Vorlesung

TECHNISCHE HOCHSCHULE LÜBECK

Achtung: Mit Dart2 haben sich einige Änderungen ergeben. U.a. die Umstellung von optionaler auf statische Typisierung. Ggf. funktionieren daher einige Beispiele in den noch für Dart1 erstellten Folien nicht mehr. Wenn Sie so etwas finden, bitte sofort mailen!

Web-Technologien

Progressive Web Apps (PWA)

Client-seitiger Zustand (Web Storage)

Asynchrone Programmierung und Unittesting für HTML

TECHNISCHE HOCHSCHULE LÜBECK

Prof. Dr. rer. nat.
Nane Kratzke
Praktische Informatik und
betriebliche Informationssysteme

- **Raum: 17-0.10**
- **Tel.:** 0451 300 5549
- **Email: nane.kratzke@th-luebeck.de**
- **Sprechzeiten: Mi. 14:00 bis 16:00**
  (nach Mail-Voranmeldung, oder jederzeit mit Termin)

Bei Ihnen entstand(en) folgende Frage(n)

TECHNISCHE HOCHSCHULE LÜBECK

*„Und zwar geht es darum, welche Möglichkeiten es gibt [...] **Daten** (z.B. Spielstände) **lokal zu speichern** – am besten auch über einen Browserneustart hinweg – und andererseits die **ganze Webapplikation im Cache** zu speichern, so dass sie auch offline verfügbar ist.“*

1. *Wie kann mit Dart **asynchrone** Programmierung realisiert werden ?*
2. *Wie kann in Dart die **GUI getestet** werden?*

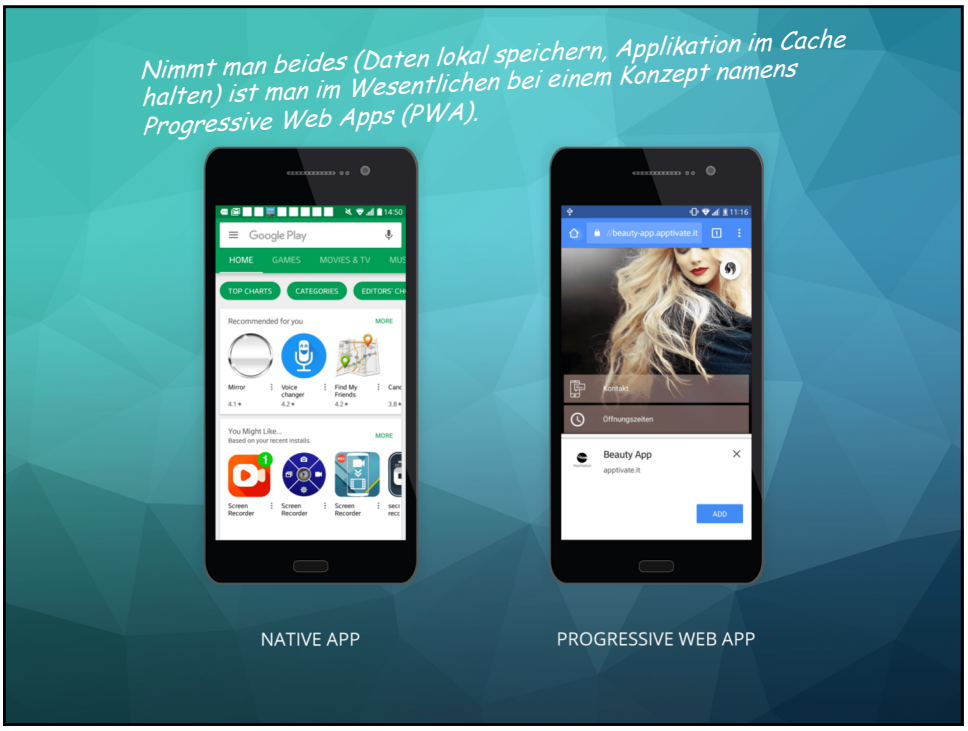
Prof. Dr. rer. nat. Nane Kratzke
Praktische Informatik und betriebliche Informationssysteme
3

Bei Ihnen entstand(en) folgende Frage(n)

