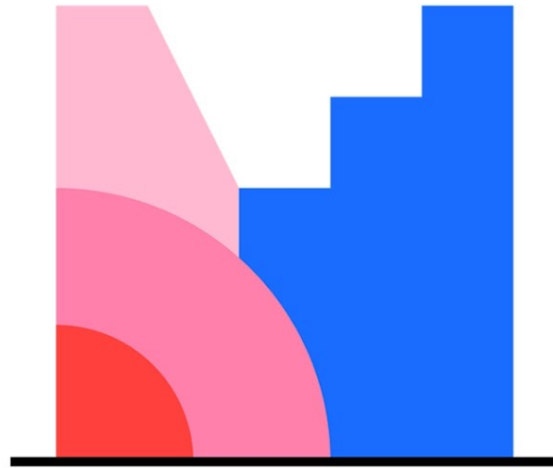













Blockwoche: Web Programming Lab 🚀

Recap: Web Architektur, JavaScript Sprachkonzepte



Mentimeter

Onboarding – Programm Blockwoche

Montag1 	Dienstag  	Mittwoch  	Donnerstag   	Freitag   
Architekturansätze von Web Anwendungen JavaScript Sprachkonzepte	Client-Side- JavaScript I	Angular	Angular	Offline- & Progressive Web Apps
JavaScript Sprachkonzepte	Client-Side- JavaScript II Frameworks & Typescript	Angular	Server-Side- JavaScript	Authentication @ Web Apps

Agenda Vormittag

- **DOM & BOM**
- **Document- & Resource Loading**
- **Events**
- **Forms & Controls**
- **Data Access via HTTP**
- **Web Components** (*evtl. Selbststudium*)
- **Building & deployment for production** (*Selbststudium*)

Github Repository für diesen Input

👉 <https://github.com/web-programming-lab/javascript-clientside>



Mislav Marohnić

@mislav



We're finally finished removing jQuery from [GitHub.com](https://github.com) frontend. What did we replace it with? No framework whatsoever:

- querySelectorAll,
- fetch for ajax,
- delegated-events for event handling,
- polyfills for standard DOM stuff,
- CustomElements on the rise.

GitHub

GitHub

GitHub is where people build software. More than 40 million people use GitHub to discover, fork, and contribute to over 100 million github.com

♥ 9,027 10:57 AM - Jul 25, 2018



💬 3,647 people are talking about this



Client-Side JavaScript I

Document- und Browser Object Model, Document- &
Resource- Loading



JS

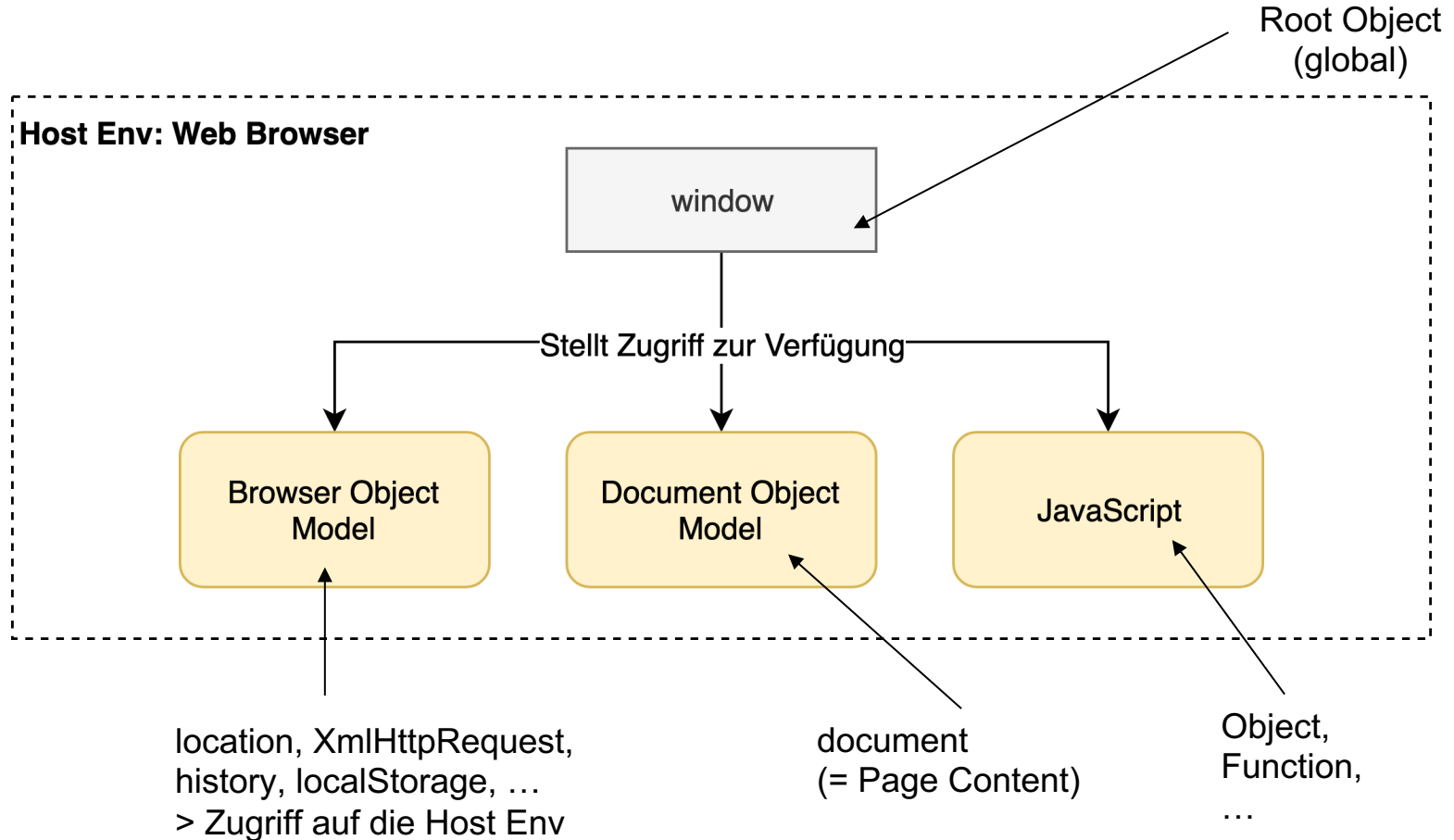
DOM & BOM – Host Environment

JavaScript wurde ursprünglich für Web Browser gemacht. Mittlerweile ist der Web Browser nicht mehr die einzige Umgebung, auf welcher JavaScript läuft (z.B. Web-Server, Raspberry-Pi, Desktop-App, Mobile-App, Entwicklungstools, etc.). Die ECMAScript Spezifikation spricht hier von sogn. **Host Environments**:

