Frontendtechnologien

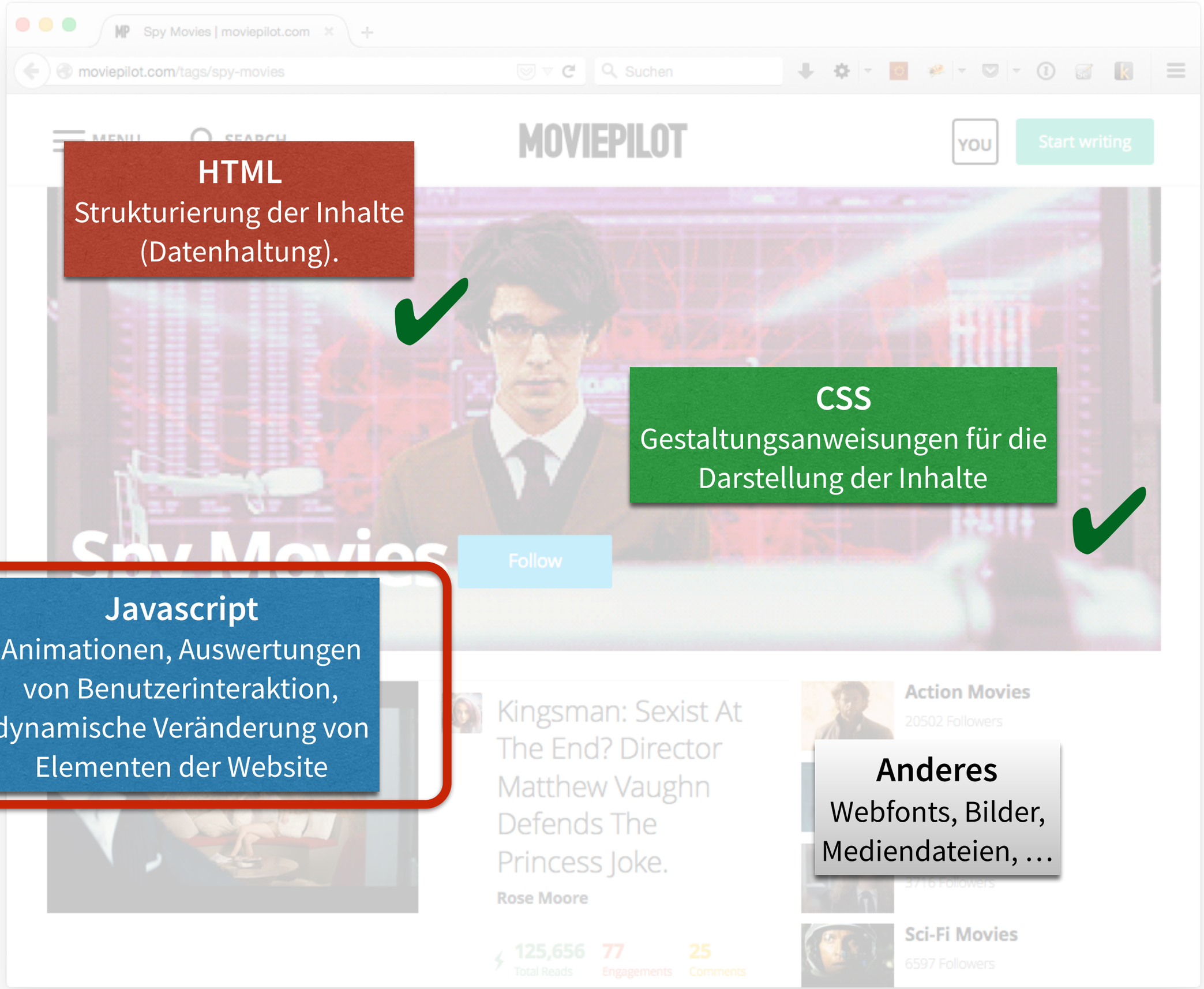
JavaScript

Einführung in Softwaretechnologien

Alexander Thomas
me@alexander-thomas.net

Überblick


Grund-
begriffe
HTML
CSS
Frontend-
technologien
Mobile Apps
JS
Javascript
Vertical
UX/UI
Service
Data
User Stories
Anwendungsschicht
Darstellerschicht
Datenschicht
Client-Server Systeme
Schnittstellen
Netzwerktopologien
Netzwerkschichten
Technische Konzeption
& Modellierung von
Softwareanwendungen
Akteur / Actor
Anwendungsfall / Use Case
UML
Server-
technologien &
Netzwerke
http://
Datenbanken
Privacy
Analytics / SEO
Content & CMS
User Experience &
User Interface
Design
Wireframes
Google
Analytics
Web Analytics
WordPress
Content Management
Content Marketing


Spy Movies | moviepilot.com
moviepilot.com/tags/spy-movies
Suchen
MENU
SEARCH
MOVIEPILOT
YOU
Start writing
HTML
Strukturierung der Inhalte (Datenhaltung).
CSS
Gestaltungsanweisungen für die Darstellung der Inhalte
Javascript
Animationen, Auswertungen von Benutzerinteraktion, dynamische Veränderung von Elementen der Website
Anderes
Webfonts, Bilder, Mediendateien, …
Follow
Kingsman: Sexist At The End? Director Matthew Vaughn Defends The Princess Joke.
Rose Moore
125,656 Total Reads
77 Engagements
25 Comments
Action Movies
20502 Followers
Sci-Fi Movies
6597 Followers

JavaScript

- **JavaScript** ist eine **Programmiersprache**, die es (u.a.) erlaubt, Anwendungen zu erstellen, die **im Browser ausgeführt** werden

- Damit ist **JavaScript** neben HTML und CSS die **dritte essentielle Technologie des WWW**

  - HTML: Datenhaltung

  - CSS: Datenpräsentation

  - JavaScript: Interaktion und Animation

JavaScript