TECHNISCHE HOCHSCHULE LÜBECK

*„Und zwar geht es darum, welche Möglichkeiten es gibt [...] **Daten** (z.B. Spielstände) **lokal zu speichern** – am besten auch über einen Browserneustart hinweg – und andererseits die **ganze Webapplikation im Cache** zu speichern, so dass sie auch offline verfügbar ist.“*

Prof. Dr. rer. nat. Nane Kratzke
Praktische Informatik und betriebliche Informationssysteme
4



Progressive Web Apps (PWA)

TECHNISCHE HOCHSCHULE LÜBECK

PWA sind Plattform-unabhängige Web Applikationen, die eine vergleichbare User Experience wie native Applikationen bieten (sollen).

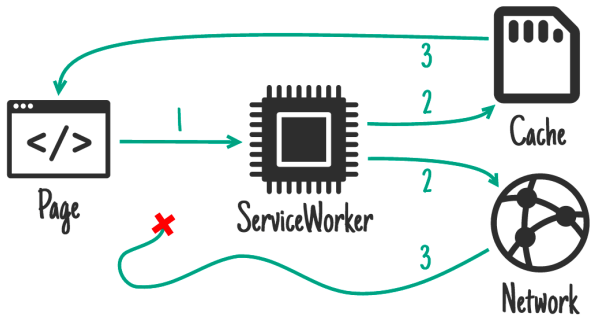
Eine der bekannteren PWA dürfte der Telegram Messenger sein.

- **Discoverable** — *PWA werden im WWW bereitgestellt und sind damit für Suchmaschinen zugänglich.*
- **Installable** — *PWA können wie andere Applikationen mittels Homescreen Icons versehen und „installiert“ werden.*
- **Linkable** — *PWA können wie Webseiten verlinkt werden.*
- **Network independent** — *PWA benötigen keine (permanente) Netzwerkverbindung.*
- **Progressive** — *PWA sollten sich auf modernen Browsern wie native Applikationen anfüllen, aber auf älteren Browsern wie Webseiten zu bedienen sein.*
- **Re-engageable** — *Native Plattformen ermöglichen es Nutzer mittels Updates oder Nachrichten zu „aktivieren“. Web Notifications und Push APIs ermöglichen ein vergleichbares Erlebnis für PWA.*
- **Responsive** — *PWA unterstützen Desktop, Mobile, Tablet, und mehr.*
- **Safe** — *Die Web Infrastruktur bietet sichere Bereitstellungs- und Updatemöglichkeiten (HTTPS).*

Prof. Dr. rer. nat. Nane Kratzke
Praktische Informatik und betriebliche Informationssysteme
6

Client-seitiges Web Caching

TECHNISCHE HOCHSCHULE LÜBECK



Idee: Bei wiederholter Benutzung einer Seite wird diese aus dem lokalen Cache geladen und ist daher viel schneller geladen.

Möglichkeiten State Client-seitig zu speichern

TECHNISCHE HOCHSCHULE LÜBECK

http://caniuse.com 18.05.2018

Key-Value bis ca. 5 MB

API	Data Model	Persistence	Browser Support	Transactions	Sync/Async
File system	Byte stream	device	52%	No	Async
Local Storage	key/value	device	93%	No	Sync
Session Storage	key/value	session	93%	No	Sync
Cookies	structured	device	100%	No	Sync
WebSQL	structured	device	77%	Yes	Async
Cache	key/value	device	60%	No	Async
IndexedDB	hybrid	device	83%	Yes	Async

Text bis ca. 4KB

Relationale DB (beliebige Größe)

Quelle: https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/

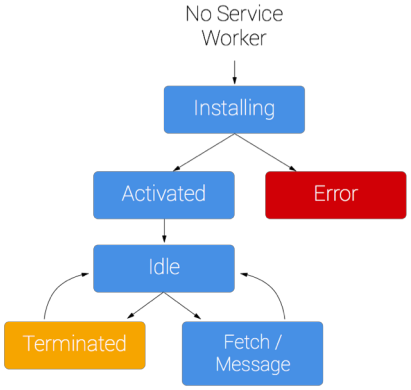
Service Worker, PWA

Key-Value-Store (beliebige Größe)

