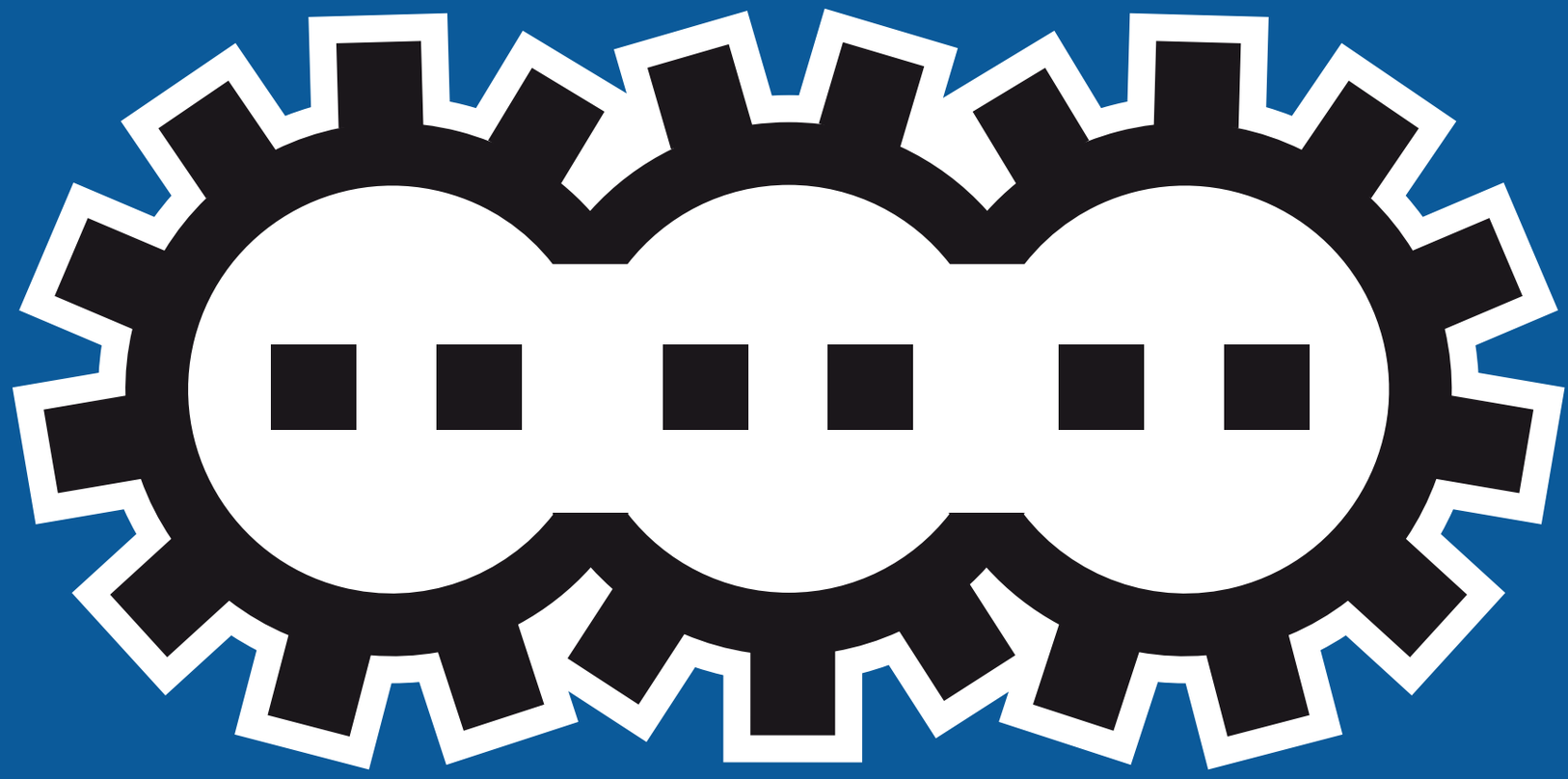
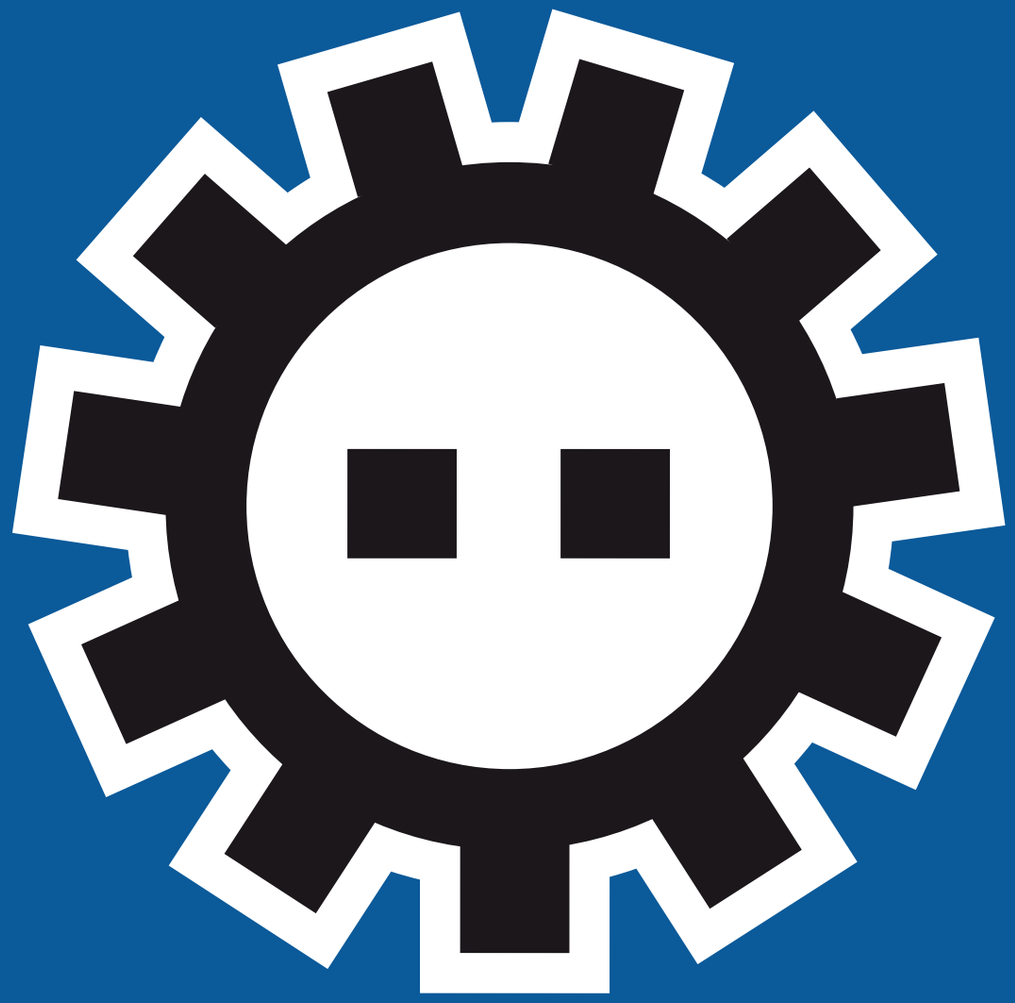


Software Engineering 2

06 - JavaScript

Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen



Motivation

Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen

Die dritte Komponente

JavaScript

Nutzen von JavaScript

- Zugriff auf das HTML-Dokument
- dynamische Änderung des HTML-Dokuments
- Reaktion auf Benutzerinteraktion
- Berechnungen / Programmverarbeitung
- Netzwerkverbindungen im Hintergrund

Was ist JavaScript?

- Eine im Browser integrierte Programmiersprache
  - Netscape 1995
  - bei Microsoft unter dem Namen JScript
  - von der ECMA 1997 standarisiert
- Enge Anbindung an HTML und CSS
- Objektorientiert (wird hier nicht weiter thematisiert)

Was ist JavaScript nicht?

