

**Semester Praxis Arbeit
Web-Engineering I
Benoteter Programmamentwurf
TINF19-ITA**

Juergen Schneider
DHBW Stuttgart

Benotung

- **Anpassungen aufgrund der Remote Sessions**
- **Benotung:**
 - Es können max 100 Punkte erreicht werden
 - 10 Punkte für die komplette Bearbeitung der praktischen Übungen (verpflichtend)
 - **Das Vorhandensein der JavaScript Übung 2 wird als Teil der Klausurarbeit mit geliefert und soweit bewertet**
 - 90 Punkte für den benoteten Programm Entwurf am Ende des Kurses = **Klausur** (verpflichtend)
 - Es kann aber auch bis zu 10 Punkte zusätzlich geben (für herausragende Zusätze)
 - 10 Punkte (als Sicherheitszone) mit einem Referat (optional)
 - Ablauf :
 - Referate bis 17.11
 - see Referate_Praktische_Arbeit.pdf im Folder Benotung
 - Vorstellung benoteter Program Entwurf (Klausur) heute den 6.11
 - Optional **1.12** Dienstag Nachmittag Zoom für Fragen (keine Anwesenheitspflicht)
 - Abgabe Klausur am 5.12.2020 (23:59:59 !!!!!)
 - Benotung bis 20.12.2020

Praxis-Arbeit Beschreibung

- Wir erweitern unsere DHBW oder eigene Home Page um die folgenden möglichen Elemente
 1. Anzeige von Aktien Verlaufskurven **(2)**
 2. Anzeige von Wetter Daten **(2)**
 3. Implementierung des Group Chats **(2)**
 4. Anzeige von mehreren Nachrichten als RSS Feed **(2)**
 5. Gelesene Wikipedia Daten werden per Click vorgelesen. Server-seitige Implementierung unter Ausnutzung von Ausnutzung eines Cloud basierten Übersetzer **(2)**
 6. Gelesene Wikipedia Daten werden per Click vorgelesen. Client-seitige Implementierung unter Ausnutzung von zum Beispiel https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API **(1)**
 7. Anzeige von Instagram Bilder **(3)**
 8. Anzeige von Spotify Podcasts **(3)**
 9. Baue mit XMLHttpRequest ein eigenes Promise based REST Interface **(1)**
 10. Benutzer Login mit dem Einsatz von Cookies (ein Password für alle Nutzer) **(2)**
 11. Individuelles Benutzer Login mit Registrierung und Speicherung der Credentials **(3)**
 12. Einsatz von jQuery (anstatt native JavaScript) **(1)**
 13. Einsatz von Typescript (Client oder Server Side) **(1)**
 14. Benutzung von AngularJS <https://angularjs.org/> **(3)**
 15. Benutzung von Vue.js <https://vuejs.org/> **(3)**
 16. Eigene CSS mit Benutzung von Sass **(2)**

Anmerkungen Apache/PHP oder Flask/Python sind möglich geben aber keine extra Credits.

Praxis-Arbeit Auswahl

- Auswahl
 - (x) sind credits points für die jeweiligen Elemente
 - Teamarbeit max. zwei Personen sind erlaubt
 - Um für die 90 Punkte eligible zu sein müssen dann **10 credit points** erreicht werden.
 - Individuelle Abgabe verpflichtend.
 - individuelle Dokumentation mit der Beschreibung der eigenen Teile (und ‘Subteile’)
 - Bitte Team Partner benennen
 - Einzelarbeiten erwünscht
 - Um für die 90 Punkte eligible zu sein müssen dann **6 credit points** erreicht werden.
 - Individuelle Abgabe..