- JavaScript hat nichts mit der verbreiteten Programmiersprache Java zu tun!

- Die Sprache wurde Ende **1995 von Netscape** zunächst unter dem Namen „LiveScript“ entwickelt und im Netscape Navigator implementiert

- Seit 1997 wird die Sprache in der **ECMAScript Spezifikation** standardisiert

- Heute unterstützen **alle Browser** die Ausführung von JavaScript, **ohne** dass **weitere Plugins** nötig wären

JavaScript

- JavaScript ist eine **Interpretersprache**. Das heißt, der Quellcode wird erst zur Laufzeit von einem Interpreter in Maschinenbefehle umgesetzt und ausgeführt

- Deswegen liegen JavaScript-Programme als Quellcode in jeder Webseite offen

JavaScript

- Zu typischen **Anwendungsfällen** von JavaScript im Browser zählen:
- **Anwendungen**: Spiele, Mobile WebApps
- **Eingabeverarbeitung**: z.B. Validierung von Formularfeldern
- **Dynamische Webseiten**: Bildergallerien, Popups u.ä.

JavaScript einbinden

- Ähnlich wie bei CSS, wird JavaScript entweder innerhalb von **Script-Bereichen im Kopf** des HTML-Dokuments oder in einer **externen Datei** notiert

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="UTF-8" />
  <title>Title</title>

  <script>
     ...
  </script>
 </head>
 <body onload="...">
```

Script Bereich im `head`

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Title</title>

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

  </head>
  <body onload="...">
```

Referenz auf externes Script

JavaScript
Event Handler

- JavaScript-Funktionalität wird über **Event-Handler an HTML-Elemente gebunden** (Event Binding)

- Es gibt Event-Handler für verschiedene **Arten von Ereignissen**:

```
<p onclick="alert('Angeklickt');">
  Klick. Mich. An.
</p>
```