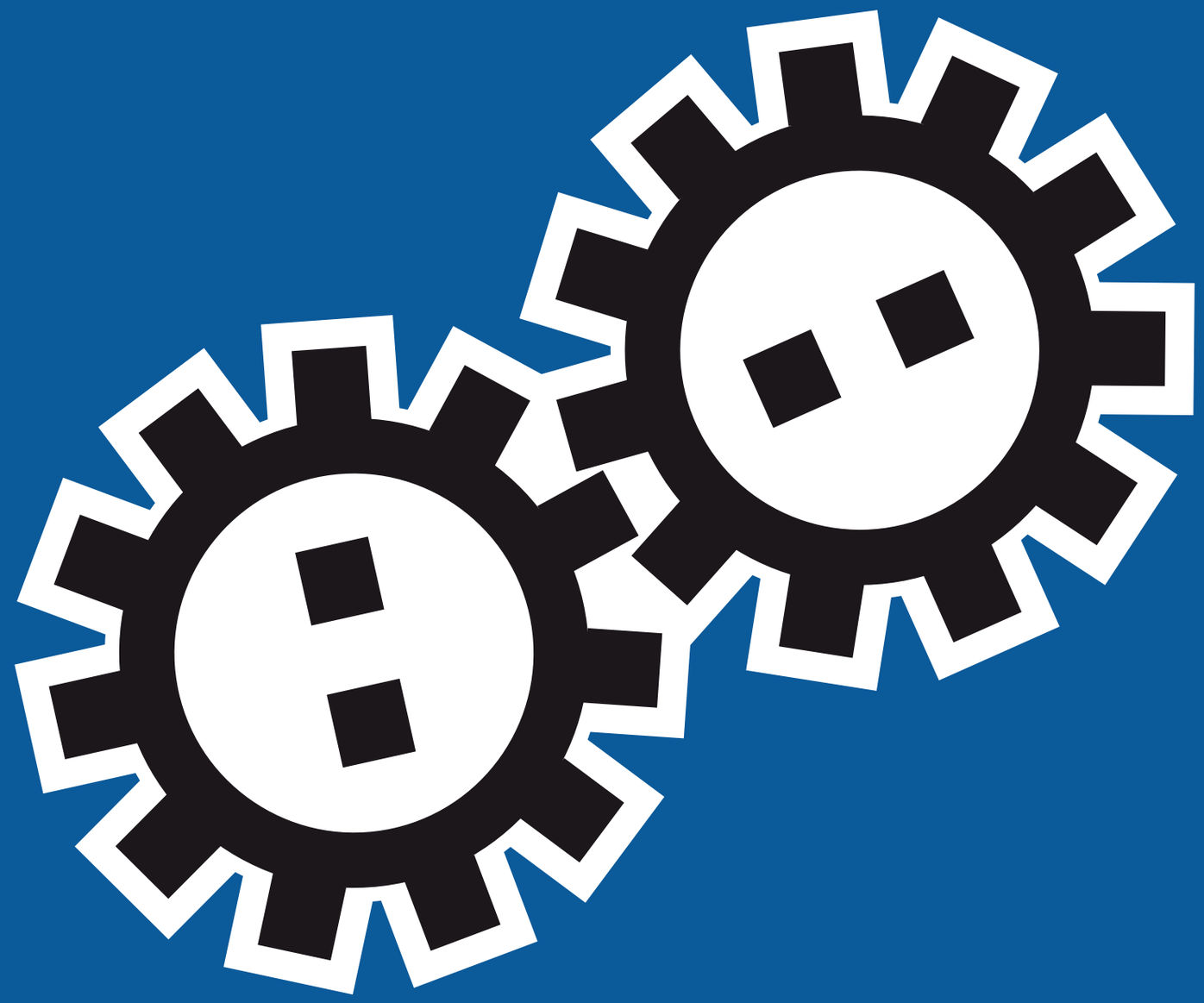
- Kein Java
  - klingt nur aus Marketinggründen so
  - Syntax ist ähnlich…
    - …Bibliotheken nicht
- Kein Script (im klassischen Sinne)
  - »für kleine, überschaubare Programmieraufgaben«

Einführendes Beispiel

```
<html>
   <body>
      <script>
        document.write("<h1>Hello World</h1>");
      </script>
   </body>
</html>
```

Hello World

- Innerhalb des Script-Elements steht das JavaScript
  - document.write…
- JavaScript kann (anders als CSS) auch im Body auftauchen
- JavaScript wird immer sofort ausgeführt



Techniken

Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen

Wie kann JavaScript in HTML verwendet werden

Einbindung von JavaScript

Einbindung (Inline)

- Im HTML-Dokument (Inline)
- Element "script" im HTML-Dokument
  - JavaScript-Programm als Inhalt dieses Elements

```
<html>
  <head>
    <script>
      alert("Hallo!");
    </script>
  </head>
  <body>
    <h1>Überschrift</h1>
    <p>Absatz</p>
  </body>
</html>
```

Einbindung (Extern)

- Als externe JavaScript-Datei
- Gleiches Element "script" im "head" der HTML-Seite
- Attribut "src" enthält URL der JavaScript-Datei

http://www.hs-furtwangen.de/web/kurs/seite.html

```
<html>
  <head>
    <script src="alert.js">
    </script>
  </head>
  <body>

  <h1>Überschrift</h1>
  <p>Absatz</p>

  </body>
</html>
```

http://www.hs-furtwangen.de/web/kurs/alert.js

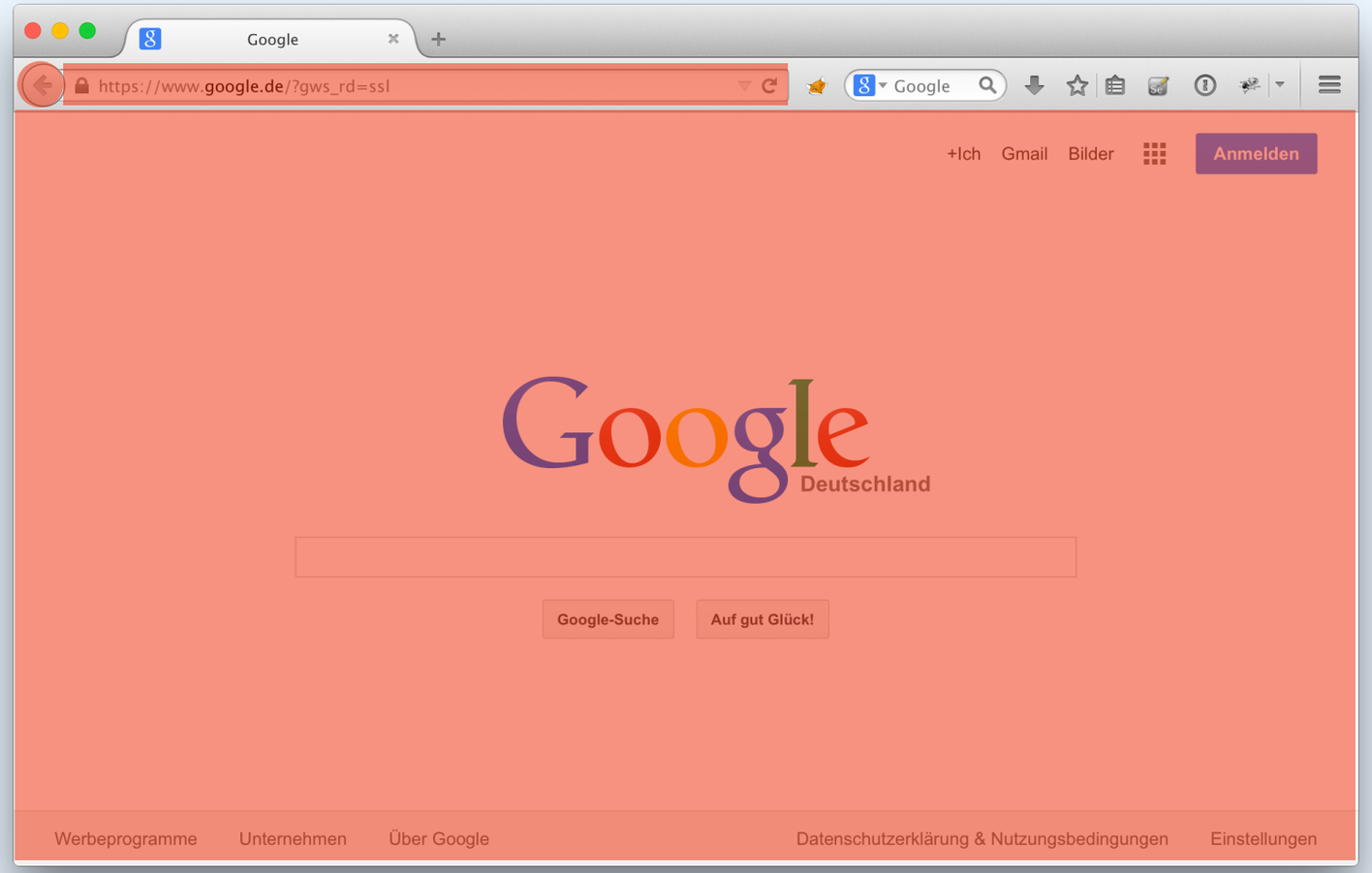
```
alert("Hallo!");
```

JavaScript in Verbindung mit HTML und CSS

JavaScript für das Web

JavaScript für das Web

- In JavaScript-Programmen kann über Variablen mit der Webseite kommuniziert werden
  - **location** → Adresse (URL) des Dokuments
  - **history** → Liste der besuchten Seiten
  - **document** → Dokument im Webbrowser



document

- zentrales Objekt, um in JavaScript auf die Webseite zuzugreifen
- Enthält zahlreiche Eigenschaften und Methoden
  - z.B.: document.write() um HTML auszugeben