Praxis Arbeit. Logistik

- Dokumentation
 - Bitte **Matrikelnummer** deutlich angeben (Folder Name !!)
 - Bitte dokumentieren welche Elemente Ihr bearbeitet habt.
 - Datenfluss Diagramm mit den wichtigsten Objekten und Beziehungen (nur die wichtigsten Objekte)
 - Installationsbeschreibung. (**Komplette und genaue Beschreibung mit NPM Install oder äquivalent**)
 - Erfahrungsbericht, was ging gut, schlecht, was habe ich gelernt
- Lauffähiger Code
 - Die lauffähige JavaScript Directory (als Nachweis der Codierung)
 - Screenshoots (falls Juergen Schneider Probleme hat, obige Scripts zum Laufen zu bringen)
 - Responsive CSS
 - Ansprechendes Layout
- **Abgabe und Bewertung für jeden einzelnen (per email an jschnei1@lehre.dhbw-stuttgart.de, 5.12 2020 EoB.. d.h. 23.59 :-))**
 - Beschreibung und Code Library haben die Matrikelnummer als Teil des Namens
 - Beschreibung als MS (.doc oder .docx) preferable aber .pdf
 - Code als Folder Anhang an die email oder über eine Art dropbox. (ohne node_modules)
 - Optional **1.12** Dienstag Nachmittag Zoom für Fragen (keine Anwesenheitspflicht)
 - Oder bei Bedarf individuell Zoom oder email

Wann werden Punkte abgezogen

- Siehe PDF KlaususrarbeitenTINF19ITA im Benotung Folder..

Wetter Daten Anzeige

- Anzeige von Wetter Daten als neues Popup
 - **Über die Navigation gibt es einen neuen Wetter Dienst, der nach Eingabe einer Stadt das heutige Wetter mit dem Ausblick über die nächsten 5 Tage aufzeigt.**
 - Es gibt Online Wetter Dienste die REST APIs for free anbieten
 - Zum Beispiel : <https://openweathermap.org/api>
 - (Man braucht einen free account und hat API call limits, you may test it with ‘fake’ data)
 - **Die Benutzung von vorgefertigten Widgets ist nicht erlaubt**
 - Dieser Teil legt sehr viel Wert auf die ‘coole’ Aufbereitung der gelesenen Wetter Daten.
 - Ideal ist auch die **Benutzung von Icons** um die Information auch bildlich zu unterstützen
 - <https://openweathermap.org/weather-conditions>

Aktienkurse anzeigen

Aufzeigen des aktuellen Aktienwertes und des historischen Verlaufes dreier Firmen (evtl. US Firmen auf NASDAQ) , Ausgabe Tageswert, Historie Max, Min etc

- Es wird empfohlen
 - <https://www.alphavantage.co/>
 - Als ein free to use REST API zu benutzen
 - <https://www.chartjs.org/>
 - Als Graphic Tool zu benutzen
- Alternativen sind natürlich immer möglich



RSS Feed

- Einlesen mehrere (mindestens 2) Nachrichten Dienste (e.g. ARD, Heise) um die Top 10 Nachrichten anzuzeigen
- RSS news (or blogs) sind typischerweise im XML format.

```
<channel>
  <title>.... the title of the ARD News </title>
  <link>.... link to the official ARD News Home page </link>
  <pubDate>.. date of news.. </pubDate>
  <copyright> .... </copyright>
  ....
  <item>
    <title>.... News title..... </title>
    <pubDate>.. date of news.. </pubDate>
    <encoded>.. here we have the whole part as HTML </encoded>
    <description> ...News text.. </description>
    <guid> link to the actual news with more detail</guid>
    </item>
  <item>...</item>
</channel>
```

- Nachdem Einlesen der Daten sollten die Daten ‘cool’ aufbereitet werden

tagesschau.de - Die Nachrichten der ARD
© ARD-aktuell / tagesschau.de

Presseschau zum Iran: Spontane Wut - keine Revolution
Datum : Tue, 02 Jan 2018 10:42:33 +0100

Folgt auf die Proteste im Iran der Machtwechsel? Kommentatoren deutscher Zeitungen sind skeptisch, dass sich aus der spontan geäußerten Wut eine Revolution entwickelt. Und sie kritisieren die Zurückhaltung der Europäer.
[Weiterlesen...](#)



Silvester: Rettungskräfte mit Böllern und Steinen beworfen
Datum : Tue, 02 Jan 2018 09:32:13 +0100

Böller, Flaschen, Steine: In mehreren deutschen Städten sind in der Silvesternacht Einsatzkräfte angegriffen worden. In Berlin wurde die Besatzung eines Rettungswagens mit Schusswaffen bedroht. Bundesjustizminister Maas sprach von "völlig inakzeptablen Angriffen". Von Iris Marx.
[Weiterlesen...](#)



RSS Feed

- API's are described here https://www.w3schools.com/xml/dom_intro.asp
- **The task is also to get the image URL which is part of CDATA construct.**
- Some other useful link
 - <https://stackoverflow.com/questions/6674322/how-to-get-values-inside-cdatavalues-using-php-dom>
 - Think about how you can access the image which is part of the CDATA construct (in HTML) but not in XML.

