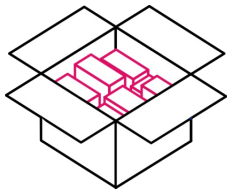
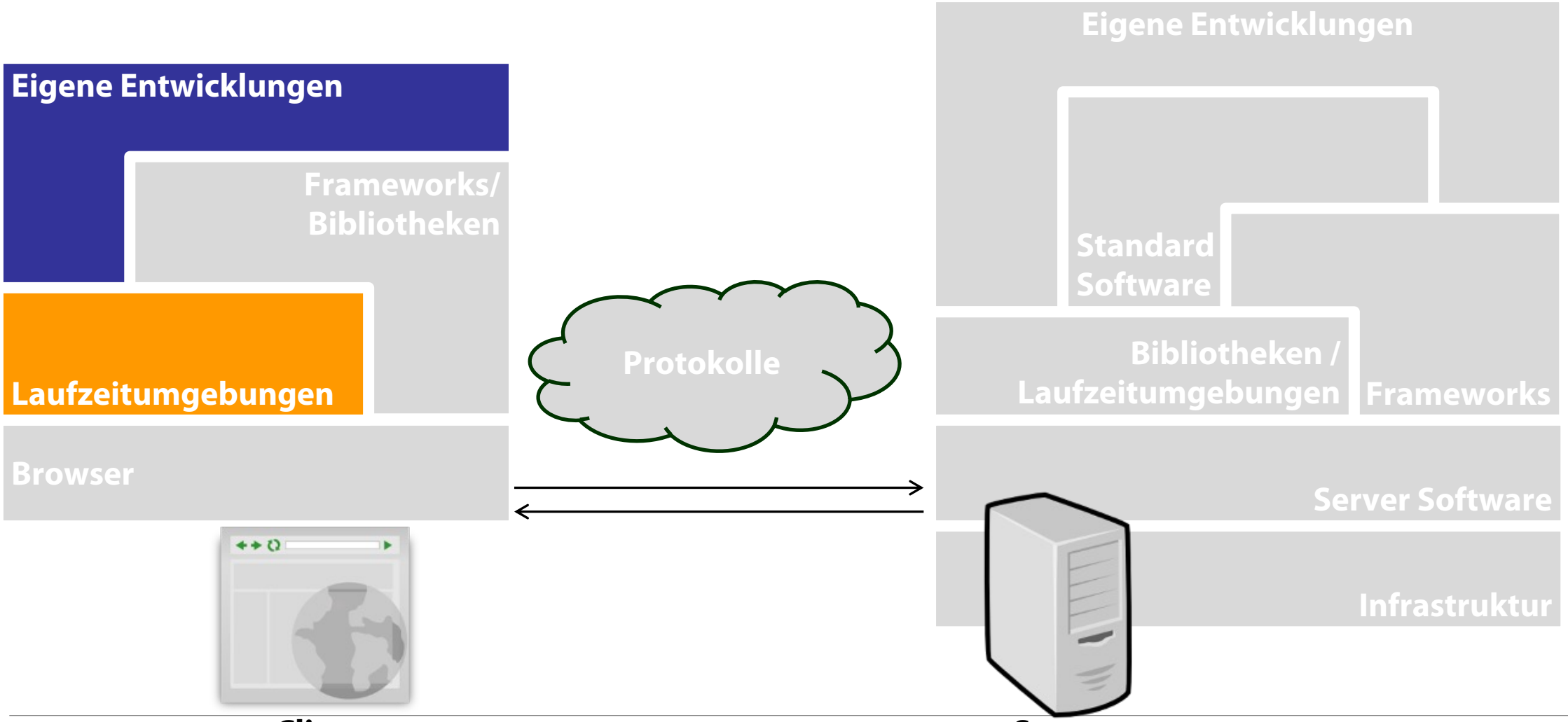
Client-seitiges JavaScript