Mausereignisse (z.B. wenn ein Element angeklickt wird)

JavaScript
Event Handler

- JavaScript-Funktionalität wird über **Event-Handler an HTML-Elemente gebunden**

- Es gibt Event-Handler für verschiedene **Arten von Ereignissen:**

```
<textarea
  onkeydown="alert('Oh, eine Taste');">

  Drück. Eine. Taste.

</textarea>
```

Tastaturereignisse (z.B. wenn eine Taste gedrückt wird)

JavaScript
Event Handler

- JavaScript-Funktionalität wird über **Event-Handler an HTML-Elemente gebunden**

- Es gibt Event-Handler für verschiedene **Arten von Ereignissen**:

```
<body onload="alert('Das Dokument wurde geladen');">...
```

Fenster- und Dokumentenereignisse

JavaScript
Event Handler

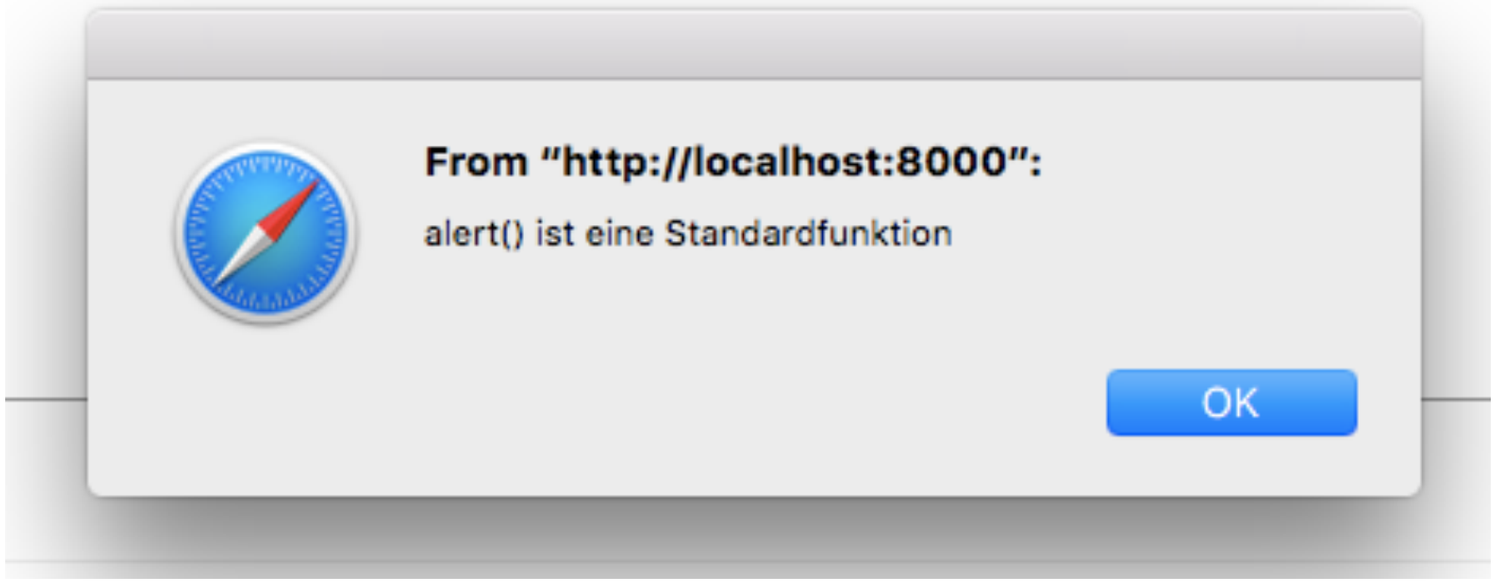
- JavaScript-Funktionalität wird über **Event-Handler an HTML-Elemente gebunden**