Implementierung des Group Chats

Siehe : JavaScript Exercise 3 im Übungen Folder

Text to Speech (Server Side)

- Gelesene Wikipedia Daten werden vorgelesen
 - Dazu wird ein Cloud Service von IBM oder einem anderen genutzt. Diesem Service wird ein Text String zur Verfügung gestellt der diesen Text ‘synthesized’ und in einer lokale (e.g. .mp3) Audio File ablegt.
 - Es wird empfohlen als Text und Vorlesesprache Deutsch zu benutzen
 - Dazu muss die lokale Node.js Implementierung (myWebServer.js) um den Zugriff auf diesen ‘Text to Speech’ Service erweitert werden.
 - Der IBM Service wird von mir zur Nutzung zu Verfügung gestellt (ist für mich kostenpflichtig, also seid bitte sparsam hier)
 - <https://cloud.ibm.com/apidocs/text-to-speech/text-to-speech?code=node>
 - Die Zugangsdaten werden per request über email bei mir erfragt.
- Der TTS Service erzeugt eine Audio File
 - die entweder direkt über das
 - <audio src= “url zum TTS Service” Attribute requestiert und ‘ge-streamed’ werden kann
 - oder die Audio File wird Server Side abgespeichert e.g. /public/audio/myaudio.mp3
 - Das SRC attribute wird beschrieben wenn die Datei vorhanden ist
 - audioobject.src = “/public/audio/myaudio.mp3?...”
 - Wir brauchen eine Internet Recherche wie wir mit ‘ge-cacheten’ Audio Dateien umgehen müssen

Text to Speech (Client Side)

- Der selbe Funktionsumfang wie vorher beschrieben, nur wird hier direkt auf der Client Side programmiert
- https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API

Anzeige von Instagram Bilder oder Spotify Audios

- Instagram und Spotify (wie auch andere) bitten die Möglichkeiten öffentliche Inhalte eines Accounts in eine Web Seite zu integrieren. Dazu müssen von uns REST Calls zur Verfügung gestellt werden.
- Dieses Element macht nur Sinn wenn man Zugriff auf einen Account hat und bereit ist sich bei im jeweiligen Developer Portal zu registrieren (ist kostenlos aber nicht umsonst)
- Die jeweilige Beschreibung der Interfaces befindet sich hier
 - <https://developer.spotify.com>
 - <https://developers.facebook.com/docs/instagram-basic-display-api>
- Aufgabe ist es diese Inhalte abzugreifen und (über einen Click) sichtbar zu machen.
- Die Schwierigkeit liegt darin sich die jeweiligen Autorisierungen und holen und auch zu bewahren. Ein typischer Flow ist
 - Agreement des Account owners (der/die muss zustimmen bei Spotify, oder ich kenn das Account Passwort bei Instagram). Zurück kommt eine URL die ich für den nächsten Schritt brauche)
 - Über eine Account id und ein Client Secret hole ich mir dann einen access token
 - Dieser access token expired oft sehr schnell aber erlaubt mir einen Refresh Token und einen längerfristigen Access Token zu holen (Die Verfahren unterscheidet sich hier ein wenig)
 - Mit dem Refresh Token kann man periodisch den eigentlich access Token erneuern
 - Mit dem Access Token kann man dann (oft sehr einfach) auf die Daten zugreifen)
 - Die Anforderung des Access Token und der Daten Zugriff müssen programmiert werden (besser Server Side)
 - Die einmaligen Aufrufe könnten auch über cURL oder bei Get über den Browser erfolgen (und müssen daher nicht nachgewiesen werden)

Take the XMLHttpRequest Object and build a Promise wrapper (Promisify)