Hoai Viet Nguyen – TH Köln

Technology
Arts Sciences
TH Köln

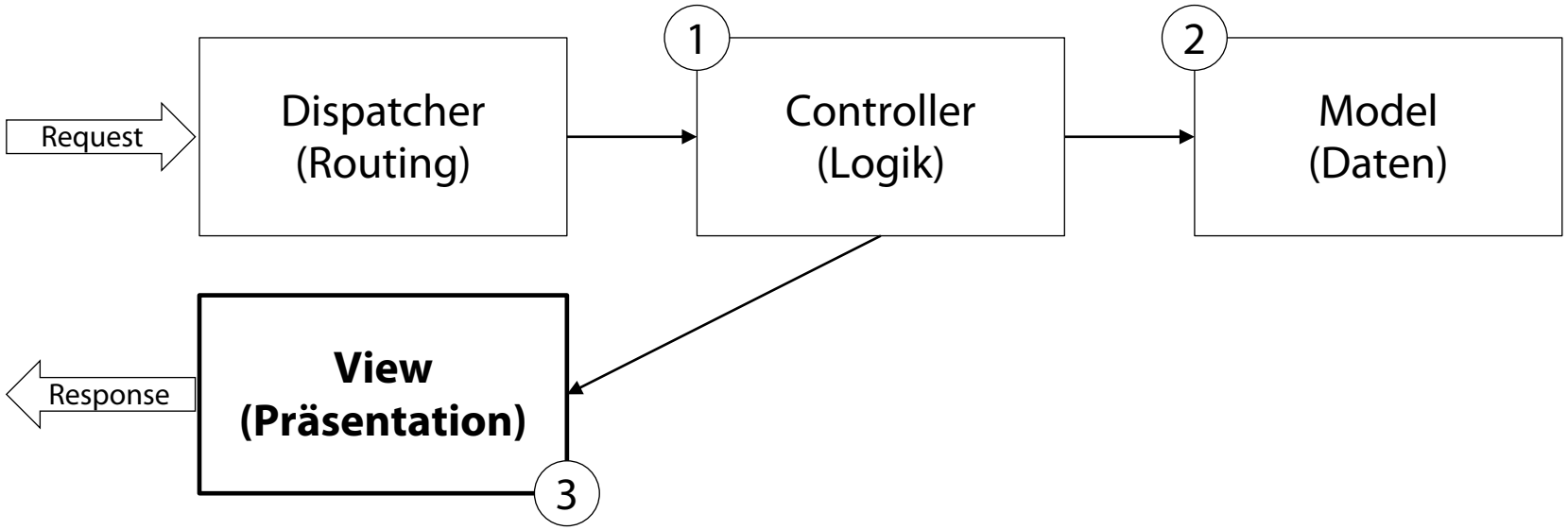


Struktur des Themengebietes



View

- **Gibt die Daten in geeignete/angefragter Repräsentation aus. Daten können verschiedene Repräsentationen haben (HTML, XML oder JSON)**



Umsetzungsmöglichkeiten MVC

Moderne Web-Anwendungen setzen
immer häufiger diese Umsetzungsmöglichkeiten ein,
da dadurch die Benutzung interaktiver
und reaktionsfreudiger realisiert werden kann.

JavaScript

- Interpretersprache
  - Wird im Browser client-seitig interpretiert
  - Seit 2009 auch server-seitig einsetzbar (node, deno, bun, vert.x)
- Von Webbrowsern verwendet, um client-seitigen Code auszuführen
  - Definition von Anwendungslogik, sobald die HTML-Seite vollständig geladen ist
  - Implementierung von Interaktionen und Animationen
  - Manipulation von HTML-Inhalten
  - Über JavaScript können auch Request versendet und Verbindungen zu Server aufgebaut werden

Integration von JavaScript in die HTML-Seite

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>JavaScript Demo</title>
    </head>
    <body>
        <script>
            //Code
        </script>
    </body>
</html>
```

Eingebettet (inline JavaScript)

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>JavaScript Demo</title>
    </head>
    <body>
        <script src="/javascripts/app.js">
        </script>
    </body>
</html>
```