Service Workers
in a Nutshell

TECHNISCHE HOCHSCHULE LÜBECK

- Die Service Worker API dient zum Client-seitigen Cachen von statischen Assets (CSS, HTML, JavaScript, etc.) einer Seite.
- Seiten können mittels eines Service Workers Caching-Strategien definieren (und nicht zentral der Browser) und auf den Anwendungsfall der Seite abstimmen.
- Wenn alle registrierten Assets gecacht werden konnten, wird die Installation erfolgreich beendet.
- Wenn ein Asset nicht gecacht werden konnte (z.B. wegen Netzwerkproblemen) scheitert die Installation (wird aber beim nächsten Ladeversuch wiederholt).
- Installierte Service Worker überwachen alle Zugriffe und versuchen Zugriffe auf gecachte Assets aus dem Cache bereitzustellen und so zeitintensive Netzzugriffe zu vermeiden.
- **Interessanter Nebeneffekt: Werden alle Assets einer Seite „gecached“ kann die Seite sogar offline genutzt werden (da keine Netzzugriffe erforderlich werden).**



Prof. Dr. rer. nat. Nane Kratzke
Praktische Informatik und betriebliche Informationssysteme
9

Client-seitiges Web Caching

TECHNISCHE HOCHSCHULE LÜBECK

„Cache bezeichnet […] einen schnellen Puffer-Speicher, der Zugriffe auf ein langsames Hintergrundmedium […] reduziert. Daten, die bereits geladen wurden, verbleiben im Cache, so dass sie bei späterem Bedarf schneller […] abgerufen werden können. Auch Daten, die vermutlich bald benötigt werden, können vorab vom Hintergrundmedium abgerufen und im Cache bereitgestellt werden (read-ahead).

Wikipedia: https://de.wikipedia.org/wiki/Cache

„There are only two hard things in computer science: cache invalidation, naming things, and of-by-one errors.“

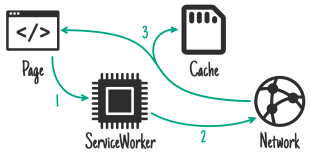
Phil Karlton, Netscape (nach Martin Fowler)

Prof. Dr. rer. nat. Nane Kratzke
Praktische Informatik und betriebliche Informationssysteme
10

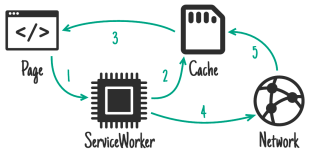
Caching Strategien für Service Worker
Aus dem „Offline Cookbook“

TECHNISCHE HOCHSCHULE LÜBECK

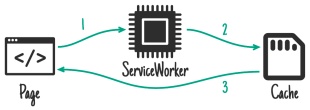
On network response pattern



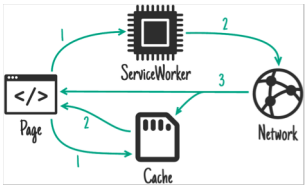
Stale-while-revalidate pattern



Cache-only pattern



Cache-then-network pattern



Und viele mehr ... *(insgesamt können sie jeweils 8 Store and Serving Pattern kombinieren, ergibt also insgesamt 64 Caching Strategien)*.

Siehe: **https://developers.google.com/web/fundamentals/instant-and-offline/offline-cookbook/**

Progressive Web Apps
Mit Dart

Die Bibliothek PWA kann hier Entscheidungen für sie abnehmen, die für viele Anwendungsfälle funktionieren.

Progressive Web Apps

- **Bestimmen** der zu cachenden **statischen Assets**, die für die Offline Benutzung erforderlich sind.
- **Erstellen** und **Registrieren** eines Service Workers, der diesen Cache aufbaut (**pre-population**).
- Sicherstellen, dass der Service Worker **Änderungen** (Updates) an den statischen Assets erkennt und neue Versionen im Cache platziert (**updating**).