“programming language for performing computations and manipulating computational objects within a **host environment**.”

“[...] the existing system is said to **provide a host environment of objects and facilities**, which completes the capabilities of the scripting language.”

DOM & BOM – Host Environment



DOM & BOM – Host Environment

Beispiele Zugriff **Web Browser**

- Zugriff auf DOM: `let body = window.document.body;`
- Zugriff auf BOM: `let storage = window.localStorage;`

Beispiel Zugriff **NodeJS**

- Zugriff auf Environment Variabeln:

```
let envVar = process.env.xyz;
```

*Schauen wir detailliert am
Donnerstag Nachmittag an!*

DOM & BOM – Spezifikationen

DOM Spezifikation (Document + Events Spec):

<https://dom.spec.whatwg.org>

HTML Spezifikation (HTML + BOM):

<https://html.spec.whatwg.org>

Hintergrundwissen:

[W3C](#): World Wide Web Consortium

[WHATWG](#): Web Hypertext Application Technology Working Group
(Arbeitsgruppe: Mozilla, Microsoft, Apple und Google)

Die WHATWG entstand aus einem W3C Workshop (2004), da einige Mitglieder beunruhigt waren über den Fokus der W3C (-> XHTML, wenig Interesse in HTML)

W3C AND WHATWG to work together to advance the open web platform:

<https://www.w3.org/blog/2019/05/w3c-and-whatwg-to-work-together-to-advance-the-open-web-platform/>

Document Object Model

HTML Dokument

snippets/dom

```
<!DOCTYPE html>
<html>
<head>
<title>Beispiel</title>
</head>
<body>
<h1>Beispiel</h1>
<p>Das ist ein<a
href="demo.html">einfaches</a>Beispiel.</p>
<!-- dies ist ein Kommentar -->
</body>
</html>
```

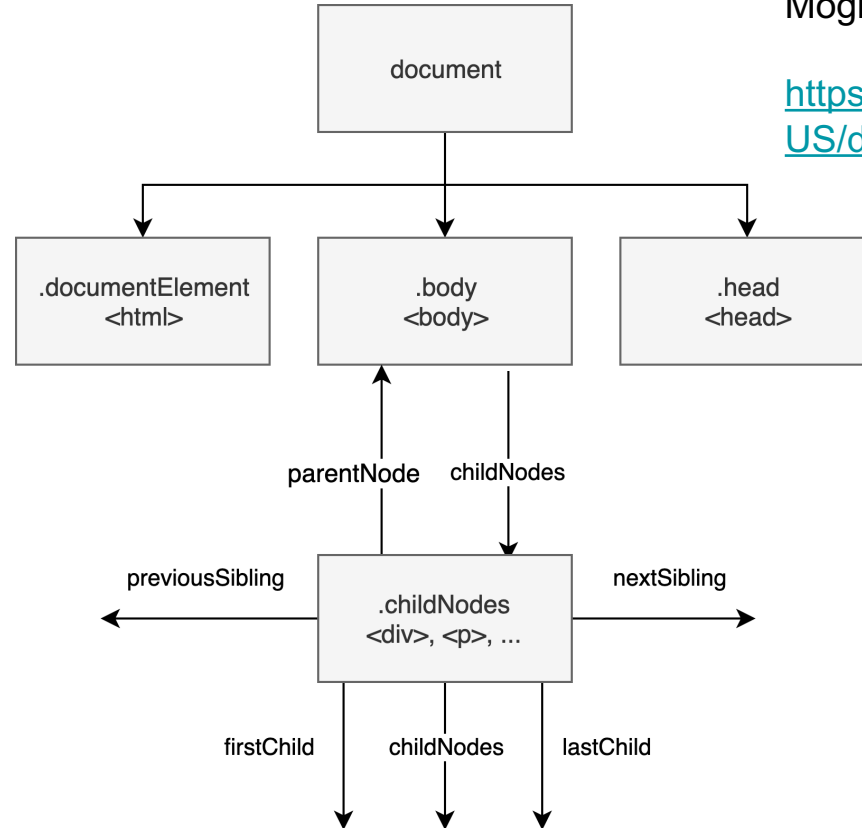
DOM

```
├ DOCTYPE: html
└ html
  ├── head
  │   ├── #text: ␣
  │   └── title
  │       └ #text: Beispiel
  └── #text: ␣
      └ body
          ├── #text: ␣
          ├── h1
          │   └ #text: Beispiel
          ├── #text: ␣
          ├── p
          │   ├── #text: Das ist ein
          │   ├── a href="demo.html"
          │   │   └ #text: einfaches
          │   └ #text: Beispiel.
          ├── #text: ␣
          ├── #comment: dies ist ein Kommentar
          └ #text: ␣
```