Externe Datei

Datentypen in JavaScript

- boolean
  - `true` oder `false`
- Number
  - keine explizite Trennung zwischen Integer und Gleitpunktzahlen
  - Zahlen werden intern als 64-Bit-Gleitkommazahlen gespeichert
- String
  - Zeichenketten
  - Definierbar mit `"Text"` oder `'Text'`
- Undefined

Dynamische Typen

- Variablen haben keinen festen Typen
- Deklaration nicht notwendig
- Schlüsselwörter `var` erzeugt eine Variable
- Im weiteren Programmverlauf kann der Typ mittels Zuweisung geändert werden

```
var a = 10
console.log(a)
a +=5
console.log(a)
a = "Hello"
console.log(a)
a = true
```

Arrays

- Ebenfalls ungetypt
- Erzeugung mit `[]` oder `new Array()`
- wachsen dynamisch
- sind Objekte und haben nützliche Eigenschaften (Variablen und Methoden)
  - `length`
  - `push()`
  - `concat()`

```
var array = [12, 1, "Hello"]

var anotherArray = new Array(1,2,3)
anotherArray[2] = "World"

console.log(anotherArray.length)

anotherArray.push(true)

array.concat(anotherArray)
```

Funktionen

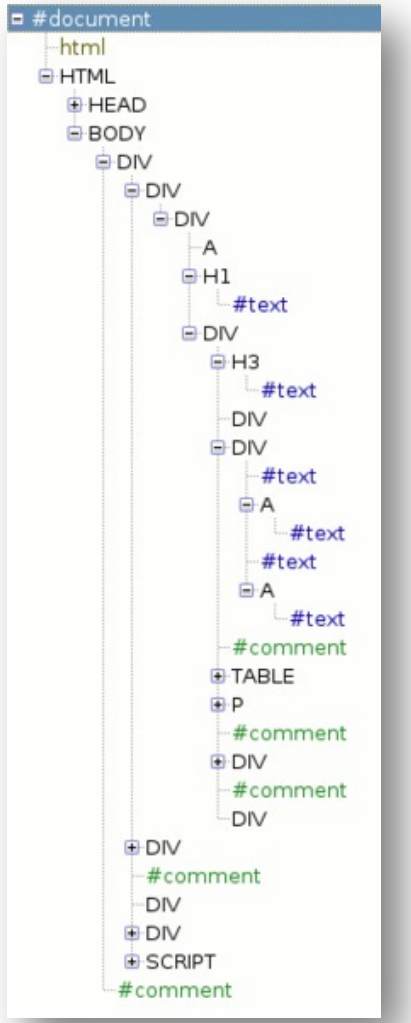
- **Werden mit dem Schlüsselwort `function` definiert**
- **Können Variablen zugewiesen werden**

```
function square(x){
    return x*x;
}
square(3)

var add = function (a, b){
    return a + b;
}
add(1,2)
```

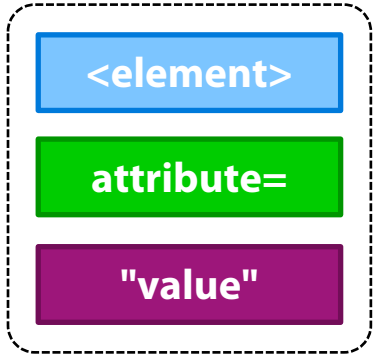
DOM (Document Object Model)

