - Cross-Browser
  - CSS3
- Achtung: teilweise nicht rückwärtskompatibel
- Umfangreicher Plugin-Support
- Lightweight footprint
  - 84,8 KB für Version 3.3.1 (minified, aber unkomprimiert)
- Meist verwendete JS-Bibliothek in über 70% der Webseiten*
  - Zweitplatzierter: Bootstrap mit ca. 15%

Offizieller Browser-Support*

Desktop			Mobile	
Internet Explorer	Chrome, Edge, Firefox, Safari	Opera	Android	iOS
9+	(*Current* - 1) und *Current*	*Current*	Stock browser on Android 4.0+	Safari on iOS 7+

Einbinden von jQuery

- Einbinden wie andere JavaScript-Dateien:

```
<script src="jquery-path" type="text/javascript"></script>
```

- Pfad zu der jQuery-Datei =
  - Pfad zu jQuery in Content Delivery Network (CDN): z.B. Google CDN
    oder:
  - lokale Datei auf dem eigenen Webserver nach Download von jQuery

Ausführen von JavaScript-Code nach dem Laden des Dokumentes

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

oder noch kürzer:

```
$(function(){
    // jQuery code goes here...
});
```

jQuery Syntax

- Grundschema bei Benutzung von jQuery:

```
$(selector).action()
```

  - `selector` folgt dabei der Syntax von CSS-Selektoren zur Adressierung von HTML-Elementen (+ zusätzliche jQuery-Selektoren)

- Beispiele von jQuery-Aktionen:

Aktionen	Beschreibung
`hide()`, `show()`, `toggle()`	Verstecken/Anzeigen/Wechselndes Anzeigen von HTML-Elementen
`fadeIn()`, `fadeOut()`, `fadeToggle()`	Einblenden/Ausblenden von HTML-Elementen
`slideDown()`, `slideUp()`, `slideToggle()`	Hereingleiten/Herausgleiten von HTML-Elementen
`click()`	Simulation eines Klicks auf ein HTML-Element

Beispiele von jQuery Selektoren

Selektor	selektiert:
`$("*")`	alle Elemente
`$(this)`	das momentane HTML Dokument/ Element (z.B. in Callback-Funktionen)
`$("p")`	alle `<p>`-Elemente
`$("#test")`	das Element mit Id `"test"` (Id-Selektor), z.B. `<span id="test"/>`
`$(".test")`	alle Elemente der Klasse `"test"` (Klassenselektor), z.B. `<div class="test">`
`$("p.intro")`	alle `<p class="intro">`-Elemente
`$("p:first")`	das erste `<p>`-Element
`$("ul li:first-child")`	das erste `<li>`-Element jedes `<ul>`-Elementes
`$("[href]")`	alle Elemente mit `href`-Attribut
`$("a[target='_blank']")`	alle `<a>`-Elemente mit `target`-Attribut mit Wert `"_blank"`
`$("tr:even")`	alle geraden `<tr>`-Elemente
`$(":α")`	alle `<input>`-Elemente vom Typ $\alpha \in$ {`text`, `passwort`, `radio`, `checkbox`, `submit`, `reset`, `button`, `image`, `file`}

Callback-Funktionen

- Optionale Parameter vieler Aktionen

```
$(selector).hide(speed, callback);
```

  - Geschwindigkeitsangabe
    - `"slow"` / `"fast"`
    - Angabe der Anzahl der Millisekunden
  - Callback-Funktion
    - Aufruf der Callback-Funktion nach Beendigung der Aktion

Beispiel:

```
$("p").hide("slow",function(){
  alert("The paragraph is now hidden");
});
```

Verkettungen von Aktionen

- Beispiel:

```
$("#p1")
  .css("color", "red")
  .slideUp(2000)
  .slideDown(2000);
```

  - Das Element mit Id "p1"
    - ändert zunächst seine Farbe auf rot,
    - wird dann herausgefahren und
    - zum Schluss wieder hereingefahren

Ereignisbehandlung

- Registrierung einer Callback-Funktion für das Eintreten eines Ereignisses:

```
$(selector).event(function(){
  // event handling goes here!!
});
```

  - Beispiel:

```
$("p").click(function(){ alert("Paragraph clicked!") });
```

- Ereignis auslösen:

```
$(selector).event();
```

  - Beispiel:

```
$("p").click();
```

Ereignisbehandlung – Wichtige Ereignisse

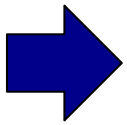
Mausereignisse	Tastatur-ereignisse	Formular-ereignisse	Dokument-/ Fensterereignisse
`click`	`keypress`	`submit`	`load`
`dblclick`	`keydown`	`change`	`resize`
`mouseenter`	`keyup`	`focus`	`scroll`
`mouseleave`		`blur`	`unload`