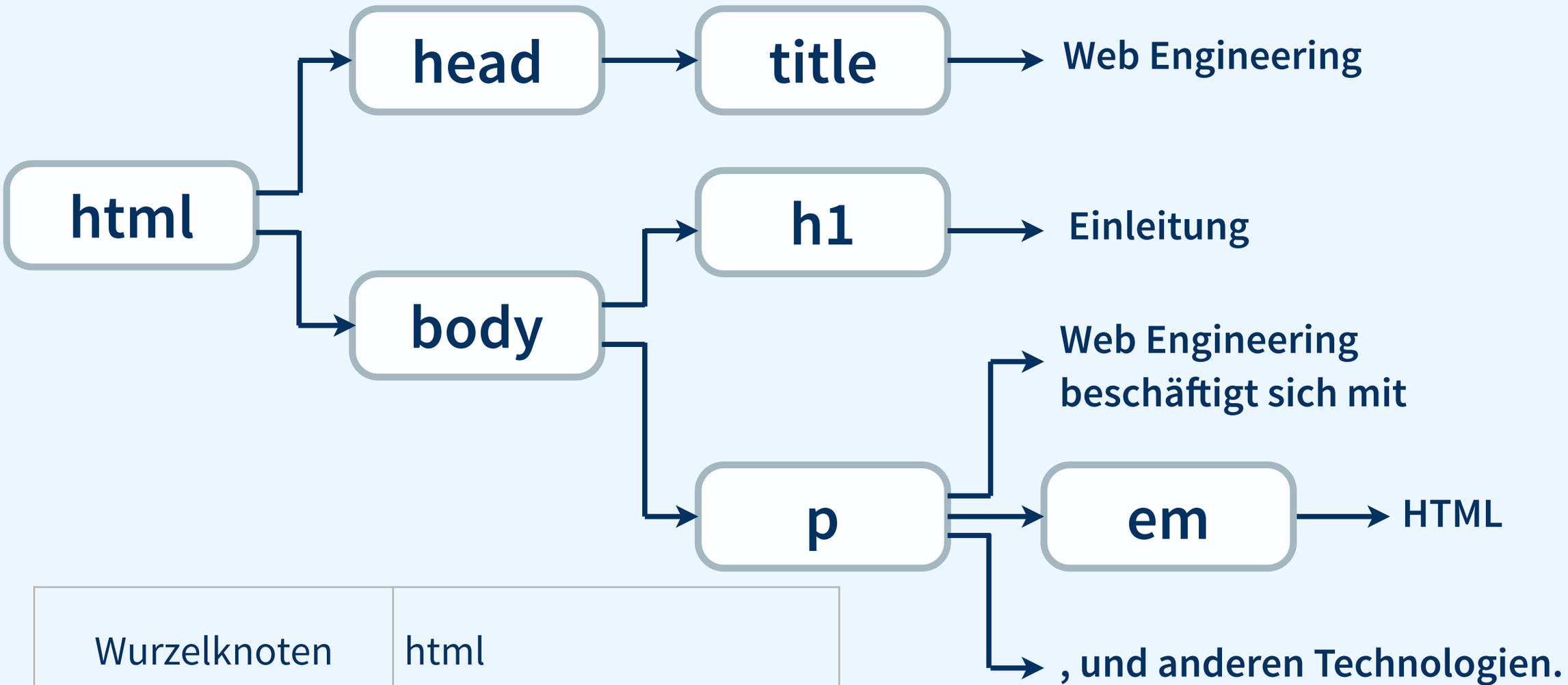
```
<html>
  <body>
    <h1>Uhrzeit</h1>
    <p>Die aktuelle Zeit:
      <script>
        var uhrzeit = new Date();
        document.write(uhrzeit.getHours());
        document.write(":");
        document.write(uhrzeit.getMinutes());
      </script>
    </p>
  </body>
</html>
```

Baum der HTML-Elemente

- Um die document-Variable auszureizen muss das Document Object Model(DOM) erläutert werden
- Der Webbrowser erzeugt einen Baum der HTML-Elemente

```
<html>
  <head>
    <title>Web Engineering</title>
  </head>
  <body>
    <h1>Einleitung</h1>
    <p>
      Web Engineering beschäftigt sich
      mit <em>HTML</em>, und anderen
      Technologien.
    </p>
  </body>
</html>
```

Baum der HTML-Elemente



Wurzelknoten	html
Elternknoten	head ist Elternknoten von title
Kinderknoten	h1 und p sind Kinderknoten von body
Geschwisterknoten	h1 ist Geschwister-knoten von p

Document Object Model

- Mittels document.documentElement kann der Wurzelknoten erreicht werden
  `var htmlNode = document.documentElement;`
- JavaScript erweitert jeden Knoten um Eigenschaften:
  - `node.childNodes` → Kinderknoten als Array
  - `node.parentNode` → Elternknoten
  - `node.nextSibling` → nächster Geschwisterknoten
  - `node.nodeName` → Name des Knotens (z.B. title)
  - `node.nodeValue` → Textinhalt des Knotens
  - `node.innerHTML` → Inhalt des Knotens als HTML

Document Object Model

- document.getElementById(id) findet ein Knoten anhand einer ID
- document.getElementsByTagName(tag) findet alle Knoten eines Typs, bspw. alle Absätze
- document.createElement(name) erstellt ein neues Element

auf Benutzereingaben reagieren

Browserereignisse

Browserereignisse

- Wir können Skripte nach dem Laden der Webseite ausführen
  - für interaktive Webseiten ist das zuwenig
- Aktionen des Benutzers müssen erfasst werden
  - Browser erzeugt Ereignisse für verschiedene Aktionen: Tastendruck, Mausklick, Scrollen
  - JavaScript-Programme können diese Ereignisse verwenden

DOM und Ereignisse

- An jedem Knoten des DOM (also an jedem Element) können für Ereignisse JavaScript-Funktionen bestimmt werden
  - Ereignishandler an Elementen
  - werden mittels (Universal-)Attributen definiert
- Es gibt viele Ereignisarten
  - z.B.: click → Ereignisattribut onclick="…"

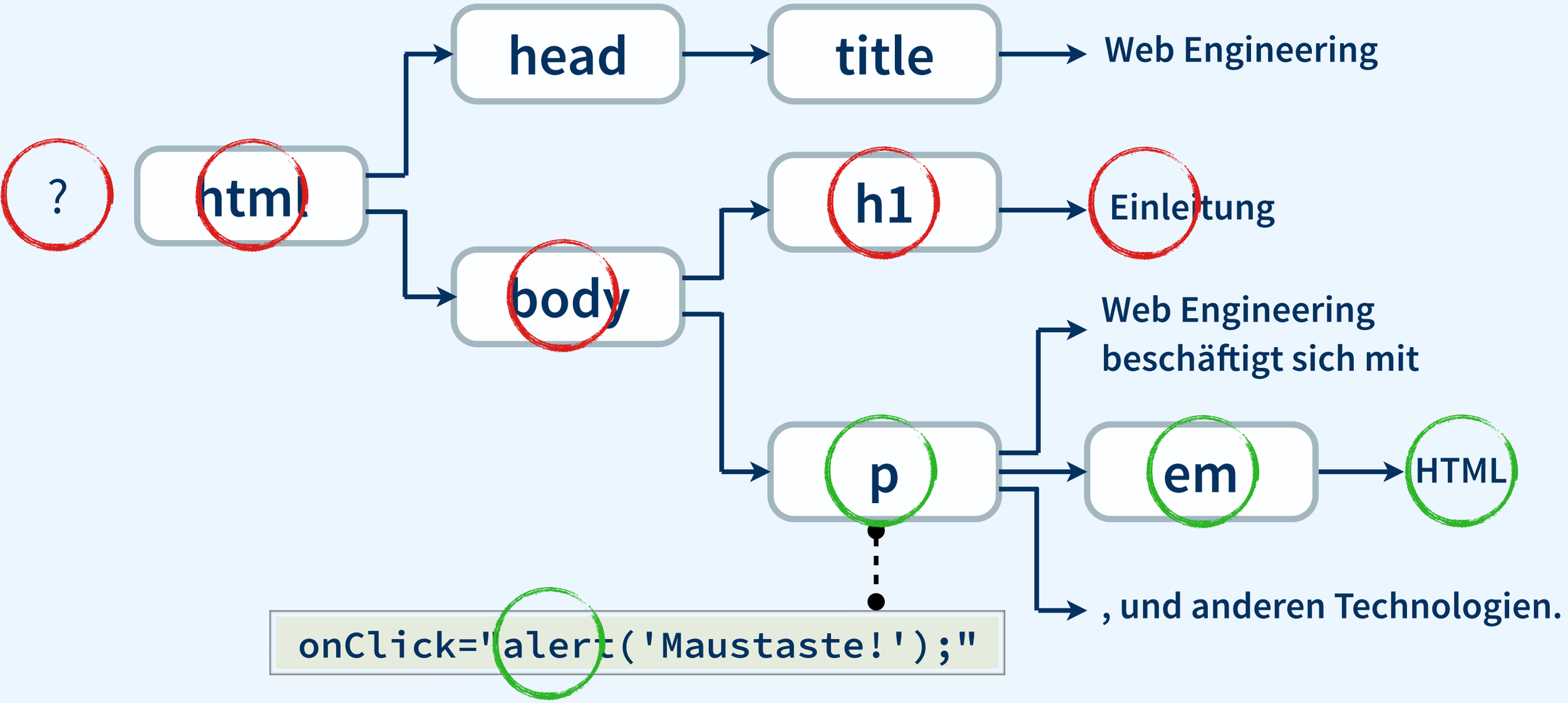
Ereignisarten

Ereignis-Attribut	Beschreibung
onclick	Ein Objekt wurde mit der Maus angeklickt
onmouseover	Der Mauszeiger wird über das Objekt bewegt
onmouseout	Der Mauszeiger wird von dem Objekt entfernt
Bei Formularfeldern	
onfocus	Ein Objekt bekommt den Eingabefokus
onblur	Ein Objekt verliert den Eingabefokus
onchange	Ein Objekt verliert den Eingabefokus und wurde verändert
onkeypress	Ein Objekt wurde verändert