- Es gibt Event-Handler für verschiedene **Arten von Ereignissen:**

```
<textarea onfocus="alert('Textfeld ist aktiv');">
    Klick. Hier. Rein.
</textarea>
```

Interaktive Ereignisse
(z.B. wenn der Cursor in einem Textfeld steht)

JavaScript Funktionen



`alert()` ist **eine Funktion**, die JavaScript standardmäßig mitbringt

JavaScript Funktionen

- Funktionen kapseln **in sich abgeschlossene** JavaScript-**Prozeduren**

- Diese Prozeduren sind über einen **eindeutigen Bezeichner** adressierbar

- Sie *können* 1-n **Eingabeparameter** erhalten (Argumente)

- Sie *können* einen **Rückgabewert** ausgeben

JavaScript Funktionen

```
alert('Hallo Welt!');
```

- Wie lautet der **Funktionsbezeichner**?
- Was ist der **Eingabeparameter**?
- Gibt es einen **Rückgabewert**?

JavaScript Funktionen

- Funktionen erlauben es **Operationen** zusammenzufassen und zu **kapseln**

- Dadurch kann diese Funktion leichter **wiederverwendet** werden (kein duplizierter Code!)

- Zusammen mit dem Namen der Funktion dient dies zudem der **Übersichtlichkeit**

JavaScript
Eigene Funktionen

```
eineFunktion ( argument1, argument2, argument... );
```

Definition:

```
function eineFunktion ( argument1, argument2, argument... )
{

  //Funktionskörper
  ...

  //Rückgabewert
  return dieAntwort;

}
```

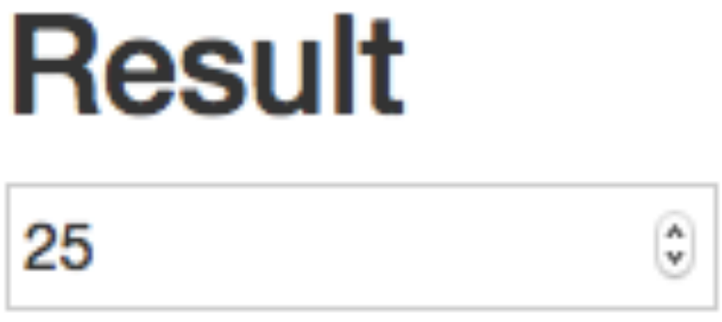
JavaScript
Eigene Funktionen

```
function berechneMwst ( brutto )
{
   var mwst = 0.19;
   return brutto * mwst;
}

// Aufruf:
berechneMwst( 25 );
```

`berechneMwst()` ist **eine Funktion**, mit der man die Mehrwertsteuer eines Betrags berechnen kann