Nutzen der Bibliothek PWA. Ergänzen Sie die Bibliothek einfach in ihrer pubspec.yaml.

```
dependencies:
  pwa: ">=0.1.0 <0.2.0"
```

Importieren der Bibliothek PWA und anlegen des Service Workers, weitere Details hier: https://pub.dartlang.org/packages/pwa

```
import 'package:pwa/client.dart' as pwa;

main() {
  new pwa.Client();
}
```

Mittels dieses Befehls (Kommandozeile) scannt PWA das Repository nach zu cachenden Assets für eine Offline Benutzung. Ändert sich die Menge dieser Assets, muss dieser Befehl erneut aufgerufen werden!

```
> pub run pwa
```

Den Effekt sehen Sie hier:

TECHNISCHE HOCHSCHULE LÜBECK

1. *Laden Sie das Spiel ganz gewöhnlich in ihren Mobile-Browser (Chrome for Android, iOS)*
2. *Schalten Sie den **Flugmodus** in ihrem Smartphone **an**.*
3. *Laden Sie das Spiel erneut in ihrem Mobile-Browser (das Spiel sollte nun laufen und der Browser sie nicht auf eine fehlende Internet-Verbindung hinweisen)*

https://github.com/nkratzke/dartmotion

Prof. Dr. rer. nat. Nane Kratzke
Praktische Informatik und betriebliche Informationssysteme
13

Einschränkungen:

TECHNISCHE HOCHSCHULE LÜBECK

- Service Worker funktionieren **nur** mit **HTTPS**
  - Um sicherzustellen, dass die statischen Assets auch (unverändert durch Dritte) von der Quelle stammen von der sie abgerufen werden.
  - In Github Pages können Sie sehr einfach HTTPS aktivieren.

- Service Worker funktionieren **nicht** mit den **Inkognito** Modes von Browsern
  - aus offensichtlichen Gründen
  - Der Cacheinhalt würde ja (Teile) des Browserverlaufs offenlegen
  - *Das gilt auch für die folgenden Möglichkeiten Storage client-seitig zu speichern.*

Prof. Dr. rer. nat. Nane Kratzke
Praktische Informatik und betriebliche Informationssysteme
14

Bei Ihnen entstand(en) folgende Frage(n)

TECHNISCHE HOCHSCHULE LÜBECK

*„Und zwar geht es darum, welche Möglichkeiten es gibt [...] **Daten** (z.B. Spielstände) **lokal zu speichern** – am besten auch über einen Browserneustart hinweg – und andererseits die **ganze Webapplikation im Cache** zu speichern, so dass sie auch offline verfügbar ist.“*

Web Storage API

TECHNISCHE HOCHSCHULE LÜBECK

- Local Storage ist ein Konzept vergleichbar mit dem „betagten“ Konzept der Cookies.
- Jedoch werden lokal (d.h. client-seitig) nicht einfach nur Zeichenketten gespeichert sondern Key-Value-Paare (String -> String).
- Ferner wird der Local Storage nicht an Server übermittelt (anders als Cookies).
- Zwei Ausprägungsarten:
  - **localStorage**: Daten werden dauerhaft persistent gehalten. Sie überdauern damit auch das Schließen eines Browser Windows/Tabs.
  - **sessionStorage**: Daten werden für die Dauer einer Session (d.h. solange das Browser Window/Tab geöffnet ist) gespeichert.

Web Storage mit Dart

TECHNISCHE HOCHSCHULE LÜBECK

```
// Web storage
// - window.localStorage (data will never be deleted and stored "forever")
// - window.sessionStorage (data will be deleted when the page is closed)
// - Both kinds of storage are simply a key value map Map<String, String>
//
Storage local = window.localStorage;
Storage session = window.sessionStorage;
```

Sowohl `session` als auch `local` sind Maps des Typs `Map<String, String>` und können wie folgt lesend und schreibend genutzt werden.

```
Element output = querySelector('#output');
Storage storage = window.localStorage;

storage['highscore'] = '2048'; // Achtung nur Zeichenketten !!!

output.append(
    new Element.div()..innerHtml = "Highscore: ${storage['highscore']}"
);
```