Ermitteln und Setzen der Werte von HTML-Elementen

Ermitteln		Setzen	
Aktion	**Beschreibung**	**Aktion**	**Beschreibung**
`text()`	Textinhalt eines selektierten Elementes	`text(t)`	Setzen des Textinhalts eines selektierten Elementes auf `t`
`html()`	Inhalt eines selektierten Elementes (mit HTML-Auszeichnung)	`html(t)`	Setzen des Inhalts eines selektierten Elementes (mit HTML-Auszeichnung) auf `t` Bsp.: `html("<b>Hello world!</b>")`
`val()`	Wert eines selektierten Formularfeldes	`val(v)`	Setzen des Werts eines selektierten Formularfeldes auf `v`
`attr(name)`	Wert des Attributes `name` eines selektierten Elementes	`attr(n,v)`	Setzen des Werts des Attributes `n` eines selektierten Elementes auf `v`

JS- zu jQuery umwandeln


```
<!DOCTYPE html>
<html>
 <head>
  <title>JavaScript Example</title>
  <script type="text/javascript">
   var number = Math.floor(Math.random()*6)+1;
   function check() {
    var guessed = document.GuessForm.inputfield.value;
    if(guessed==number){
     alert("Congratulations! The dice value is "
           + number + ".");
    } else {
     if(number<guessed) alert("My number is smaller!");
     else alert("My number is bigger!");
    }
   }
  </script>
 </head>
 <body>
  I have thrown a dice. Please guess the number:
  <form name="GuessForm">
   <input type="number" name="inputfield"
                size="1" min="1" max="6"/>
   <input type="button" value="Guess!" onClick="check()"/>
  </form>
 </body>
</html>
```



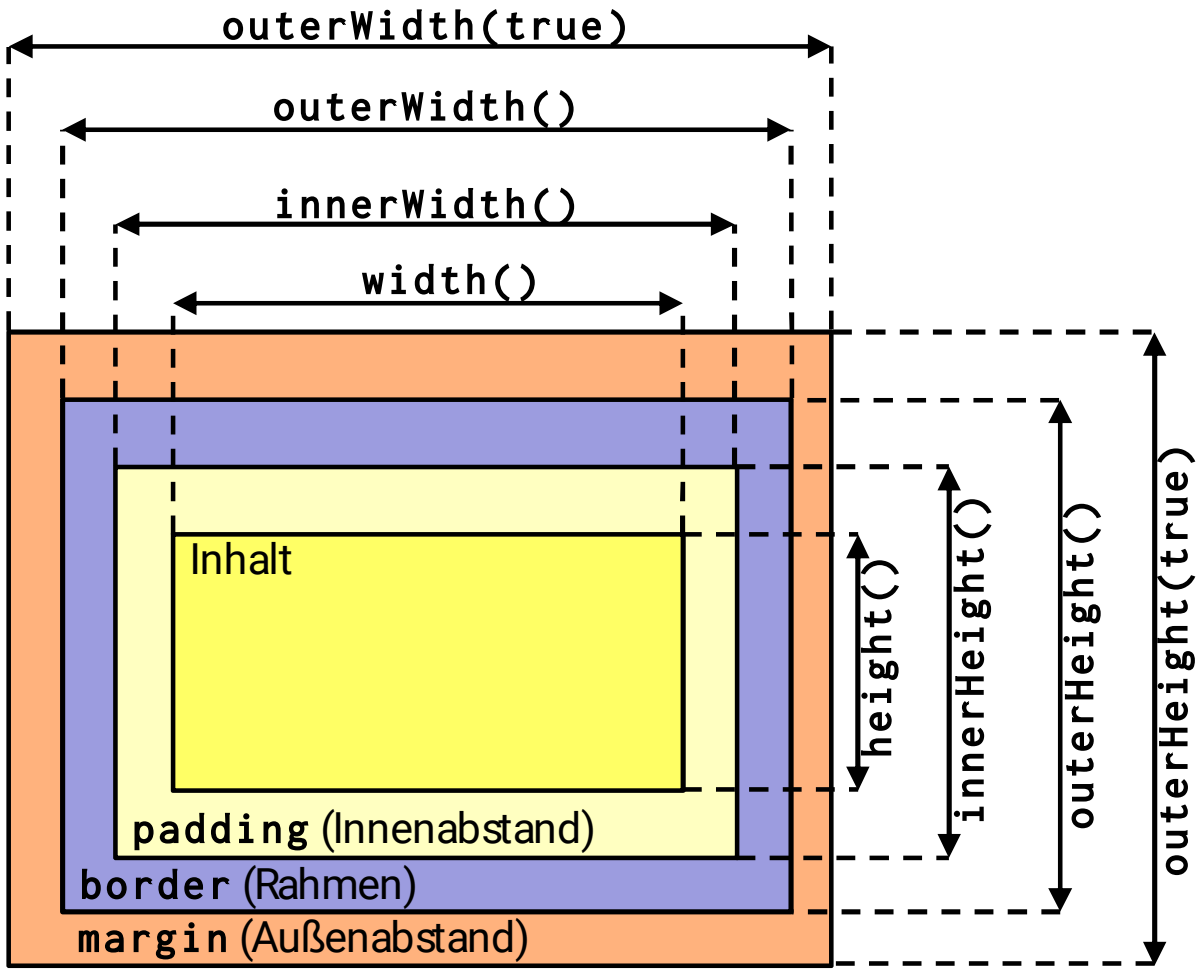
I have thrown a dice. Please guess the number:

Guess!

Manipulation des HTML-Dokumentes

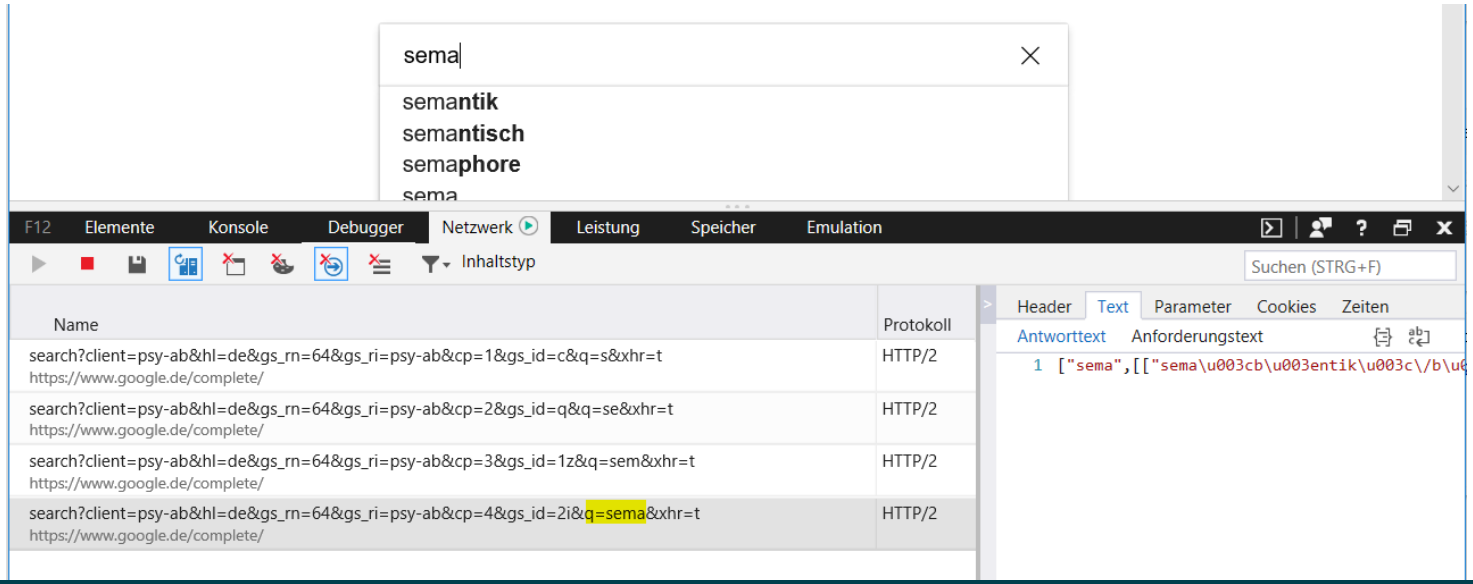
	Beispiele
Hinzufügen am Ende der selektierten Elemente	`$("p").append("Some appended text.");`
Entfernen der selektierten Elemente	`$("#div1").remove();`
Entfernen von Kindelementen	`$("#div1").empty();`
Setzen von CSS-Eigenschaften	`$("p").css("background-color","yellow");`

Box-Modell


outerWidth(true)
outerWidth()
innerWidth()
width()
Inhalt
padding (Innenabstand)
border (Rahmen)
margin (Außenabstand)
height()
innerHeight()
outerHeight()
outerHeight(true)

Asynchronous JavaScript and XML (AJAX)