Ereignishandler

- Ereignisse werden vom Browser in den Blättern des DOM-Baums erzeugt
- Der Browser geht den Baum die Elternknoten entlang, bis er einen Ereignishandler für das Ereignis findet
- Wenn kein Ereignishandler das Ereignis verarbeitet (nach Besuch des Wurzelknotens)
  - führt der Browser seinen eigenen Ereignishandler aus
    z.B.: Link verfolgen bei Klick auf einem Link

Ereignishandler



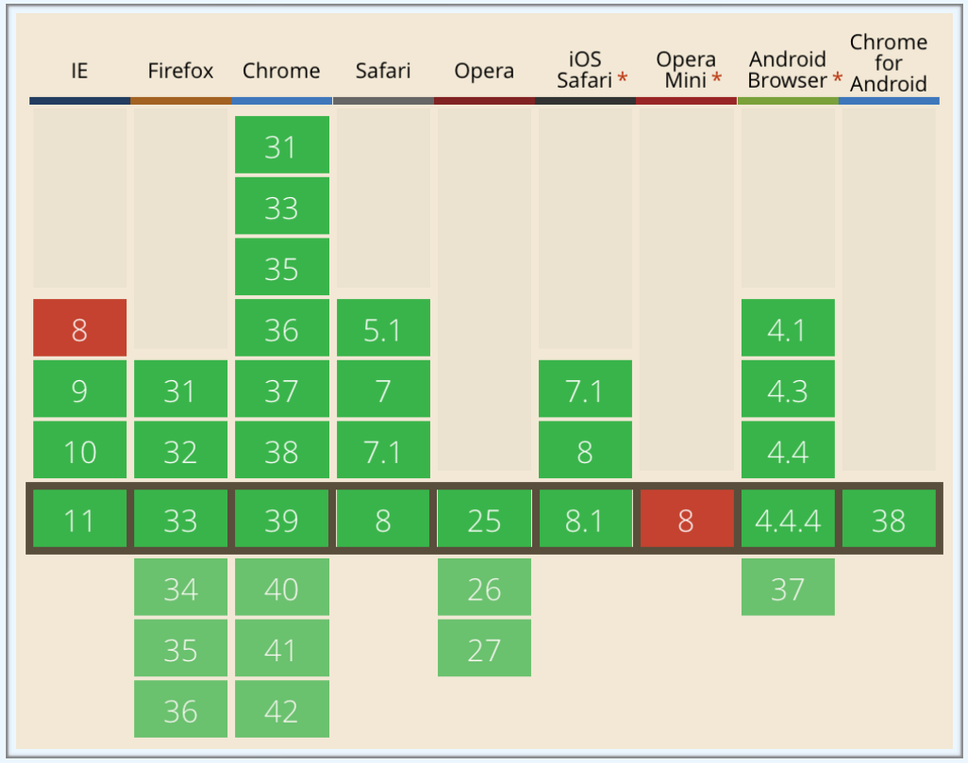
aktuellen Ort in die Webseite integrieren (Selbststudium)

JavaScript Beispiel

Geolocation

- Neues `geolocation`-Objekt mit drei Funktionen:
  - `getCurrentPosition()` → lokalisiere den aktuellen Ort
  - `watchPosition()` → starte ein dauerhaftes Lokalisieren
  - `clearWatch()` → beende ein dauerhaftes Lokalisieren

- Eine Lokalisierung beinhaltet:
  - Längen- und Breitengrad
  - Höhe
  - Geschwindigkeit
  - Richtung



Ausgabe in die Konsole

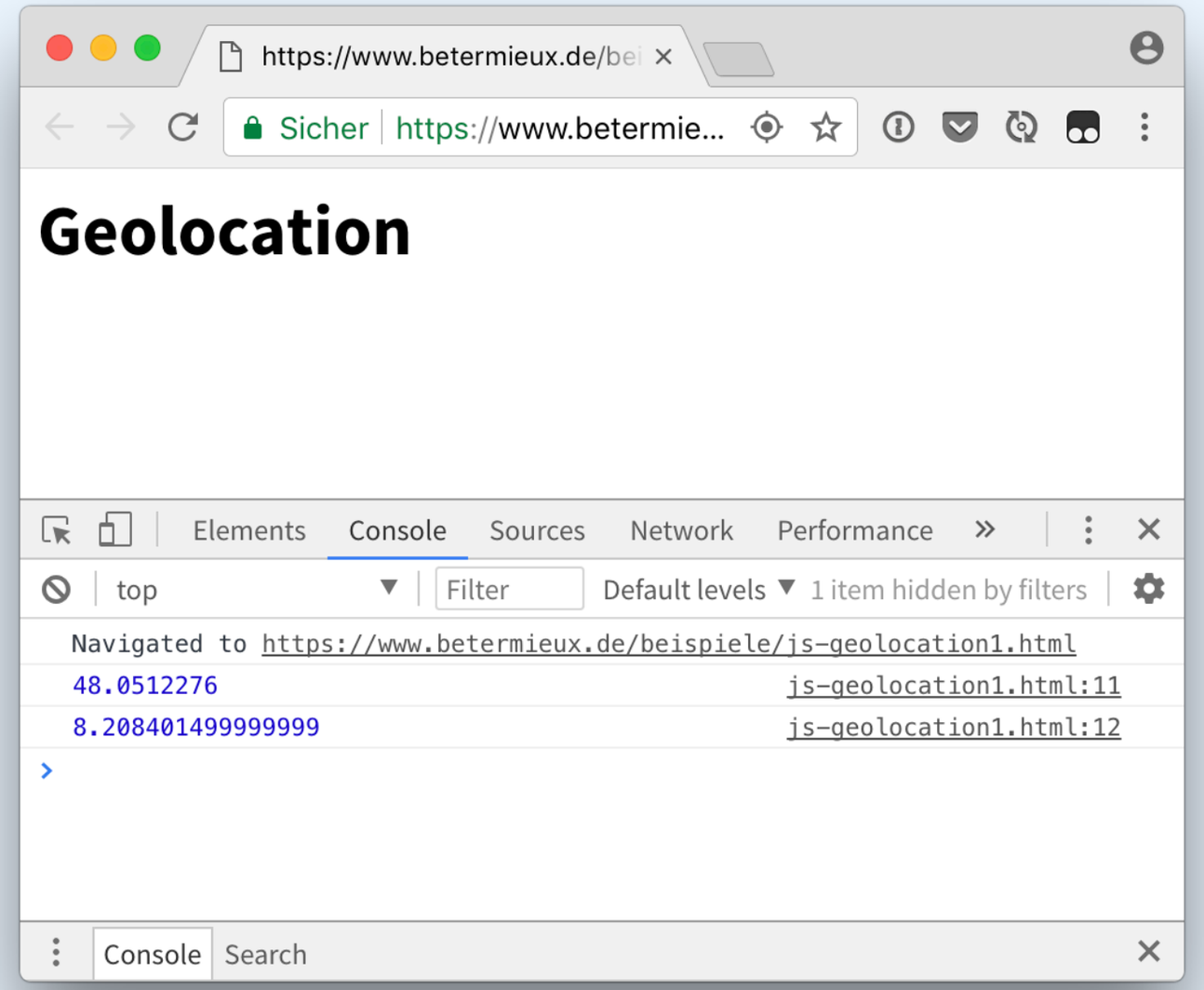
https://www.betermieux.de/beispiele/js-geolocation1.html

```
<html>
  <head>
    <script>
      function locateMe() {
        window.navigator.geolocation.getCurrentPosition(success, error);
      }
      // Funktion die bei erfolgreicher
      // Positionsbestimmung ausgeführt wird
      function success(position) {
        // "position" enthält die Geodaten
        console.log(position.coords.latitude);
        console.log(position.coords.longitude);
      }
      // Funktion die bei misslungener
      // Positionsbestimmung ausgeführt wird
      function error(msg) {
        console.log(msg);
      }
    </script>
  </head>
  <body>
    <h1>Geolocation</h1>
    <button onclick="locateMe()">Locate me!</button>
  </body>
</html>
```

Ausgabe in die Konsole

JavaScript-Konsole aus den Menüs öffnen:

- Chrome: weitere Tools/Entwicklertools
- Firefox:
  Web-Entwickler/Web-Konsole
- Safari:
  Entwickler/Fehlerkonsole
- …



Umwandlung in Ortsnamen

- Google bietet einfache Möglichkeit der Umwandlung von Längen-/ Breitengrad in vollständige Adresse
  - Eingabe → Breite 48.0512 Länge 8.2084
  - Ausgabe → Robert-Gerwig-Platz,
    78120 Furtwangen im Schwarzwald
    Deutschland
- Der Dienst kann einfach in JavaScript verwendet werden
- Ergebnisse können direkt abgefragt werden:
  `http://maps.googleapis.com/maps/api/geocode/json?sensor=false&latlng=48.0512276,8.20840149`
- Dokumentation findet sich unter:
  `https://developers.google.com/maps/documentation/geocoding/`