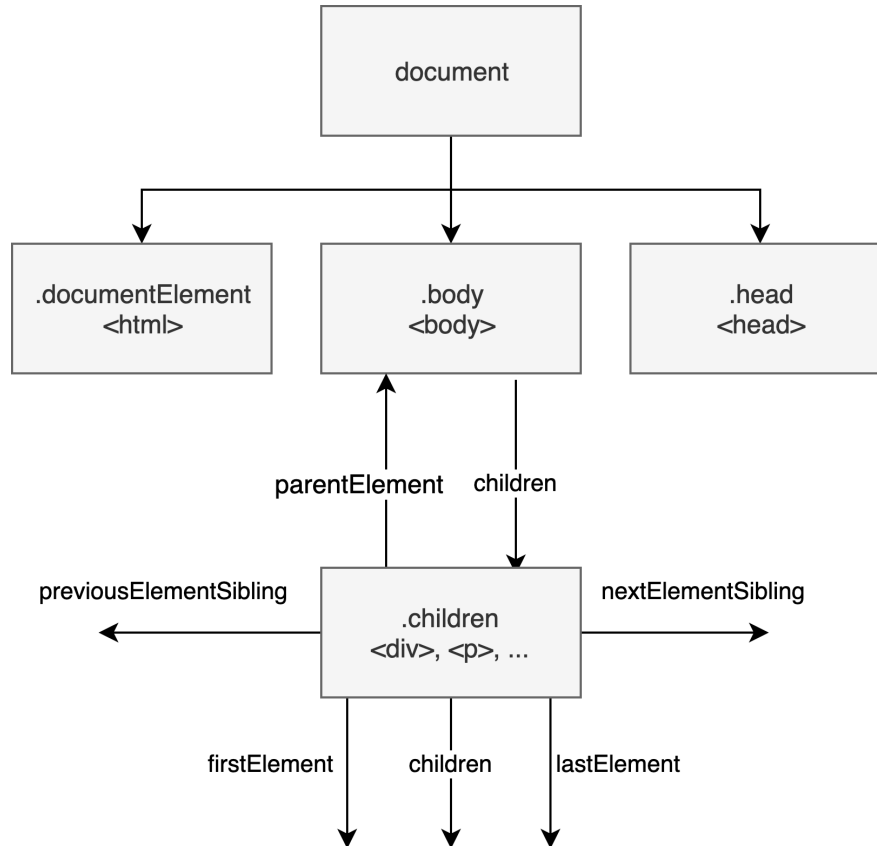
DOM – Zugriff auf DOM Nodes

Mögliche Node-Types:

<https://developer.mozilla.org/en-US/docs/Web/API/Node/nodeType>



DOM – Zugriff auf DOM Element Nodes



DOM – Methoden um DOM zu durchsuchen

snippets/dom-methods

Methode	Sucht via	Beispiel
querySelector (gibt erstes Element zurück)	Selector	<code>document.querySelector('ul > li:last-child')</code>
querySelectorAll (gibt alle Elemente als NodeList zurück)	Selector	<code>document.querySelectorAll('ul > li');</code>
getElementById	Id	<code>document.getElementById('myDiv');</code>
getElementsByName	Name	<code>document.getElementsByName('myDivName');</code>
getElementsByTagName	Tag	<code>document.getElementsByTagName('input');</code>
getElementsByClassName	Class	<code>document.getElementsByClassName('myCssClass');</code>

QuerySelector Cheatsheet: <https://gist.github.com/magicznyleszek/809a69dd05e1d5f12d01>
Vertiefung CSS heute Nachmittag!

DOM – Wichtige Node Properties

Methode	Beschreibung
nodeType	Typ des Nodes (numerischer Wert – siehe: https://developer.mozilla.org/en-US/docs/Web/API/Node/nodeType)
nodeName / tagName	Tag Name oder Beschreibung was es ist
innerHTML	HTML innerhalb vom selektierten Element
outerHTML	Vollständiges HTML von einem Element
textContent	Text von einem Element (alles minus HTML Tags)

DOM – Node Properties

- Je nach Element gibt es zusätzliche spezifische Properties
(z.B. input Element = HTMLInputElement hat weitere Properties)
- Beispiel IDL-Beschreibung in VS Code

DOM – Attribute & Properties

Der Browser parst das HTML, erstellt DOM Objekte für Tags, erkennt die (standard) Attribute und erstellt entsprechende DOM Properties dafür.

```
<body id="myBodyId" irgendetwas="das ist kein standard">
```

```
console.log(document.body.id); // myBodyId (property)
console.log(document.body.irdendetwas); // undefined, da nicht standard
console.log(document.body.getAttribute('id')); // myBodyId
console.log(document.body.getAttribute('irdendetwas')); // das ist kein standard
document.body.yay = {name: 'test'};
console.log(document.body.yay); // {name: "test"}
```

DOM – Document bearbeiten

examples/dom-manipulation

Methoden	Beschreibung
<code>document.createElement(tag)</code>	Erstellt ein Element mit dem entsprechenden Tag Namen <pre>const element = document.createElement('h1');</pre>
<code>element.cloneNode(deep)</code>	Klont ein Element (deep = true, der ganze Baum wird geklont)
<code>parent.appendChild(node)</code>	Fügt ein Child Element hinzu
<code>parent.removeChild(node);</code>	Löscht ein Child-Element
<code>parent.replaceChild(newElement, node);</code>	Ersetzt ein bestehendes Child-Element