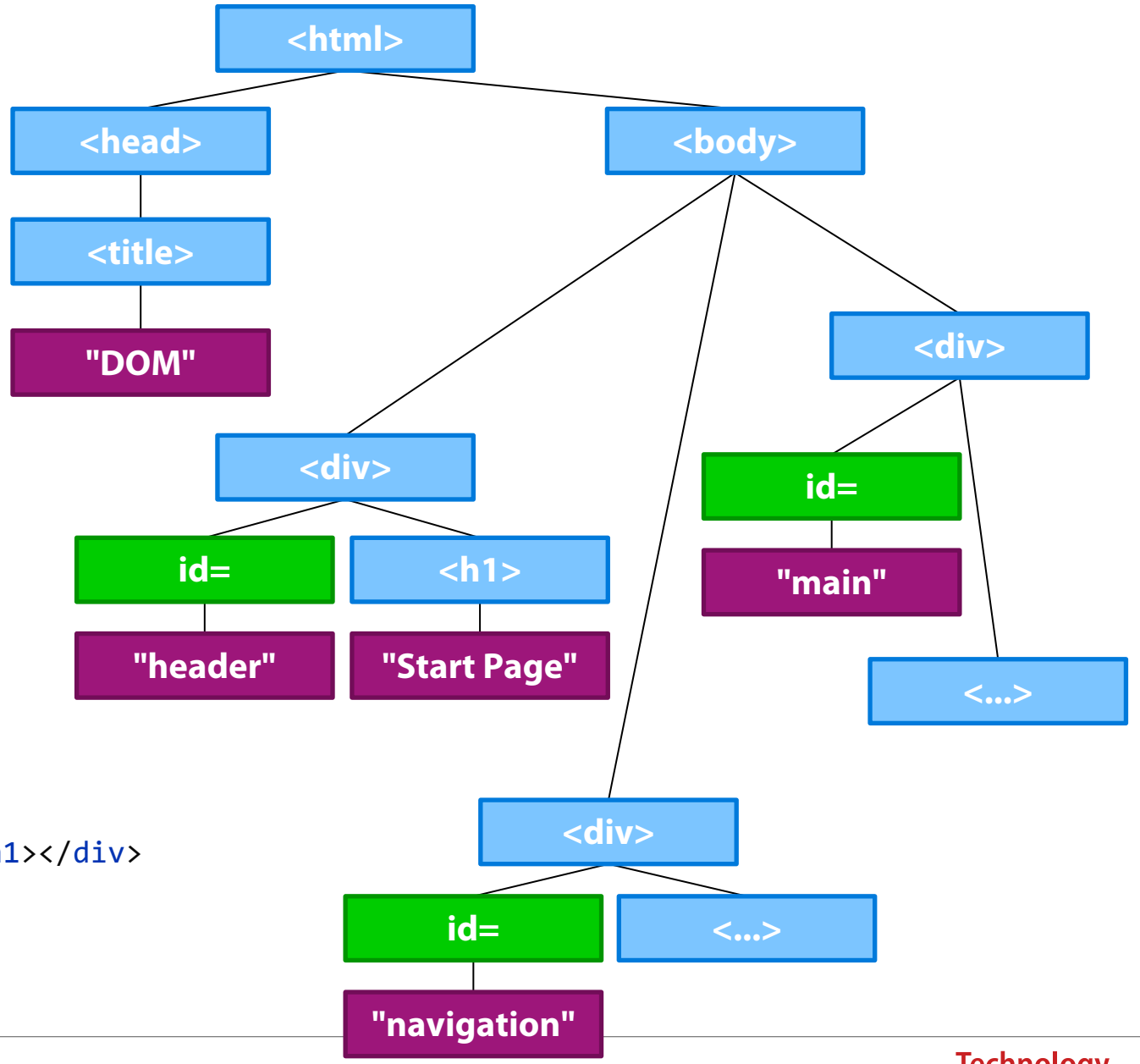
- Spezifikation einer Schnittstelle für den Zugriff auf HTML- oder XML-Dokumente
- Standardisiert von World Wide Web Consortium (W3C)
- Dokument wird als Baumstruktur repräsentiert