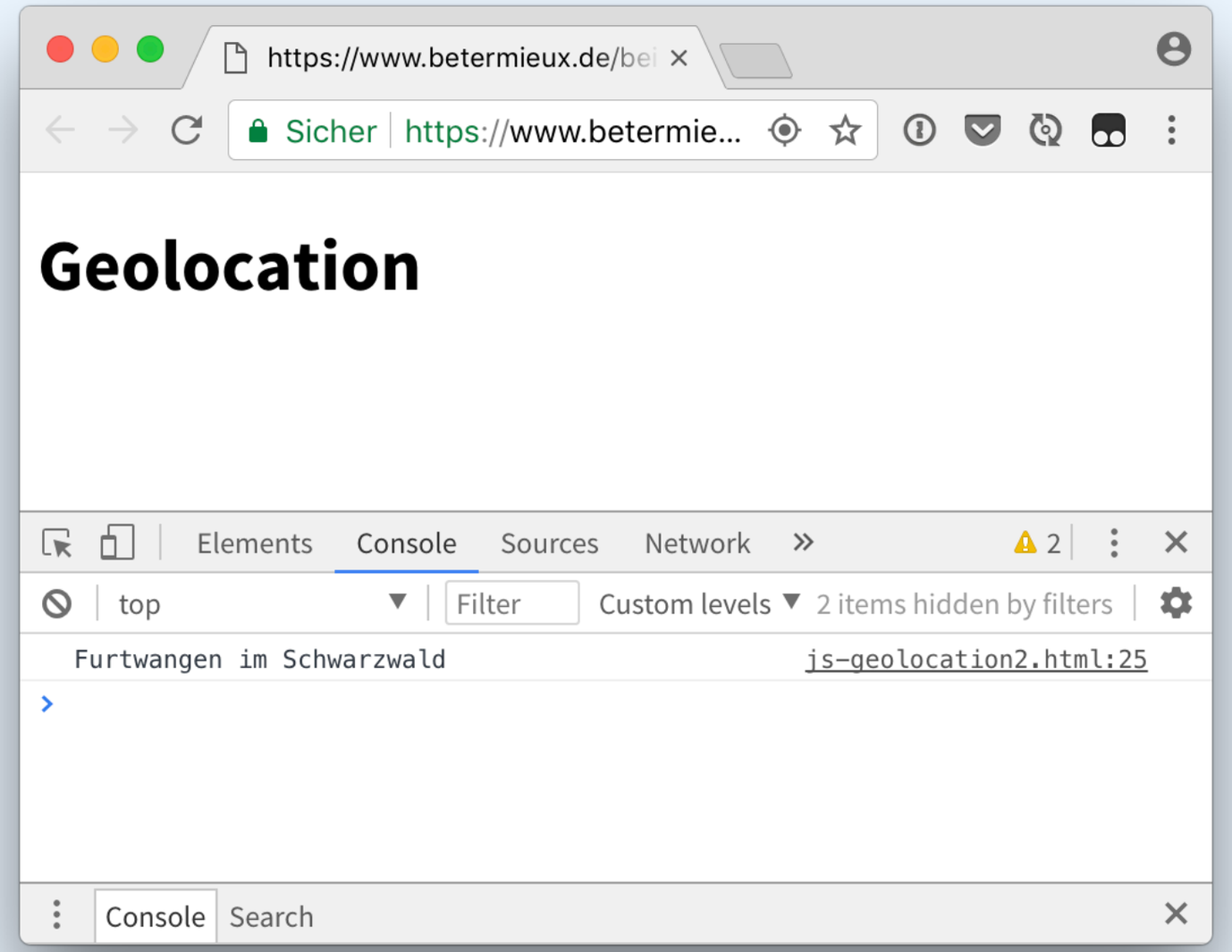
Umwandlung in Ortsnamen

https://www.betermieux.de/beispiele/js-geolocation2.html

```
<!DOCTYPE html>
<html>
  <head>
    <script>
      function locateMe() {
        window.navigator.geolocation.getCurrentPosition(geolocationSuccess);
      }
      function geolocationSuccess(position) {
        var coords = position.coords;
        var url = 'https://nominatim.openstreetmap.org/reverse?format=jsonv2&lat='+
coords.latitude +'&lon='+ coords.longitude;

        fetch(url).then(function(response) {
          return response.json();
        }).then(function(json) {
          var city = json.address.city;
          console.log(city);
        });
      }
    </script>
  </head>
  <body>
    <h1>Geolocation</h1>
    <button onclick="locateMe()">Locate me!</button>
  </body>
</html>
```

Umwandlung in Ortsnamen



Ausgabe der Daten

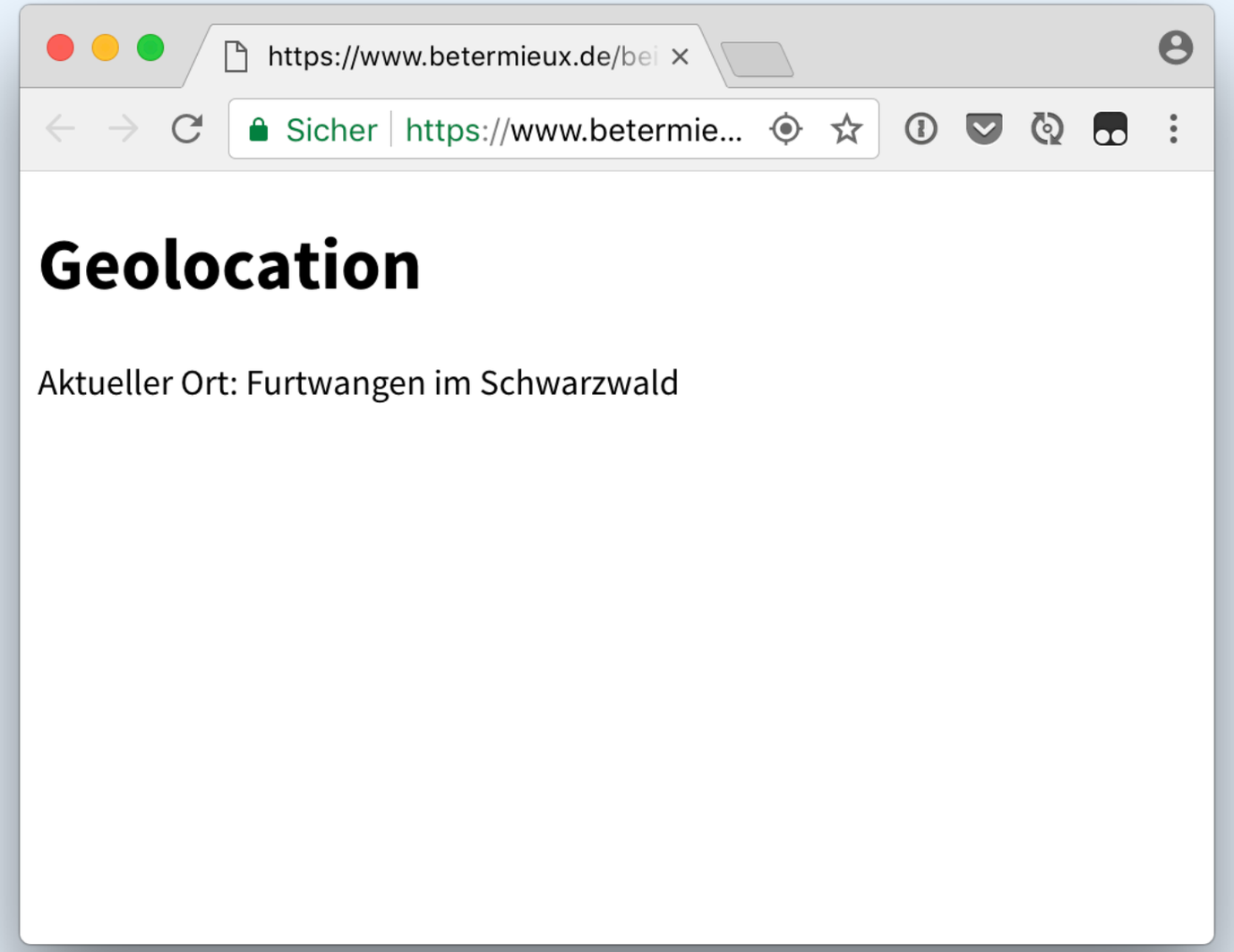
https://www.betermieux.de/beispiele/js-geolocation3.html