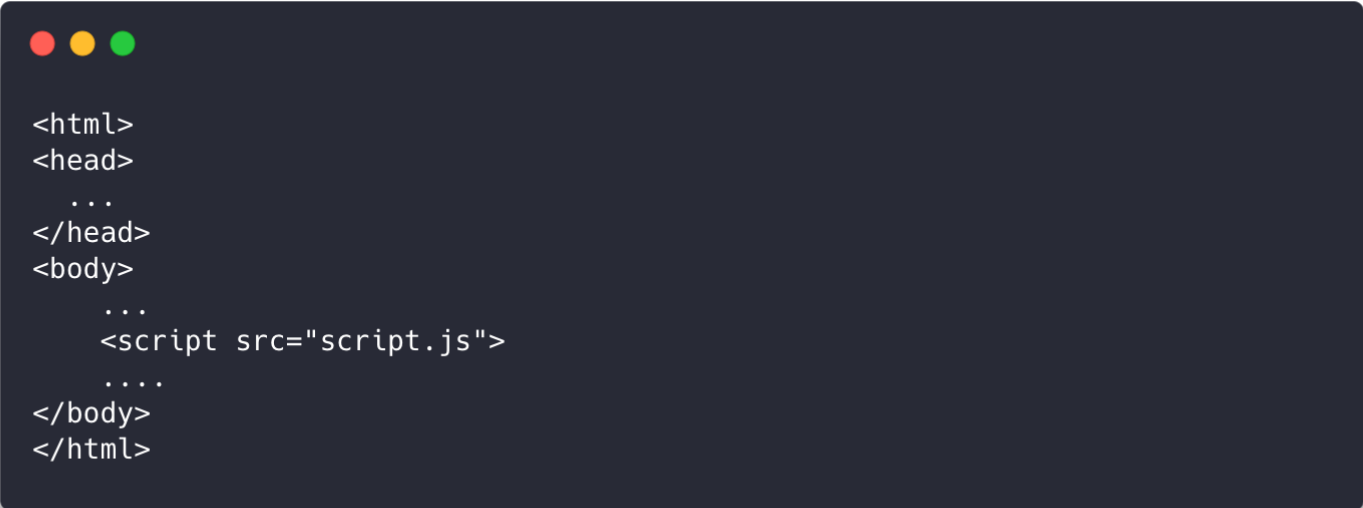
DOM – Document bearbeiten

snippets/dom-manipulation

Methode	Beschreibung
<code>node.append(...nodes oder Strings)</code>	Fügt einen neuen Node am Ende hinzu
<code>node.prepend</code>	Fügt einen Node am Anfang hinzu
<code>node.before</code>	Fügt einen Node vor dem Node hinzu
<code>node.after</code>	Fügt einen Node nach dem Node hinzu
<code>node.replaceWith</code>	Ersetzt den Node
<code>node.remove</code>	Löscht den Node
<u><code>element.insertAdjacentHTML</code></u>	Interpretiert das angegebene HTML und fügt den resultierenden Knoten an angegebener Position ein.
<u><code>document.write</code></u>	Fügt HTML hinzu bevor der Ladeprozess abgeschlossen ist.

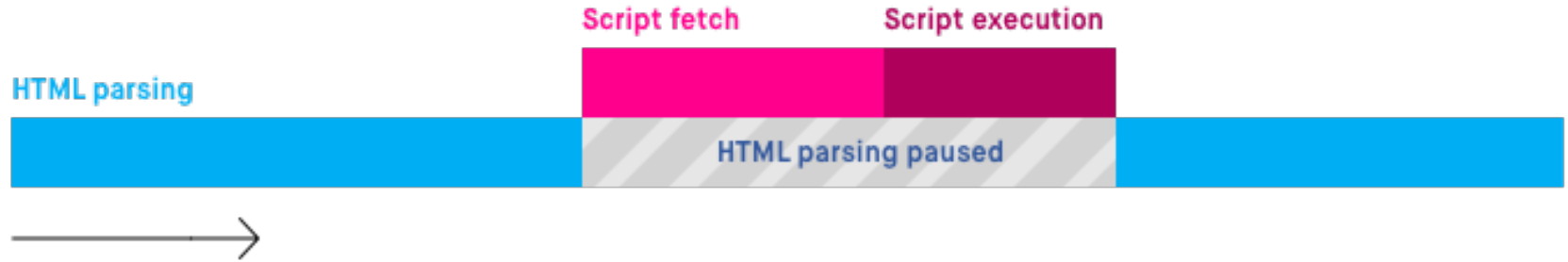
Document- & Resourceloading

- Wie lädt man JavaScript Dateien?



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

Document- & Resource loading



Document- & Resource loading

Aus dieser Thematik ergeben sich folgende zwei Problematiken:

- Scripts sehen den DOM “unter” ihnen noch nicht.
- Ist ein schwergewichtiges Script am Anfang des Markups, blockiert dieses das komplette Laden der Webseite und der User sieht nichts bis dieses fertig geladen wurde.

Was sind mögliche Lösungen?

Document- & Resourceloading - Lösungsansätze

- **Script an das Ende des HTMLs fügen**

```
<body>
  ...
  <script src="myscript.js"></script>
</body>
```

- Das Script wird erst am Schluss bearbeitet. Der HTML-Content kann bereits angezeigt werden.
- Problematik: Für lange HTML-Dokumente kann hier eine bemerkbare Verzögerung eintreten (insb. bei langsamen Verbindungen)

Document- & Resourceloading - Lösungsansätze

- **Script mit defer markieren**

```
<p>...</p>

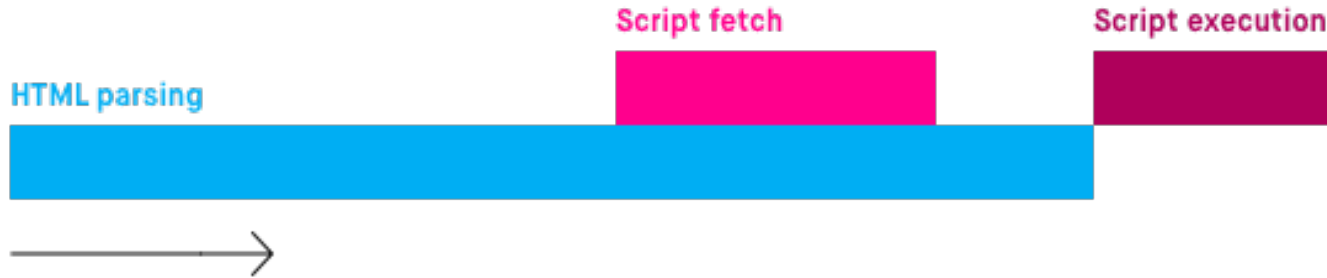
<script defer src="script2.js"></script>

<!-- wird sofort angezeigt -->
<p>...</p>
```

- `defer` teilt dem Browser mit, dass er mit dem Abarbeiten des HTMLs weitermachen und das Script im Hintergrund laden soll (nicht blockierend).
- Das Event `DOMContentLoaded` wird erst nach sämtlichen `defer` Scripts gefeuert.