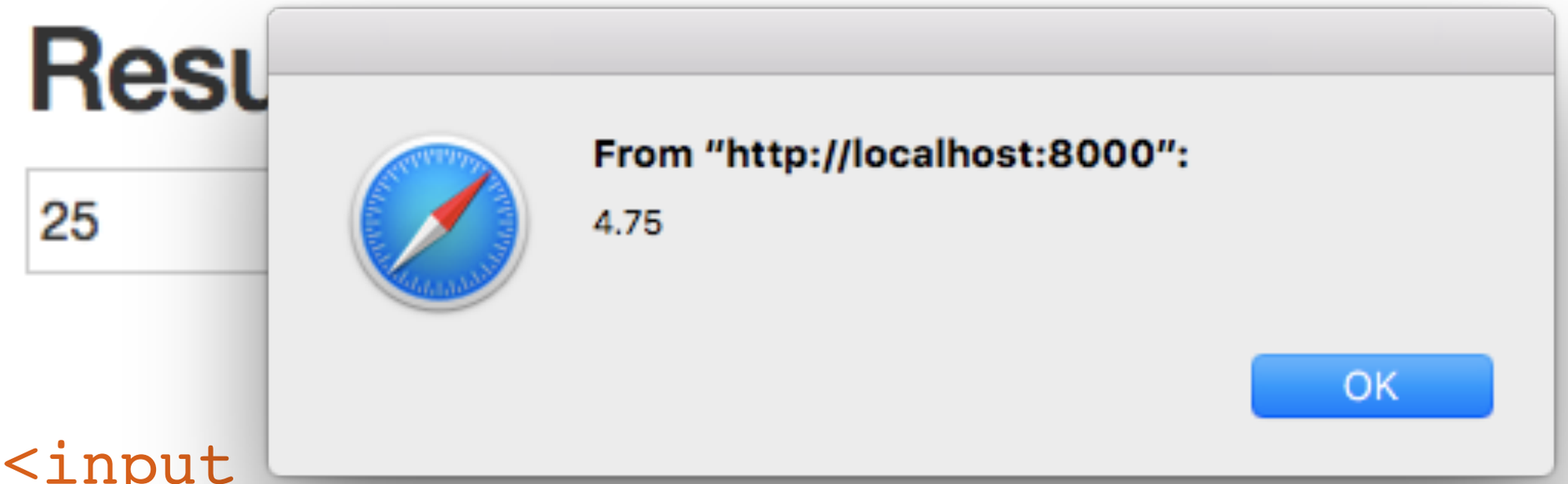
JavaScript
Eigene Funktionen




```
<input
 type="number"
 onblur="berechneMwst( this.value )"
/>
```

Der **Inhalt des Eingabefelds** (`this.value`) wird **an die Funktion übergeben** und ausgeführt, wenn ein anderes Element der Seite aktiviert wird

JavaScript
Eigene Funktionen




```
<input
 type="number"
 onblur="alert(berechneMwst( this.value ))"
/>
```

Wir können das berechnete Ergebnis als Alert-Fenster ausgeben

JavaScript
DOM-Scripting

- Der Baum eines HTML-Dokuments steht in JavaScript über die Variable `document` bereit

- Über diesen Weg kann man **programmatisch** auf beliebige **Elemente des Dokuments zugreifen** und **Zustand und Eigenschaften hinzufügen, ändern oder entfernen**

- Man bezeichnet dies üblicherweise als **DOM-Scripting**

JavaScript
DOM-Scripting

- Das `document` Objekt stellt eine Reihe von **Funktionen** bereit, die es erlauben, **Knoten im Dokumenten-Baum zu selektieren**

JavaScript
DOM-Scripting

- Das `document` Objekt stellt eine Reihe von **Funktionen** bereit, die es erlauben, **Knoten im Dokumenten-Baum zu selektieren**

```
<script type="text/javascript">
  document.getElementById('books');
</script>

<ul id="books">
  <li>ein Buch</li>
  ...
```

`document.getElementById()`

JavaScript
DOM-Scripting

- Das `document` Objekt stellt eine Reihe von **Funktionen** bereit, die es erlauben, **Knoten im Dokumenten-Baum zu selektieren**

```
<script type="text/javascript">
  document.getElementsByTagName('li');
</script>

<ul id="books">
  <li>ein Buch</li>
  <li>noch ein Buch</li>
  ...
```

`document.getElement`**`s`**`ByTagName()`

JavaScript
DOM-Scripting

- Das `document` Objekt stellt eine Reihe von **Funktionen** bereit, die es erlauben, **Knoten im Dokumenten-Baum zu selektieren**

```
<script type="text/javascript">
  document.querySelector('li.special-book');
</script>

<ul id="books">
  <li>ein Buch</li>
  <li class="special-book">mein Lieblingsbuch</li>
  ...
```

document.querySelector()