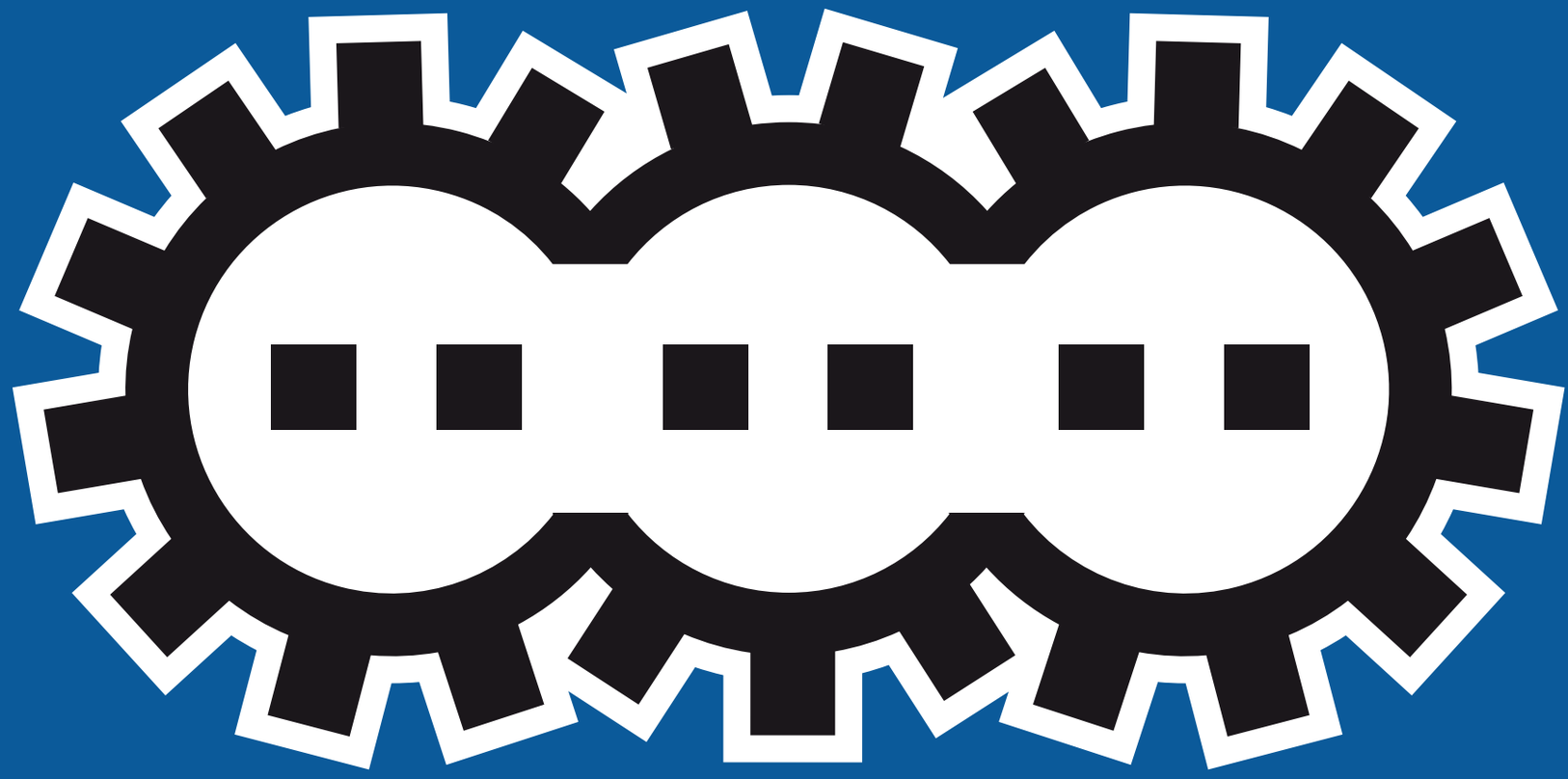
```
<!DOCTYPE html>
<html>
  <body>
    <h1>Geolocation</h1>
    <p>Aktueller Ort: <span id="cityElement">Planet Erde</span></p>
    <button onclick="locateMe()">Locate me!</button>

    <script>
     function locateMe() {
        window.navigator.geolocation.getCurrentPosition(geolocationSuccess);
      }
      function geolocationSuccess(position) {
        var coords = position.coords;
        var url = 'https://nominatim.openstreetmap.org/reverse?format=jsonv2&lat='+
coords.latitude +'&lon='+ coords.longitude;

        fetch(url).then(function(response) {
          return response.json();
        }).then(function(json) {
          var city = json.address.city;
          var cityElement = document.getElementById("cityElement");
          cityElement.innerHTML = city;
        });
      }
    </script>
  </body>
</html>
```

Ausgabe der Daten





JQuery

Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen

Motivation

Was können wir bisher?

- Wir können mit HTML und CSS schöne Seiten bauen!
  - Inhalt und Präsentation sauber getrennt
  - valide Dokumente
  - … aber (weitgehend) statisch
- Besucher möchten mehr Interaktion!
  - (dezente) Animationen und Transitionen
  - Eingabefelder mit Warnmeldungen
  - Automatisches Nachladen von Daten

JavaScript

- Vieles lässt sich mit JavaScript umsetzen!
  - …das meiste leider recht umständlich
  - z.B.: einen existierenden Textabsatz mit HTML füllen:
    ```
    document.getElementsByTagName("p")[0]
    .innerHTML = "neuer Absatz";
    ```
  - oder noch schlimmer, alle Textabsätze neu füllen:
    ```
    for (i = 0; i <= 4; i++) {
      document.getElementsByTagName("p")[i]
      .innerHTML="neuer Absatz";
    }
    ```
  - sieht nicht sehr elegant aus.

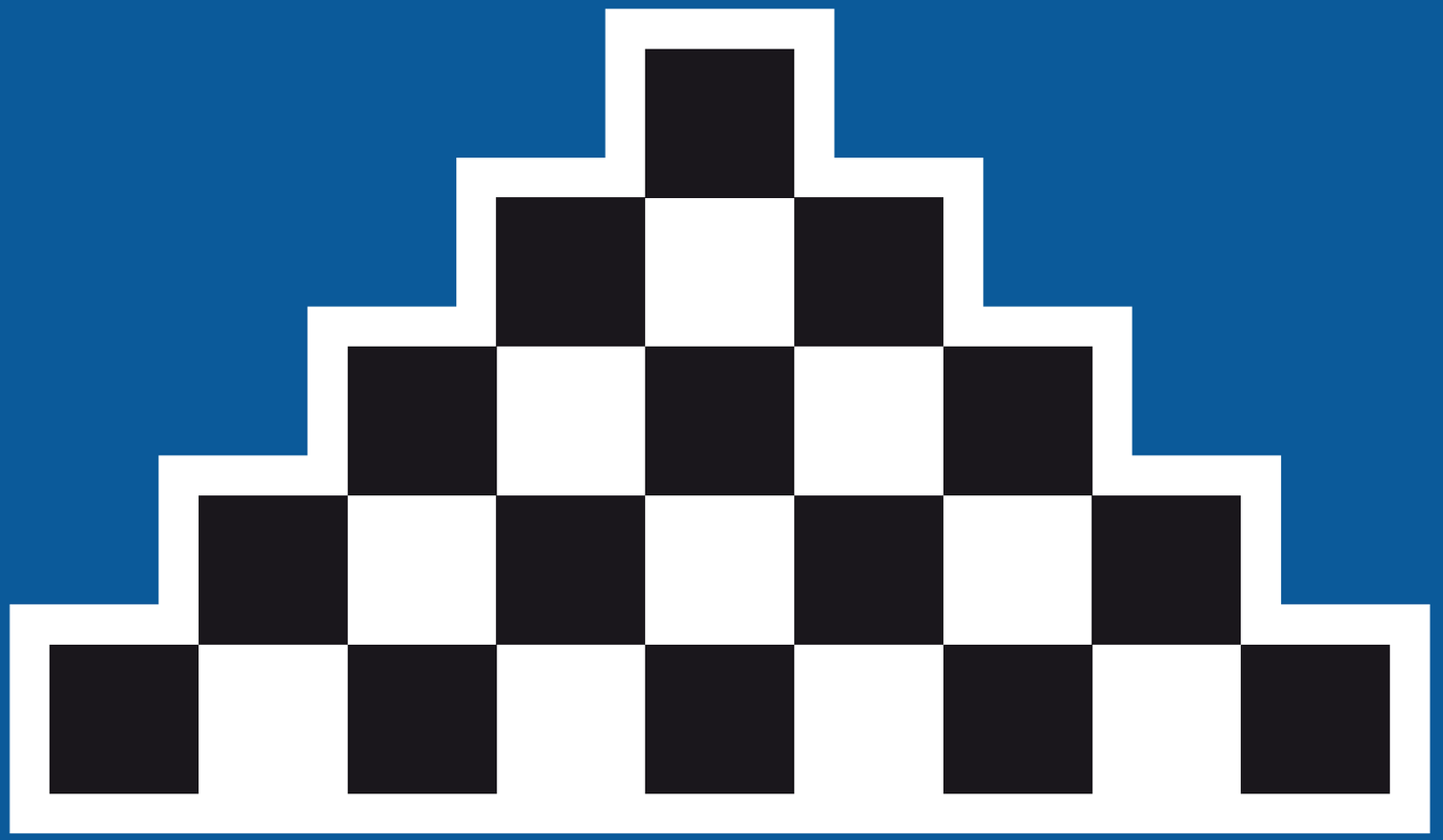
JavaScript Frameworks

- Frameworks erweitern eine Sprache um zusätzliche Funktionen
- unterscheiden sich von Bibliotheken, weil Frameworks auch eine Architektur vorgeben
  - nicht nur einen Haufen Funktionen…
  - …sondern auch Vorgaben, wie diese zu benutzen sind!
- JavaScript Frameworks korrigieren Browserunterschiede und erleichtern die Programmierung

JavaScript Frameworks

- Es existieren sehr viele JavaScript Frameworks
  - für unseren Einsatzzweck bieten alle ähnliche Funktionen
  - wir konzentrieren uns auf das Framework mit dem leichtesten Zugang!