Document- & Resourceloading - Lösungsansätze

- **Script mit `defer` markieren**



Document- & Resourceloading - Lösungsansätze

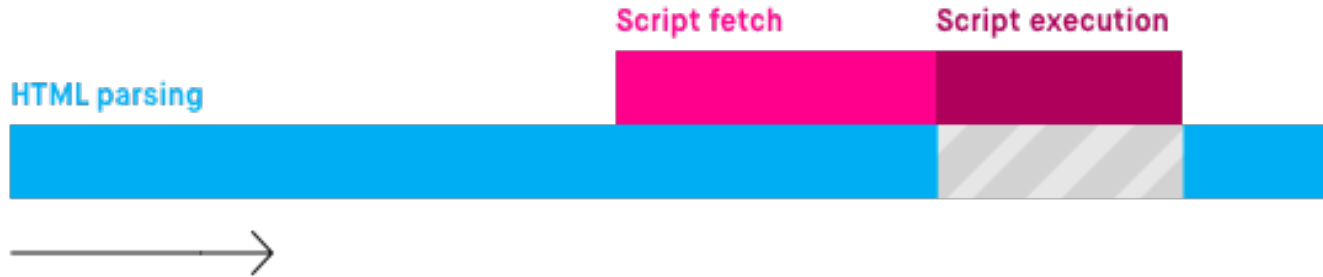
- **Script mit `async` markieren**

```
<p>...</p>  
  
<script async src="script2.js"></script>  
  
<!-- wird sofort angezeigt -->  
<p>...</p>
```

- `async` markiert das Laden des Scripts als komplett unabhängig, d.h.
 - > Es wird nicht auf die `async` markierten Scripts gewartet
 - > `DOMContentLoaded` und `async` Scripts warten nicht aufeinander

Document- & Resource loading - Lösungsansätze

- **Script mit `async` markieren**



Document- & Resourceloading - Lösungsansätze

Ansatz	Wann sollte der Ansatz eingesetzt werden?
Standard	<ul style="list-style-type: none">• Inline Script• “Kleines“ externes Script• Abhängigkeit zu anderen Scripts• DOM muss noch nicht vollständig geparst sein
Deferred	<ul style="list-style-type: none">• Externes Script• Abhängigkeiten unter den Scripts resp. Reihenfolge ist relevant• DOM muss geparst sein• (Falls möglich in den Head)
Async	<ul style="list-style-type: none">• “Self-Contained“ Script, d.h. keine Abhängigkeiten zu anderen Files oder zum geparsten DOM• Typischerweise 3rdParty Integration wie z.B. Analytics

Script Loading Demo

`./examples/script-loading`

Übungsaufgabe I (5')

- Selektiere folgende Elemente
 - > H1 Element
 - > Alle Studenten Elemente
 - > Bruno Element

Ordner: exercises/dom-selection

```
<body>
  <h1>JavaScript Exercises</h3>
  <div>
    Students
  </div>
  <ul>
    <li>Patrick</li>
    <li>Andreas</li>
    <li>Thomas</li>
    <li>Harald</li>
    <li>Bruno</li>
  </ul>
</body>
```

Übungsaufgabe II (5')

- Lese das Attribut `data-hslu-module` aus.
- Ändere den Wert auf `web-programming-lab`
- Gib den Wert erneut auf der Konsole aus

```
<body>  
  <div data-hslu-module="internet topics"></div>  
</body>
```

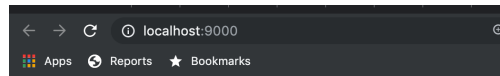
- Ordner: `exercises/dom-manipulation`

Übungsaufgabe III (20')

- Rendere den Technologie Radar ab einem JavaScript Objekt

Ordner: exercises/rendering-technologies

```
const techRadar = {
  Meta: {
    generatedAt: new Date(),
  },
  TechRadar: {
    Techniques: [
      'Micro Frontends',
      'Polyglot programming',
      'Secrets as a service',
      'Chaos Engineering',
    ],
    Tools: ['Cypress', 'Helm', 'Traefik', 'Humio'],
    Platforms: ['Contentful', 'AWS Fargate', 'InfluxDB'],
    'Languages & Frameworks': ['TypeScript', 'Rust'],
  },
};
```



Generated at 29/07/2019, 17:56:41

- Techniques
 - Micro Frontends
 - Polyglot programming
 - Secrets as a service
 - Chaos Engineering
- Tools
 - Cypress
 - Helm
 - Traefik
 - Humio
- Platforms
 - Contentful
 - AWS Fargate
 - InfluxDB
- Languages & Frameworks
 - TypeScript
 - Rust

Listen in HTML: <https://developer.mozilla.org/de/docs/Web/HTML/Element/ul>

Client-Side JavaScript I

Events, Form & Controls, Data Access via HTTP



DOM – Nützliche Events (Auszug)