Web Storage mit Dart
und JSON Encoding und Decoding

TECHNISCHE HOCHSCHULE LÜBECK

Da es „lästig“ ist, komplexere Zustandsstrukturen immer als einzelne Key-Value-Paare zu speichern und String encodings und decodings vorzunehmen, kann man einen State auch als komplexere Key-Value-Map halten und sich des JSON Encodings und Decodings bedienen.

```
import 'dart:convert';


Element output = querySelector('#output');
Storage storage = window.localStorage;

storage['game'] = jsonEncode({
  'name': 'Max Mustermann',
  'scores': [2048, 64, 128, 512, 32, 16, 4096, 64]
});

final state = jsonDecode(storage['game']);
state['scores'].add(32716); // Yeah, new highscore !!!
state['scores'].sort();

final welcome = new Element.div()..innerHtml = "Welcome: ${state['name']}";
final score = new Element.div()..innerHtml = "Highscore: ${state['scores'].last}";

output.append(welcome);
output.append(score);
```

Diese Strukturen können beliebig komplex und tief geschachtelt sein. Alle Schlüssel müssen Strings sein, alle Werte müssen JSON encodable sein (selbst geschriebene Klassen müssen die toJson() Methode implementieren).

Bei Ihnen entstand(en) folgende Frage(n)

TECHNISCHE HOCHSCHULE LÜBECK

1. *Wie kann mit Dart asynchrone Programmierung realisiert werden ?*
2. *Wie kann in Dart die GUI getestet werden?*

Antwort: Vermutlich programmieren sie bereits asynchron. Dart ist (client-seitig) per se schon asynchron in weiten Teilen realisiert.

Beispiel: Alle DOM-Tree Events werden zum Beispiel in Form von Streams asynchron verarbeitet (ohne dass sie sich darüber bislang explizit Gedanken gemacht haben).

I/O Funktionen (z.B. `http.get()`) sind meist auch **`async`**hron definiert (lange Ladezeiten).

Asynchron programmieren

TECHNISCHE HOCHSCHULE LÜBECK

Dart führt (wie jede andere imperative Programmiersprache auch) Anweisungen sequentiell aus. Die sequentielle Ausführung lassen sich mit Kontrollstrukturen (Schleifen, Bedingte Anweisungen, etc.) „umlenken“, jedoch werden immer zwei Anweisungen hintereinander ausgeführt und grundsätzlich existiert nur ein Ausführungsthread.

Insbesondere in der Webprogrammierung bedeutet sequentielle Ausführung von Anweisungen häufig „stockende Reaktionen“ auf Benutzerinteraktionen, da der Ausführungsthread gerade „anderes zu tun hat“.

Asynchrone Programmierung bedeutet diese sequentielle Anweisungsabfolge zu durchbrechen und Anweisungen bspw. nur dann auszuführen, wenn Ereignisse eingetreten sind.

Dart sieht hierzu u.a. folgende Konzepte vor:

Streams

Alle DOM-Tree Events sind nichts weitere als Streams.

Futures

async / await

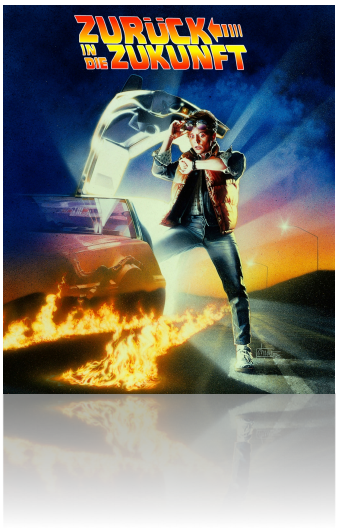
Futures

TECHNISCHE HOCHSCHULE LÜBECK

*„Eine **Future** […] bezeichnet in der Programmierung einen Platzhalter […] für ein Ergebnis, das noch nicht bekannt ist, meist weil seine Berechnung noch nicht abgeschlossen ist.*

*Eine Future ist meist das Ergebnis eines **asynchronen** Aufrufs einer Funktion oder einer Methode und kann verwendet werden, um auf das Ergebnis zuzugreifen, sobald es verfügbar ist. […] Das Konzept der Futures wurde 1977 […] von Henry G. Baker und Carl Hewitt vorgestellt.“*

Quelle: Wikipedia, Future (Programmierung)



`await` / `async` seit Dart 1.9 (Teil I)