Name	Beschreibung
Prototype	ältestes Framework, entstanden 2005 in Ruby on Rails
script.aculo.us	erweitert Prototype um visuelle Effekte
jQuery	aktuell beliebtestes Framework, viele "Plugins"
MooTools	komplexeres Framework mit mehr Funktionen
Dojo Toolkit	Framework um Desktop-Anwendungen im Browser zu erstellen



Grundlagen

Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen

jQuery Funktionsumfang

- Vereinfacht JavaScript für folgende Anwendungsfälle:
  - Selektion von HTML-Elementen
  - Veränderung von HTML-Elementen
  - Veränderung von CSS-Eigenschaften
  - Ereignisverarbeitung
  - JavaScript Effekte und Animationen
  - Ajax

jQuery Einbindung

- jQuery wird wie jede andere externe JavaScript-Datei mit `<script src="jquery.js"></script>` eingebunden.
- Die Datei kann entweder lokal (z.B. im selben Ordner) oder aus dem im Internet geladen werden:
  ```
  <script src="http://code.jquery.com/jquery-latest.js"></script>
  ```
- Nach der Einbindung können die jQuery-Funktionen im eigenen JavaScript verwendet werden.

jQuery Aufruf

- jQuery besteht nur aus einer Funktion:
  - `jquery(…);`
- Weil die Funktion aus eigenem JavaScript häufig aufgerufen werden wird, existiert noch eine Abkürzung:
  - `$(…);`
- Die Mächtigkeit von jQuery steckt in den Parametern der Funktion

Funktionsweise

- Erweitert JavaScript um die Möglichkeit mit CSS-Selektoren DOM-Knoten zu ermitteln.
  - Statt in JavaScript:
    `document.getElementsByTagName("p")[0]`
    `.innerHTML = "neuer Absatz";`
  - mit jQuery:
    `$("p").html("neuer Absatz");`
- Der erste Teil `$("p")` selektiert DOM-Knoten
- Der zweite Teil `html("neuer Absatz");` ruft die jQuery-Funktion `html()` auf allen gefunden Knoten auf

JavaScript Fallstricke

Warum funktioniert dies:

```
<!DOCTYPE html>
<html>
  <body>
    <p>Absatz</p>
    <script>
      var pNode = document.getElementsByTagName("p")[0];
      pNode.innerHtml = "neuer Absatz";
    </script>
  </body>
</html>
```

Dies aber nicht:

```
<!DOCTYPE html>
<html>
  <head>
    <script>
      var pNode = document.getElementsByTagName("p")[0];
      pNode.innerHtml = "neuer Absatz";
    </script>
  </head>
  <body>
    <p>Absatz</p>
  </body>
</html>
```

Lösung mit jQuery

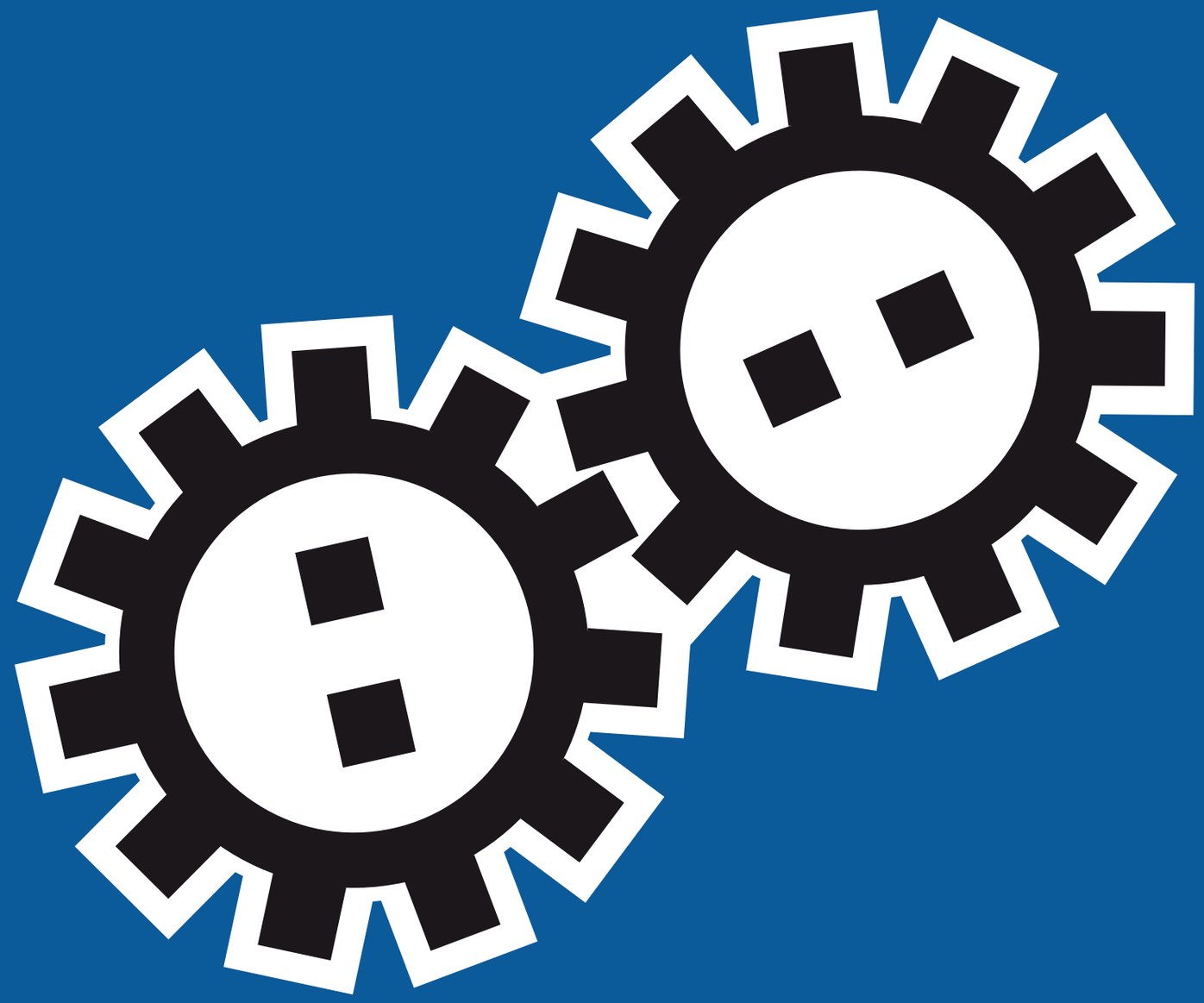
- jQuery benötigt Zugriff auf den DOM-Baum
- Der ist noch nicht aufgebaut, wenn JavaScript bereits ausgeführt wird.
  - jQuery definiert Hilfsmethode `ready(…);`
  - wenn der DOM Baum aufgebaut ist, wird die JavaScript-Funktion im Parameter ausgeführt
- `$(document).ready(alert("DOM fertig!"));`

jQuery Grundgerüst

```html
<!DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-latest.js">
    </script>
    <script>
      $(document).ready(domReady);

      function domReady() {
        $("p").html("neuer Absatz");
      }
    </script>
  </head>
  <body>
    <h1>Überschrift</h1>
    <p>Absatz</p>
  </body>
</html>
```

- Die fettgedruckten Zeilen sollten Sie unverändert übernehmen!
- Ihre jQuery-Anweisungen können Sie in `domReady()` schreiben



Techniken

Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen

DOM-Knoten mit CSS-Syntax selektieren

Selektoren

ID-Selektoren

HTML-Datei

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Überschrift</h1>
    <p id="absatz">Absatz</p>
  </body>
</html>
```

jQuery

```
$("#absatz").html("neuer Absatz");
```

klassisches JavaScript

```
document.getElementById(„absatz").innerHTML = "neuer Absatz";
```

Klassen-Selektoren

HTML-Datei

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Überschrift</h1>
    <p class="absatz">Absatz</p>
  </body>
</html>
```

jQuery

```
$(".absatz").html("neuer Absatz");
```

klassisches JavaScript

```
document.getElementsByClassName(„absatz")[0].innerHTML = "neuer Absatz";
```

Typ-Selektoren

HTML-Datei

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Überschrift</h1>
    <p>Absatz</p>
    <p>Absatz</p>
    <p>Absatz</p>
  </body>
</html>
```

jQuery

```
$("p").html("neuer Absatz");
```

klassisches JavaScript

```
for (i = 0; i <= 3; i++) {
  document.getElementsByTagName("p")[i].innerHTML="neuer Absatz";
}
```

gefundene DOM-Knoten mit jQuery verändern

Funktionen

HTML Inhalt

- verändert den Inhalt der gefunden Elemente
- `.html("<p>neues html</p>");`
- kann einfacher Text oder auch komplexes HTML sein

HTML-Datei

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Überschrift</h1>
    <p>Absatz</p>
  </body>
</html>
```

jQuery

```
$("p").html("neuer <em>Absatz</em>!");
```

Elemente einfügen

- `.before("<p>neues html</p>");`
- `.after("<p>neues html</p>");`
- fügt neue Element vor / nach den gefunden Elementen ein

HTML-Datei

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Überschrift</h1>
    <p>Absatz</p>
    <p>Absatz</p>
    <p>Absatz</p>
  </body>
</html>
```

jQuery