Event
Mouse Events: <ul style="list-style-type: none">• click• contextmenu• mouseover / mouseout• mousedown / mouseup• mousemove
Form Events: <ul style="list-style-type: none">• submit• focus
Keyboard Events: <ul style="list-style-type: none">• keydown• keyup
Document Events <ul style="list-style-type: none">• DOMContentLoaded

examples/dom-events

Event Reference: <https://developer.mozilla.org/de/docs/Web/Events>

DOM – Events abonnieren

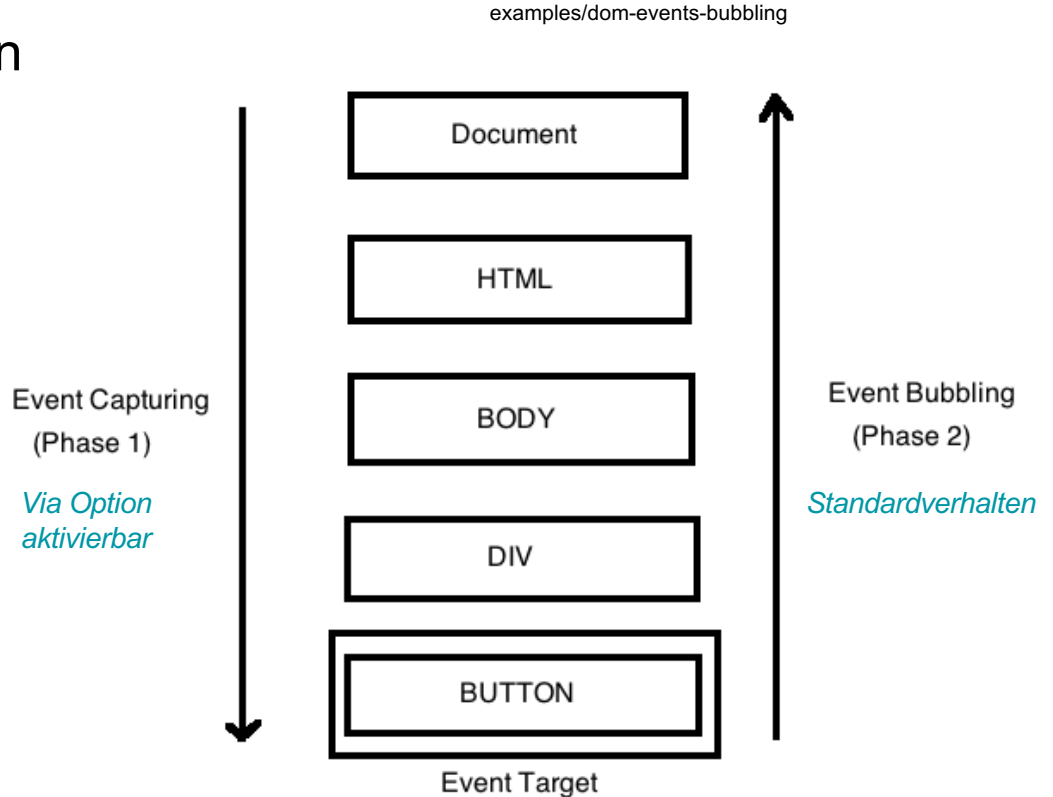
- HTML Attribut `onclick="..."`
- DOM Property `element.onclick = function(eventObj) {}`
- Via Methode `element.addEventListener(event, handler);`
entfernen via `element.removeEventListener`

```
function clickMe(event) {  
    console.log('clicked');  
    console.log(event);  
}
```

```
document.getElementById('myButton').onclick = clickMe;  
document.getElementById('myButton').addEventListener('click', clickMe);
```

DOM – Event Bubbling

- Jeder Event hat drei Phasen
 1. Event Capturing
 2. Event Target
 3. Event Bubbling



Forms & Control Elements I

- Deklaration eines Forms

examples/dom-form-declaration

```
<form id="myForm">  
  <input type="range" name="numberOfStudents" value=10>  
  <button type="submit">Send</button>  
</form>
```

- Zugriff auf Formular und Elemente

```
const myFormElement = document.forms.myForm;  
const numberOfStudentsElement = myFormElement.elements.numberOfStudents;
```

Forms & Control Elements II

- Behandlung vom Submit

examples/dom-form-declaration

```
const form = document.getElementById('myForm');
form.addEventListener('submit', (event) => {

    event.preventDefault();

    // Get & print form values
    let data = new FormData(form);
    for (const [name, value] of data) {
        console.log(name,value);
    }

});
```

Forms & Control Submit

- “Submit” triggert den Event, dass JavaScript die Werte des Formulars validieren resp. zum Server schicken soll.
- Es gibt drei Arten ein Formular zu “submitten”
 - > Via Click auf `<input type="submit">`
 - > Enter drücken innerhalb von einem Input-Control
 - > Via `form.submit()`; ← Hier wird der Submit Event nicht ausgelöst!
- Auf dem Form können Action und die entsprechende HTTP Methode (GET / POST) hinterlegt werden im Submit-Fall. Bei SPA's wird dies in der Regel nicht hinterlegt, sondern JavaScript übernimmt die Kommunikation. Das Standardverhalten wird via `event.preventDefault()` überschrieben.

Sending Form Data: https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Sending_and_retrieving_form_data

Sending Forms through JavaScript: https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Sending_forms_through_JavaScript

HTTP Data Access via fetch

- `fetch()` ist der de facto Standard um im Browser als Client mit einem Server via HTTP zu kommunizieren.
> Siehe auch <https://caniuse.com/#search=fetch>
- Syntax: `const promise = fetch(resource, [options])`
- Die Promise wird resolved sobald der Server antwortet. Der Zugriff verläuft via Response-Klasse.

HTTP Data Access via fetch

- GET

examples/fetch

```
async function getStudents() {  
  const response = await fetch(  
    'https://5d0e3cd1eba6ef0014561072.mockapi.io/students');  
  return response.json();  
}
```

- POST

examples/fetch

```
async function postStudents(student) {  
  const response = await fetch('https://<>>', {  
    method: 'POST',  
    headers: { 'Content-Type': 'application/json' },  
    body: JSON.stringify(student),  
  });  
  
  return response.json();  
}
```

HTTP Data Access via XHR

examples/xhr

```
// Setup our listener
const xhr = new XMLHttpRequest();

xhr.onload = function () {
  if (xhr.status >= 200 && xhr.status < 300) {
    console.log(JSON.parse(xhr.responseText));
  }
};

xhr.open('GET', 'https://5d0e3cd1eba6ef0014561072.mockapi.io/students');
xhr.send();
```

- **XMLHttpRequest** ist (wie fetch) ein built-in Browser Objekt, über welches man HTTP Requests absetzen kann. Unabhängig vom XML im Namen des Objektes kann man nicht nur Daten im XML-Format transportieren.
- Im Gegensatz zum neuen promise-basierten fetch() Ansatz ist XHR “event-getrieben” und benötigt so das Binden von Event Listern für das Abonnieren der Daten.