JavaScript
DOM-Scripting

- Das `document` Objekt stellt eine Reihe von **Funktionen** bereit, die es erlauben, **Knoten im Dokumenten-Baum zu selektieren**

```
<script type="text/javascript">
  document.querySelectorAll('li.boring-book,
                             li.exciting-book');
</script>

<ul id="books">
  <li class="boring-book">ein langweiliges Buch</li>
  <li class="exciting-book">ein spannendes Buch</li>
  <li>ein Buch, das nicht selektiert werden soll</li>
  ...
```

`document.querySelectorAll()`

JavaScript
Element Eigenschaften

- Rückgabewert der Selektor-Methoden ist stets ein Element oder eine Kollektion von Elementen

- Ein Element ist die JavaScript Repräsentation des entsprechenden Knotens im HTML-Baum

```
<script>
  var eingabeFeld = document.querySelector('#beispiel');

  //  [type="number", value="23", id="beispiel"]
  console.log(eingabeFeld.attributes)
</script>

<input type="number" id="beispiel" value="23">
```

JavaScript
DOM-Scripting

Betrag wird eingegeben

Mehrwertsteuer wird berechnet


```
Betrag:
<input
  type="number"
  onblur="berechneMwst( this.value )"
/>

Mehrwertsteuer:
<input
  type="number"
  id=“mwst"
/>
```

JavaScript
DOM-Scripting

```
function berechneMwst ( brutto )
{
  var mwst = 0.19;
  var betrag = brutto * mwst;
  var eingabeFeld =

  document.querySelector('#mwst');
  eingabeFeld.value = betrag;
}
```

JavaScript
CSS Eigenschaften

- Durch **DOM-Scripting** kann natürlich nicht nur der Inhalt, sondern auch die **Gestaltung von HTML-Elementen verändert** werden.

JavaScript
CSS Eigenschaften

- Entweder durch **Hinzufügen von CSS-Klassen**

```
<script>
  el = document.getElementById('beispiel');
  el.classList.add('error');
</script>

<input type="number" class="warning" id=“beispiel"
value="23">
```

JavaScript
CSS Eigenschaften

- Oder durch **das Entfernen von CSS-Klassen**

```
<script>
  el = document.getElementById('beispiel');
  el.classList.remove('warning');
</script>

<input type="number" class="warning" id="beispiel"
               value="23">
```

JavaScript
CSS Eigenschaften

- Oder durch **Zugriff auf das `style` Attribut**

```
<script>
  el = document.getElementById('beispiel');
  el.style.width = '20px';
</script>

<input type="number" style="width:10px" id="beispiel"
value="23">
```

JavaScript
Knoten hinzufügen

```
<script>
  var body = document.querySelector('body');
  var p = document.createElement('p');
  var img = new Image('https://goo.gl/ez8ydC');

  p.appendChild(img);
  body.appendChild(p);
</script>
```

- **HTML-Knoten** können **erzeugt** und dem Dokumenten-Baum zur Laufzeit **hingefügt** werden!

<u>=> So funktionieren bspw. Ads und Tracking!</u>

JavaScript

- Wenn eine Webseite die **Ausführung von JavaScript voraussetzt**, sollte immer ein **`noscript` Bereich** definiert werden, denn der Nutzer kann die Ausführung von JavaScript unterbinden

- ***Besser** ist es allerdings, wenn die **Webseite auch ohne JavaScript nutzbar** bleibt!*

```
<!DOCTYPE html>
<html>
  <head>
    ...
  </head>
  <body>
    <noscript>
      Dieser Text wird angezeigt,
      wenn der Nutzer kein
      JavaScript aktiviert hat.
    </noscript>
    ...
```

Vielen Dank.

Zum Weiterlesen

- https://jsfiddle.net/
  Plattform zum Herumspielen mit JavaScript

- http://youmightnotneedjquery.com
  Sehr gute Übersicht über zeitgemässe JavaScript Features

- http://www.webdirections.org/blog/html5-selectors-api-its-like-a-swiss-army-knife-for-the-dom/

- http://webkompetenz.wikidot.com/html-handbuch:dom-scripting-grundlagen#toc0