TECHNISCHE HOCHSCHULE LÜBECK

Seit Dart 1.9 sind die Schlüsselworte **async** und **await** hinzugekommen. Diese vereinfachen es Methoden mit langer Laufzeit asynchron zu definieren und auf Methodenergebnisse asynchroner Methoden zu warten.

```
// Eine klassische synchrone Methode.
num fibo(num n) {
  if (n == 0) return 0;
  if (n == 1) return 1;
  return fib(n-1) + fib(n-2);
}
```

```
// Asynchrone Variante (async/await)
Future<num> fib (num n) async {
  if (n == 0) return 0;
  if (n == 1) return 1;
  return await fib(n-1) + await fib(n-2);
}
```

```
// Klassischer synchroner Aufruf
int n = fibo(10);
```

*Es bleibt damit manchmal das Problem auf in der Zukunft liegende Ereignisse (z.B. das Lesen einer Datei von einem Server) warten zu müssen. Hierfür sieht Dart die Schlüsselwörter **await** vor. Asynchrone Methoden können mittels **async** deklariert werden. Sie liefern ein Future auf einen Wert zurück.*

```
// Klassischer Aufruf (vor 1.9)
fib (10).then((result) {
  int n = result;
});
```

```
// Neuer Aufruf (mit await, seit Dart 1.9)
int n = await fib(10);
```

`await` / `async` seit Dart 1.9 (Teil II)

TECHNISCHE HOCHSCHULE LÜBECK

Mit den Schlüsselworten **async** und **await** kann auch auf mehrere Futures gewartet werden. Hierzu kombiniert man einfach das bereits bekannte `Future.wait` mit `await`.

```
// Asynchrone Variante (async/await)
Future<num> fibAsync(num n) async {
  if (n == 0) return 0;
  if (n == 1) return 1;
  return await fib(n-1) + await fib(n-2);
}
```

```
// Alles asynchrone Aufrufe!
final f1 = fibAsync(1);
final f2 = fibAsync(2);
final f4 = fibAsync(4);
final f8 = fibAsync(8);

// Warten auf deren Ergebnisse
List<num> results = await Future.wait([f1, f2, f4, f8]);
print("Summe: ${results.reduce((a, b) => a + b) }");
```


Mehr Details finden sich im Moodle Kurs

TECHNISCHE HOCHSCHULE LÜBECK

- Unit 3: Dart Teil II (dart:async)

1. *Wie kann mit Dart asynchrone Programmierung realisiert werden ?*
2. *Wie kann in **Dart** die GUI getestet werden?*

Ihre GUI ist eine HTML Seite. Dafür gibt es jede Menge Tools ...

Denken Sie outside the box ;-)

GUI == Oberfläche

TECHNISCHE HOCHSCHULE LÜBECK

- Es gibt in Dart Unit-Tests wie z.B. das test package (vglb. JUnit).
- Das kann man nutzen, das eignet sich aber ggf. besser um ihr Model und die Spiellogik zu testen.

- Wieso nutzen sie keine Frameworks, die sich auf HTML spezialisiert haben (allerdings nicht zum Dart Universum gehören)?
- HTMLUnit zum Beispiel ist eine Library mit der Sie in Java HTML Unittests schreiben können.

HtmlUnit http://htmlunit.sourceforge.net/

HTMLUnit Beispiele

TECHNISCHE HOCHSCHULE LÜBECK

Automatisierung von Tests in Java:

```
@Test
public void homePage() throws Exception {
    try (final WebClient webClient = new WebClient()) {
        final HtmlPage page = webClient.getPage("http://htmlunit.sourceforge.net");
        Assert.assertEquals("HtmlUnit - Welcome to HtmlUnit", page.getTitleText());

        final String pageAsXml = page.asXml();
        Assert.assertTrue(pageAsXml.contains("<body class=\"composite\">"));

        final String pageAsText = page.asText();
        Assert.assertTrue(pageAsText.contains("Support for the HTTP and HTTPS protocols"));
    }
}
```