Übungsaufgabe Technology Radar (30')

- Lege auf [mockapi](#) einen Account an und modelliere die Technologie Resource, auf welcher Du die Technologien beziehen und neue Technologien hinzufügen kannst (GET und POST).
- Erstelle ein Formular für die Erstellung der Technologien
- Erstelle eine Liste mit der Anzeige der erstellten Technologien

Tipp für das Styling: Du kannst z.B. das [Pico.css Framework](#) verwenden.

Übungsvorlage: exercises/technology-radar

Add a technology

Name

Category

--Please choose a category--

Ring

--Please choose a category--

Description

Add technology

Technologies

Trial

ArgoCD

Kubernetes Cluster Management

Tools

Client-Side JavaScript I


Web Components




JS


Startseite

Was gibt's Neues?


 **Dominik Schürmann** @smilebox90 · 33 s
#SYMFONISK Regal-WiFi-Speaker - ab jetzt gibts die #Sonos-Lautsprecher bei IKEA 🤗


 SYMFONISK Regal-WiFi-Speaker - IKEA
IKEA - SYMFONISK, Regal-WiFi-Speaker, Der Regal-Speaker ist eine Gemeinschaftsproduktion ...
ikea.com


docToolchain hat retweetet

 **Darren Cooper** @dc7590 · 1 Min.
Nice article from @RalfDMueller on @jobpushy about his work at @dbsystel including docToolchain, a Gradle based AsciiDoc Toolchain for Software Architecture Documentation. Check out Github for more info
github.com/docToolchain/d...

pls RT I think such goodness needs sharing

 **Marc-Oliver Scheele (Mos)** @Jobpushy · 1. Aug.
Hintergründe, Meinungen und Tipps mit Mr. @RalfDMueller 🤗
jobpushy.de/blog/34/softwa...

 **Tara Vancil** @taravancil · 35 s
Antwort an @taravancil
Apparently I'm at a nude beach? Sick!

 **Deborah Kurata** @DeborahKurata · 41 s
Doing CRUD with Observable sequences? Merge a data stream with a "modified" action stream, when an action is emitted: put/post/delete, then reflect the change in the data stream immutably.

Code here: github.com/DeborahK/Angul...

Course here: app.pluralsight.com/library/course...

```
// Save the product via http
// And then modify the full list of products with scan.
productsWithCRUD$ = merge(
  this.productsWithCategory$,
  this.productModifiedAction$
    .pipe(
      concatMap(product => this.saveProduct(product)),
    )
    .pipe(
```

Twitter durchsuchen

Trends für dich

Trend in Schweizerische Eidgenossenschaft
Bitcoin
82.700 Tweets

Trend in Schweizerische Eidgenossenschaft
#Frankfurt
8.672 Tweets
Z ZEIT ONLINE twittert darüber

Trend in Schweizerische Eidgenossenschaft
Dybala
116.000 Tweets

Trend in Schweizerische Eidgenossenschaft
Syria
52.600 Tweets

Trend in Schweizerische Eidgenossenschaft
#Seehofer
2.001 Tweets
Neue Zürcher Zeitung und ZEIT ONLINE twittern darüber

Mehr anzeigen

Wem folgen?

 **Siemens Mobility** @SiemensMobility
Gesponsert [Folgen](#)

 **Oliver Drotbohm** @odrotbohm [Folgen](#)

 **Sean Larkin (廖肖恩)** @TheLarkinn [Folgen](#)

Mehr anzeigen

Bedingungen Datenschutzrichtlinien Cookies
Anzeigen-Info Mehr © 2019 Twitter, Inc.



Startseite

Entdecken

Mitteilungen

Nachrichten

Lesezeichen

Listen

Profil

Mehr

Twittern

Startseite



Was gibt's Neues?



Twittern



Dominik Schürmann @smilebox90 · 33 s
#SYMFONISK Regal-WiFi-Speaker - ab jetzt gibts die #Sonos-
Lautsprecher bei IKEA 🤖



SYMFONISK Regal-WiFi-Speaker - IKEA
IKEA - SYMFONISK, Regal-WiFi-Speaker, Der
Regal-Speaker ist eine Gemeinschaftsproduktion ...
ikea.com



docToolchain hat retweetet



Darren Cooper @dc7590 · 1 Min.
Nice article from @RalfDMueller on @jobpushy
about his work at @dbsystel including docToolchain, a Gradle based
AsciiDoc Toolchain for Software Architecture Documentation. Check out
Github for more info
github.com/docToolchain/d...

pls RT I think such goodness needs sharing



Marc-Oliver Scheele (Mos) @Jobpushy · 1. Aug.
Hintergründe, Meinungen und Tipps mit Mr. @RalfDMueller 🤖
jobpushy.de/blog/34/softwa...



Tara Vancil @taravancil · 35 s
Antwort an @taravancil
Apparently I'm at a nude beach? Sick!



Deborah Kurata @DeborahKurata · 41 s
Doing CRUD with Observable sequences? Merge a data stream with a
"modified" action stream, when an action is emitted: put/post/delete,
then reflect the change in the data stream immutably.

Code here: github.com/DeborahK/Angul...

Course here: app.pluralsight.com/library/course...

```
// Save the product via http
// And then modify the full list of products with scan.
productsWithCRUD$ = merge(
  this.productsWithCategory$,
  this.productModifiedAction$
    .pipe(
      concatMap(product => this.saveProduct(product)),
    )
    .pipe(
```

Twitter durchsuchen

Trends für dich



Trend in Schweizerische Eidgenossenschaft
Bitcoin
82.700 Tweets

Trend in Schweizerische Eidgenossenschaft
#Frankfurt
8.672 Tweets
Z ZEIT ONLINE twittert darüber

Trend in Schweizerische Eidgenossenschaft
Dybala
116.000 Tweets

Trend in Schweizerische Eidgenossenschaft
Syria
52.600 Tweets

Trend in Schweizerische Eidgenossenschaft
#Seehofer
2.001 Tweets
Neue Zürcher Zeitung und ZEIT ONLINE twittern
darüber

Mehr anzeigen

Wem folgen?