- Verwendung von AJAX, um Daten im Hintergrund (⇝ asynchron) zu laden und auf der Webseite darzustellen, **ohne** die *gesamte* Seite erneut zu laden
  - **Vorteil:** Schnellere Darstellung und Bedienungsgefühl einer lokalen Anwendung, da Laden Benutzereingaben nicht blockiert



Asynchronous JavaScript and XML (AJAX)

- Beispielanwendungen:
  - Gmail
  - Google Maps
  - Youtube
  - Facebook tabs
- Einführung in AJAX-**jQuery**-Funktionen, *nicht* in die puren JS-Funktionen für AJAX
  - Cross-Browser-Support
  - Viel einfacher in der Handhabung

Grundlegende jQuery AJAX-Funktion

- `var jqXHR = $.ajax(settings);`
  - `settings` ist ein Objekt zur Konfiguration mit optionalen Eigenschaften

Eigenschaft	Bemerkung
`url`	URL der Anfrage
`method`	zu verwendende HTTP-Methode (z.B. `"POST"`, `"GET"`, `"PUT"`)
`data`	Daten, die für die Anfrage zum Server gesendet werden. Im Falle der HTTP-Methode `"GET"` werden die Daten in der URL kodiert.
	Viele weitere Eigenschaften, siehe jQuery Dokumentation

  - `jqXHR` ist ein `Promise`
    - Aufruf von via `done`/`fail` anzugebende Methoden(⇝ asynchron) nach Rückantwort vom Server
    - Falls die Rückantwort bereits vorliegt, werden die Funktionen sofort ausgeführt

Erfolgsfall	`jqXHR.done(function(data, textStatus, jqXHR) {…});`
Fehlerfall	`jqXHR.fail(function(jqXHR, textStatus, errorThrown) {…});`

Beispiel: jQuery AJAX-Funktion

```
$.ajax({
    url: "locationAwarePersonalizedInfo.php",
    method: "POST",
    data: { name: "Peter", location: "Lübeck" }
  })
  .done(function( data ) {
    alert( "Data returned: " + data );
  })
  .fail(function( jqXHR, textStatus ) {
    alert( "Request failed: " + textStatus );
  });
```

Short-Hand-Funktionen von jQuery

Anfrage	Short-Hand-Funktion	Äquivalenter Aufruf von `$.ajax`
HTTP Post	`$.post( u, d, s )`	`$.ajax({url: u, method: "POST", data: d}) .done(s);`
HTTP Get	`$.get( u, d, s )`	`$.ajax({url: u, method: "GET", data: d}) .done(s);`
HTTP Get von JSON-Datei	`$.getJSON(u, d, s)`	`$.ajax({dataType: "json", url: u, method: "GET", data: d}) .done(s);`
HTTP Get & Ersetze Dokumenten-inhalt	Lädt Daten unter der angegebenen URL `u` vom Server und setzt die Daten in das selektierte Element `sel`	
	`$(sel).load( u, d )`	`$.ajax({url: u, method: "GET", data: d}) .done(function(result){ $(sel).html(result); });`

`d` und `s` sind jeweils optional

Links zu JQuery

- JQuery Webpage
- Tutorials
  - W3Schools
  - Codecademy

Auswahl anderer bekannter Frameworks

- „jQuery Alternativen“: Mootools, Prototype.js
- Widget-Bibliotheken: Bootstrap, Dojo toolkit
- Responsive Webpages: Bootstrap, Pure, Material Design Lite
- script.aculo.us
  - Visual Effects Engine, Drag-and-Drop Bibliothek, Ajax-basiertes Autovervollständigen, …
- Angularjs
  - Clientseitiges Webframework für Single-Page-Web-Apps, die nur mit dem Server kommuniziert, um Teile seiner Seite zu ersetzen
- Ember.js
  - Als Schlüsselkonzept werden Templates (geschrieben in der mächtigen *handlebars templating language*) verwendet, die die Benutzerschnittstelle der Web App beschreiben
- Backbone.js
  - Schlüsselkonzept: Modell, Änderungen im Modell (im Client oder Server) werden automatisch im Client visualisiert ➡ Data-Rich Web Applications

Web Komponenten - Motivation

- Existierende Frameworks zur Webentwicklung
  - oftmals inkompatibel zueinander
  - oftmals Konflikte der Abhängigkeiten bei parallelem Einsatz
  - in jedes Framework neue Einarbeitung notwendig
  - nur für erfahrene und versierte JavaScript/CSS-Experten wirklich einsetzbar
- Web Komponenten als Überbegriff für viele verschiedene Technologien, die das Erstellen eigener, gekapselter Komponenten ermöglichen