```
$("p").before("<h2>Unterabschnitt</h2>");
```

CSS ändern

- `.css(eigenschaft, wert);`
- ändert eine CSS-Eigenschaft auf allen gefundenen Elementen

HTML-Datei

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Überschrift</h1>
    <p>Absatz</p>
    <p>Absatz</p>
    <p>Absatz</p>
  </body>
</html>
```

jQuery

```
$("p").css("background-color", "Linen");
```

Klassen ändern

- Die Gestaltung sollte besser in der CSS-Datei erfolgen.
- Mittels Änderung des Klassen-Attributs kann der Browser angewiesen werden, das Element neu zu gestalten.
- `.addClass(CSS-Klasse);`
- `.removeClass(CSS-Klasse);`

HTML-Datei

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Überschrift</h1>
    <p>Absatz</p>
    <p>Absatz</p>
    <p>Absatz</p>
  </body>
</html>
```

jQuery

```
$("p").addClass("farbig");
```

Ereignisverarbeitung in jQuery

Ereignisse

Ereignisse

- Ereignisverarbeitung kann direkt in jQuery definiert werden
  - Keine on…-Attribute in den HTML-Elementen mehr notwendig
- Funktioniert genau wie alles andere in jQuery:
  - erst Elemente auswählen
  - dann für ein Ereignis eine Verarbeitungsfunktion setzen
- Wenn das Ereignis eintritt, wird die Funktion aufgerufen

Ereignisse: Beispiel

- Für jedes Ereignis gibt es eine Funktion in jQuery:
  click → `click(…)` focus → `focus(…)` …
- Parameter ist die Verarbeitungsfunktion

HTML-Datei

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Überschrift</h1>
    <p>Absatz</p>
  </body>
</html>
```

externer Link

jQuery

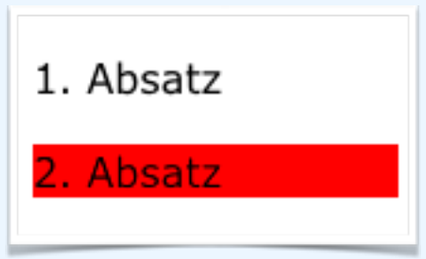
```
$("p").click(function() {
  alert("Absatz angeklickt!");
});
```

Ereignisse

- Wenn in jQuery mit dem Selektor mehrere Elemente gewählt wurden, bekommen alle die Ereignisverarbeitung
- Innerhalb der Ereignisverarbeitung kann mittels `$(this)` auf das Element zugegriffen werden, auf dem das Ereignis ausgelöst wurde

HTML-Datei

```
<!DOCTYPE html>
<html>
  <body>
    <p>1. Absatz</p>
    <p>2. Absatz</p>
  </body>
</html>
```



externer Link

jQuery

```
$("p").click(function() {
  $(this).css("background-color", "red");
});
```

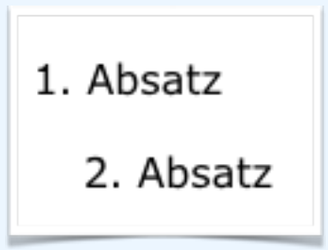
Animationen

Animationen

- keine Animationen im Sinne von Film oder Video
- Animationen als jQuery-äquivalent zu CSS3-Transitionen
  - verändere CSS-Eigenschaft langsam von Wert A zu Wert B
    z.B. `padding-left` von 0 auf 20
  - generische Form um Animationen zu programmieren

HTML-Datei

```
<!DOCTYPE html>
<html>
 <body>
    <p>1. Absatz</p>
    <p>2. Absatz</p>
  </body>
</html>
```



externer Link

jQuery

```
$("p").click(function() {
  $(this).animate({"padding-left": 20});
});
```

Hover

- Das hover Ereignis wertet nur die Mausbewegung über einem Element aus
- hover() erwartet zwei Funktionen:
  - erste wird aufgerufen, wenn die Maus das Element berührt
  - zweite wird aufgerufen, wenn die Maus das Element verlässt

HTML-Datei

```
<!DOCTYPE html>
<html>
 <body>
    <p>1. Absatz</p>
    <p>2. Absatz</p>
  </body>
</html>
```

1. Absatz

2. Absatz

externer Link

jQuery

```
$("p").hover(
  function() {
    $(this).animate({"padding-left": 20});
  },
  function() {
    $(this).animate({"padding-left": 0});
  }
);
```

Animationen

- Sind das jQuery-äquivalent zu CSS3-Transitionen
- Funktionieren leider nur mit CSS-Eigenschaften, die ganzzahlige Werte enthalten:
  - Abstände
  - Größen
- Funktionieren nicht mit komplexen Werten:
  - Farben
  - Schriftarten

dynamische Webseiten

Ajax

Ajax

Ajax JavaScript

```
var request = XMLHttpRequest();
request.open("GET", "download/data.txt", true);
request.onreadystatechange = verarbeiteDaten(request);
request.send(null);
…

function verarbeiteDaten(request) {
  if (request.readyState == 4) {
    textNode.innerHTML = request.responseText;
  }
}
```

readyState	Beschreibung
0	Zustand nach dem Erstellen des XMLHttpRequests
1	nach request.open()
2	nach request.send()
3	wenn die ersten Datenpakete angekommen sind
4	wenn die komplette Antwort angekommen ist

Ajax jQuery

- jQuery kapselt die Ajax Schritte in der Funktion `ajax();`
- Parameter enthalten url und Rückruffunktion

HTML-Datei

```
<!DOCTYPE html>
<html>
 <body>
    <p>1. Absatz</p>
    <p>2. Absatz</p>
  </body>
</html>
```

externer Link

ajaxdaten.txt

```
Text vom <em>Server</em> geladen!
```

jQuery

```
$.ajax({
  url: "ajaxdaten.txt",
  success: function(data) {
    alert("Ajax Daten: " + data);
  }
});
```

Ajax jQuery

- Die Antwort vom Server sollte im Dokument verwendet werden
- z.B. in die Absatzelemente einfügen

HTML-Datei

```
<!DOCTYPE html>
<html>
 <body>
    <p>1. Absatz</p>
    <p>2. Absatz</p>
    <button>Absätze ändern</button>
  </body>
</html>
```

ajaxdaten.txt

```
Text vom <em>Server</em> geladen!
```

jQuery

```
$("button").click(function() {
  $.ajax({
    url: "ajaxdaten.txt",
    success: function(data) {
      $("p").fadeOut().html(data).fadeIn();
    }
  });
});
```

Text vom *Server* geladen!

Text vom *Server* geladen!

Absätze ändern

externer Link

Ajax jQuery

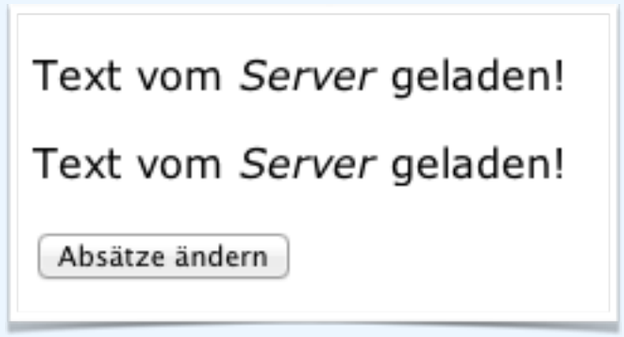
- noch einfacher mit der load(url) Funktion
- geladene Daten werden als HTML in das Element geschrieben

HTML-Datei

```
<!DOCTYPE html>
<html>
 <body>
    <p>1. Absatz</p>
    <p>2. Absatz</p>
    <button>Absätze ändern</button>
  </body>
</html>
```

ajaxdaten.txt

```
Text vom <em>Server</em> geladen!
```



externer Link

jQuery

```
$("button").click(function() {
  $("p").fadeOut().load('ajaxdaten.txt').fadeIn();
});
```

jQuery Erweiterungen

Plugins

jQuery Plugins

- jQuery ist sehr kompakt
  - Funktionsumfang ist klein
  - auf den letzten Folien haben wir fast alles vorgestellt
- Viele Erweiterungen gibt es als Plugins
  - zusätzliche JavaScript-Datei
  - Einbindung in die jQuery-Syntax:
    `$("#titel").neuePluginFunktion();`

Beispiel: Bilderkarusell

- Liste mit Bildern, Links, … als Endlosanimation:
  `.carouFredSel(…);`

```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-latest.js">
    </script>
    <script src="js/jquery.carouFredSel.js">
    </script>
    <script>
      $(document).ready(domReady);
      function domReady() {
        $("#carousel").carouFredSel({items : 4});
      }
    </script>
    <style type="text/css">…</style>
  </head>
  <body>
    <ul id="carousel">
      <li> c </li>
      <li> a </li>
      <li> r </li>
      <li> o </li>
      <li> u </li>
      <li> s </li>
      <li> e </li>
      <li> l </li>
    </ul>
  </body>
</html>
```

externer Link

Zusammenfassung

Vergleich jQuery und HTML5

HTML5 (+CSS3)	HTML4 + JavaScript(jQuery)
Nur aktuellste Browser unterstützen HTML5	Alle Browser können unterstützt werden
HTML und CSS sind immer aktiv	JavaScript kann abgeschaltet/gefiltert werden
HTML-Dokument kann bandbreitenschonend heruntergeladen werden	JavaScript-Bibliotheken müssen heruntergeladen werden
Semantische Informationen sind vorhanden	Semantische Informationen können nicht aus dem HTML-Dokument herausgelesen werden
Browser "garantiert" Validierung der Formulare	JavaScript "garantiert" Validierung der Formulare

Danke