Siemens Mobility
@SiemensMobility
Gesponsert

Folgen



Oliver Drotbohm
@odrotbohm

Folgen



Sean Larkin (廖肖恩)
@TheLarkinn

Folgen

Mehr anzeigen

Bedingungen Datenschutzrichtlinien Cookies
Anzeigen-Info Mehr © 2019 Twitter, Inc.

Web Components

- [Web Components](#) sind Building Blocks von modernen Web Applikationen.
- Bestehen aus den folgenden Hauptkonzepten
 - > [Custom Elements](#) – Erweitere HTML und erstelle die eigenen Tags
 - > [Shadow DOM](#) – Style und Markup Entkopplung
 - > [HTML Templates](#) – Templates für die Erstellung von ähnlichen El.

Web Components – Custom Elements I

- Mit den Custom Elements können neue “HTML Elemente” erstellt resp. bestehende HTML Tags erweitert werden.
- So können wiederverwendbare Web Komponenten mit JavaScript erstellt werden.
- Dadurch gibt es weniger Code durch Modularisierung und mehr Wiederverwendung in unseren Web Applikationen.
- Das Markup wird dadurch lesbarer. So kann man mit dem Markup die Fachlichkeit deklarativ beschreiben.

Web Components – Custom Elements II

- Eigenes HTML Element erstellen

examples/web-components-custom-elements

```
class MyCustomElement extends HTMLElement {  
  constructor() {  
    super();  
    this.innerHTML = '<p>this is my custom element</p>';  
  }  
  // Wird aufgerufen wenn das Element dem DOM hinzugefügt wird  
  connectedCallback() {  
    console.log('custom element added to page');  
  }  
}  
  
window.customElements.define('my-custom-element', MyCustomElement);
```

```
<my-custom-element></my-custom-element>
```

Web Components – Shadow DOM

- Mittels dem Shadow DOM kann die Markup-Struktur, der Style sowie das Verhalten der Web Komponente von aussen geschützt resp. gekapselt werden.
- **Isolated DOM**
Der DOM von einer Web Component mit einem Shadow DOM ist somit “self contained”, z.B. `document.querySelector()` gibt nicht die Nodes des Shadow DOMs von der Web Component zurück.
- **Simplifies CSS**
Da auch das CSS direkt an den (Shadow-)DOM gekoppelt ist, muss nicht auf globale Naming Konflikte Rücksicht genommen werden.

Web Components – Shadow DOM

- Shadow DOM erstellen

```
class MyCustomElement extends HTMLElement {  
  constructor() {  
    super();  
    const shadowDom = this.attachShadow({mode: 'open'});  
    shadowDom.innerHTML = '<style>p {color: blue}</style><p>this is  
                           my custom element</p>';  
  }  
}
```

Web Components – HTML Templates

- [HTML Templates](#) werden verwendet, um Markup Strukturen wieder zu verwenden.
- HTML Templates bestehen aus dem `<template>` und `<slot>` Element.

```
<template id="technology-details">
  <p>Technology X</p>
</template>
```

examples/web-components-html-templates

- Das Template wird nicht gerendert, solange es nicht an den DOM gehängt wird. Es wird lediglich geparkt und auf Richtigkeit geprüft.

```
const template = document.getElementById('technology-details');
const templateContent = template.content;

const shadowRoot = this.attachShadow({ mode: 'open' })
  .appendChild(templateContent.cloneNode(true));
```

examples/web-components-html-templates

Web Components – HTML Templates

- Mittels `<slots>` können HTML Templates flexibel erweiterbar gemacht werden.

```
<template id="technology-details">
  <p><slot name="technology-name">Technology Placeholder</slot></p>
</template>
```

examples/web-components-html-templates

- Bei der Anwendung des Custom Elements können entsprechend die Slots über das slot Attribut überschrieben werden.

```
<techradar-technology-details>
  <span slot="technology-name">ArgoCD</span>
</techradar-technology-details>
```

examples/web-components-html-templates

Web Components – Kommunikation

- Mittels Events kann zwischen den Web Components kommuniziert werden.

```
connectedCallback() {  
  const buttonEl = this.shadowDom.querySelector('#exampleButton');  
  buttonEl.addEventListener('click', () => {  
    this.dispatchEvent(new Event('customelement-one-click'));  
  });  
}
```

```
connectedCallback() {  
  const customElementOneEl = document.querySelector('my-custom-element-one');  
  customElementOneEl.addEventListener('customelement-one-click', () => {  
    this.shadowDom.innerHTML = `${this.shadowDom.innerHTML} <p>${new Date()}</p>`;  
  });  
}
```

Web Components – Unterstützende (leichtgewichtige) Frameworks

- Lit - <https://lit.dev/>

Skip web components boilerplate code, small footprint, future ready.



- Stencil - <https://stenciljs.com/>

Stencil is an opensource compiler that generates standards-compliant web components.



Übungsaufgabe Web Components (30')

- Erstelle für die Techradar Übung entsprechende Web Components

Übung: exercise/technology-radar-web-components

```
<body>
  <techradar-add-technology></techradar-add-technology>
  <techradar-list-technologies></tech-radar-list-technologies>
</body>
```

- Tipp: Für die Komponenten übergreifende Kommunikation kann der Event-Mechanismus verwendet werden.

```
<body>
  <techradar-add-technology></techradar-add-technology>
  <techradar-list-technologies></tech-radar-list-technologies>
</body>
```

Client-Side JavaScript I

Deployment



JS

Zusatz: Building & Deployment for Production

- Deploye deinen Tech Radar auf [Netlify](#) oder [Vercel](#)



netlify