Quelle: http://de.wikipedia.org/wiki/Document_Object_Model

Beispiel





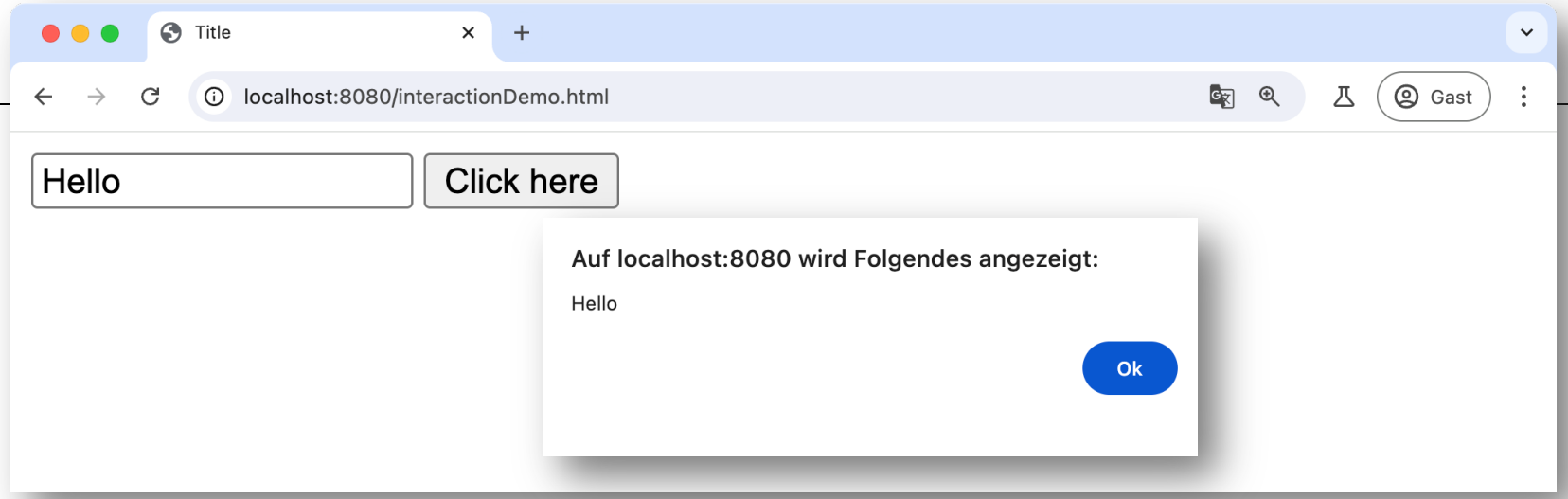

```
<html lang="en">
    <head>
        <title>DOM</title>
    </head>
    <body>
        <div id="header"><h1>Start Page</h1></div>
        <div id="navigation"></div>
        <div id="main"></div>
    </body>
</html>
```

HTML-Elemente verändern

```
<html lang="en">
<head>
    <title>GetElementById Demo</title>
</head>
<body>
    <h1 id="title">Title here</h1>
    <h2 id="subtitle">Subtitle here</h2>
    <script>
        document.getElementById("title").innerText = Date()
        document.getElementById("subtitle").innerHTML = "<b>New Subtitle" + Math.random() + "</b>"
    </script>
</body>
</html>
```

Interaktionen – Auf Ereignisse reagieren (1)

```
<html lang="en">
  <head>
      <title>Title</title>
  </head>
<body>
  <input type="text" id="message">
  <button type="button" id="myButton">Click here</button>
  <script>
    document.getElementById("myButton").addEventListener("click",clickListener)
    function clickListener(){
      alert(document.getElementById("message").value)
    }
  </script>
</body>
</html>
```



Interaktionen – Auf Ereignisse reagieren (2)

```
<html lang="en">
<head>
    <title>On Click Demo</title>
</head>
<body>
    <button onclick="showDate()">Display date</button>
    <p id="date"></p>
    <script>
        function showDate(){
            document.getElementById("date").innerText = Date()
        }
    </script>
</body>
</html>
```

Ereignisse

- **Laden der Webseite oder eines Bildes**
  - `onload, unload (load, unload)`
- **Mauszeiger über einem bestimmten Element der Webseite**
  - `onmouseover, onmouseout (mouseover, mouseout)`
- **Mausclick**
  - `onclick (click)`
- **Events in Input-Feldern**
  - `onfocus, onblur, onchange, onselect`
- **Absenden eines Formulars**
  - `onsubmit (submit)`
- **Tastendruck**
  - `onkeydown, onkeypress, onkeyup (keydown, keypress, keyup)`