Web Komponenten - Zentrale Technologie: Benutzerdefinierte HTML-Elemente

- ermöglicht das Registrieren & Verwenden eigener HTML-Tags
  - HTML Tags sind einfach zu verwenden – selbst für Nicht-Experten
  - (sehr) kurze Einarbeitungszeit
  - Kapselung ihres inneren Aufbaus (HTML, CSS, JS) und damit keine Konflikte/Inkompabilitäten mit anderen Frameworks/Scripts
  - einfach kombinierbar durch Schachtelung der benutzerdef. Tags
  - W3C entscheidet bisher über zu unterstützende Tags in W3C-Recommendations
    - Nützliche Tags brauchen sehr lange oder finden gar nicht ihren Weg in die W3C Recommendations
    - Selten, gar nicht verwendete oder überholte Tags müssen aus Kompabilitätsgründen in allen zukünftigen Browser-Versionen unterstützt werden
      - bläht Browser-Binaries auf, evtl. Performanceeinbußen

Web Komponenten - Beispiel

- Traditionelles Einbinden von Google Maps:

```
<script src="http://maps.googleapis.com/maps/api/js?key=APIKEY&sensor=true"/>
…
<script>
 new google.maps.Map(document.getElementById('Map'), {
    center: new google.maps.LatLng(-52.033, 8.533),
    zoom: 8,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  });
</script>
```

- Als Web-Komponente:

```
<link rel="import" href="google-map-plugin.html"/>
…
<google-map latitude="-52.033" longitude="8.533" zoom="8" type="roadmap">
</google-map>
```

Polymer

- Framework zur Unterstützung von Web-Komponenten in gängigen Browsern
- Alpha-Stadium, ständig starke Veränderungen in der API
- Architekur besteht aus folgenden Schichten:

Elements	• Grundlegende Menge von Elementen (z.B. AJAX, Animation, Layout) als Baukasten für Applikationen durch Kapselung von Browser APIs und CSS layouts
Core	• Notwendige Infrastruktur für benutzderfinierte HTML-Elemente (aufbauend auf den Native and Foundations-Schichten)
Foundations	• Polyfills, welche alle benötigten Features implementieren, die noch nicht nativ von den Browsern unterstützt werden • Diese Schicht soll mit steigender Unterstützung durch die Browser nach und nach verschwinden • u.a. Shadow DOM, HTML Imports, Custom Elements
Native	• Benötigte Features, die momentan in allen gängigen Browsern nativ verfügbar sind

Web-Komponenten –
Eigenbau eines Wetter-Tags*

Ergebnis: Das Wetter in **Lübeck**: 33 ° C, sonnig

Webseite:

```
…
<script src="platform.js">
</script>
<link rel="import"
      href="x-weather.html"/>
…
<x-weather city="Lübeck">
</x-weather>
```

x-weather.html:

```
<link rel="import" href="polymer.html">
<link rel="import" href="core-ajax.html">

<polymer-element name="x-weather" attributes="city">
  <template> Das Wetter in <strong>{{city}}</strong>:
            {{weather.main.temp}} ° C,
            {{weather.weather[0].description}}
    <core-ajax auto
      url="http://api.openweathermap.org/data/2.5/weather"
      params='{"q":"{{city}}", "mode":"json", ' +
             '"units": "metric", "lang": "de"}',
      handleAs="json"
      response="{{weather}}">
    </core-ajax>
  </template>

  <script> Polymer('x-weather', { city: 'Berlin' }); </script>
</polymer-element>
```

* Adaptiert von: Web Components erklärt, Teil 2

Web Komponenten – Status

- Von gängigen Browsern bereits durch Polymer unterstützt
- W3C arbeitet an Standardisierung (Web Platform Working Group)
- Viele Teile aus Foundations-Schicht bereits in Living Standards von WHATWG integriert*
  - z.B. Shadow DOM, Custom Elements, HTML Templates

Weitere Informationen zu Web Komponenten

- Status der Standardisierung
  - W3C
  - Auf Github
- WebComponents.org (Allg. Informationen)
- WebComponents.org (Galerie von Web-Komponenten)

Zusammenfassung

- Erleichterung der JavaScript-Programmierung durch Frameworks
  - Vereinfachte API für gängige Funktionen
  - Cross-Browser-Kompabilität
  - Umfangreiche Erweiterungen (z.B. Widget-Bibliotheken)
  - jQuery als beliebtes Basis-Framework
- Web Komponenten
  - Registrierung und Verwendung von benutzerdefinierten HTML-Tags
  - Einfachere Verwendung und bessere Kapselung
  - Bereits umfangreiche Sammlung von Web-Komponenten
  - Zukunft des Webs…