Browser-spezifische Tests:

```
@Test
public void homePage_Firefox() throws Exception {
    try (final WebClient webClient = new WebClient(BrowserVersion.FIREFOX_52)) {
        final HtmlPage page = webClient.getPage("http://htmlunit.sourceforge.net");
        Assert.assertEquals("HtmlUnit - Welcome to HtmlUnit", page.getTitleText());
    }
}
```

HTMLUnit Beispiele

TECHNISCHE HOCHSCHULE LÜBECK

Selektion von HTML Elementen mittels CSS Selektoren:

```
@Test
public void cssSelector() throws Exception {
    try (final WebClient webClient = new WebClient()) {
        final HtmlPage page = webClient.getPage("http://htmlunit.sourceforge.net");

        //get list of all divs
        final DomNodeList<DomNode> divs = page.querySelectorAll("div");
        for (DomNode div : divs) {
            ....
        }

        //get div which has the id 'breadcrumbs'
        final DomNode div = page.querySelector("div#breadcrumbs");
    }
}
```

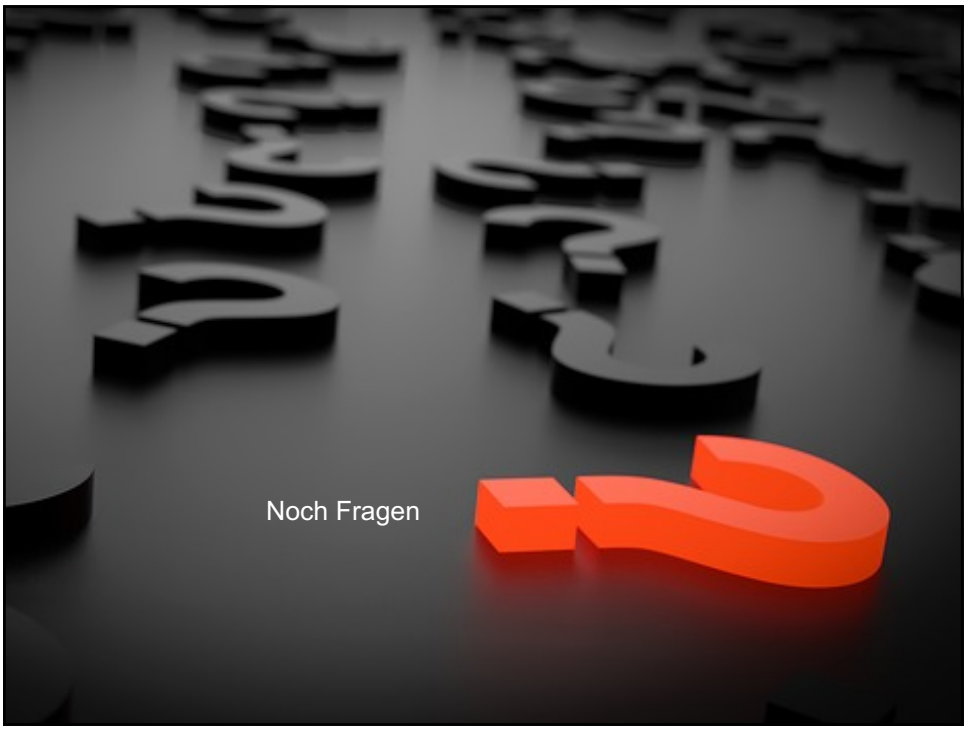

HTMLUnit Beispiele

TECHNISCHE HOCHSCHULE LÜBECK

Fernsteuern von Events in einem Browser:

Hier am Beispiel eines Formulars und Klicks

```
@Test
public void submittingForm() throws Exception {
    try (final WebClient webClient = new WebClient()) {

        // Get the first page
        final HtmlPage page1 = webClient.getPage("http://some_url");

        // Get the form that we are dealing with and within that form,
        // find the submit button and the field that we want to change.
        final HtmlForm form = page1.getFormByName("myform");

        final HtmlSubmitInput button = form.getInputByName("submitbutton");
        final HtmlTextInput textField = form.getInputByName("userid");

        // Change the value of the text field
        textField.type("root");

        // Now submit the form by clicking the button and get back the second page.
        final HtmlPage page2 = button.click();
    }
}
```


Noch Fragen