- Promise Object werden immer zur Standard Benutzung in JavaScript (Server und Client Side) Es ermöglicht uns auch die Nutzung des asynch/await pattern
- Dieses Element nimmt den XMLHttpRequest object und baut daraus ein Promise Object mit resolve and reject Funktionen für erfolgreiches oder fehlerhaftes Prozessieren einer HTTP Anfrage. Bei Aufruf der resolve und reject Funktionen werden entsprechend auch Daten übergeben.
 - <https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest>
 - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise
- Dabei wäre es auch sinnvoll eine neue Funktion zu kreieren die Parameter annimmt. Diese Funktion ist dann unsere neuer myHTTPService. Diese Funktion returned dann das Promise Objekt welcher im Konstrukt den XMLHttpRequest implementiert
 - myHTTPService function (url,method,data) { return new Promise (function(resolve , reject) { ... XMLHttpRequest Code.. mit url, method, data als Daten in der Execution... }); }
 - Aufruf. myHTTPService("/myServices","get","xyz");
- Eine einfachste Variante (< 50 LOCs) ist ausreichend.

Einfachen User Login mit Cookies

- User Login und die Benutzung von einem ‘login’ cookie.
 - Wir definieren 2 unterschiedliche Benutzer Gruppen
 - **Poor Users** dürfen die Homepage nur lesen aber nichts Aktives tun (also kein Chat benutzen oder Wikipedia oder Wetter abfragen)
 - **Rich Users** dürfen alles
 - Über die Existenz eines Cookies wird festgestellt ob es sich um ein ‘Rich’ oder ein ‘Poor’ User handelt oder ob es sich um den ersten Aufruf handelt.
 - Beim ersten Aufruf muss sich der Benutzer identifizieren (und wird über ein Password als ‘rich’ oder ‘poor’ identifiziert (und autorisiert)
- Im Header der Page zeigen wir den Login Status an. Gibt es keinen Benutzer müssen wir ein Login panel ausgeben.
- Der Login fragt ein Password ab und klassifiziert entsprechend
- Ein Re-Login wird als zusätzliche Option angeboten
- Ein Re-Login wird forciert wenn der Browser gestoppt und wieder gestartet wurde (expiration des cookies)
- Die Cookie Implementierung muss im Node.js Server erfolgen (mögliche Packages sind cookie parser (`require('cookie-parser')`) und basic authentication in express (`require('express-basic-auth')`))

User Management für individuelle Benutzer

- Dieses Element ähnelt dem vorherigen Element, aber der Benutzer muss sich individuell anmelden. Um das zu ermöglichen muss er/sie sich zuerst registrieren. (Email, eigenes Password und mögliches Profile) (Server Side)
- Dabei ist zu beachten daß, der Server nun in der Lage sein muss, eine Art Session State zu verwalten um (zumindest theoretisch) die verschiedenen Nutzer Profile zu unterscheiden.
- Unterschiedliche Benutzer werden als rich und poor (oder auch anders) klassifiziert (Profile) und der Server verhindert (über das Session Cookie) den Zugriff auf Daten.
 - Mögliche NPM Packages :
 - <https://stackabuse.com/a-sqlite-tutorial-with-node-js/> (Abspeichern der Benutzer)
 - <https://www.npmjs.com/package/express-session> (Session Cookie)
 - <https://www.npmjs.com/package/cookie-parser>
- Möglicher Bonus (Man implementiert die “Passwort vergessen” Funktion über email

Vue.js Framework

- Achtung diese Übung geht schon etwas tiefer in der JavaScript Technologien, aber dieser Exkurs ist sehr lehrreich.
- <https://vuejs.org/>
- Es wird empfohlen (aber nicht vorgeschrieben) den Single File Component Ansatz zu wählen.
 - <https://vuejs.org/v2/guide/single-file-components.html>
 - Dort befindet sich auch eine einfache Installations Beschreibung.
 - Siehe Artikel “Users New to Module Build Systems in JavaScript” mit der Installation des Vue CLI <https://cli.vuejs.org/>
 - Mit dem vue create Kommando kann ein vue Projekt erstellt werden. (alle Default annehmen). <https://cli.vuejs.org/guide/creating-a-project.html#vue-create>
 - Ansonsten braucht man nur noch den **npm run serve** Befehl für die Entwicklung und den **npm run build** Befehl für die Produktion
 - Nach dem Create Kommando (mit allen Default) gibt es schon eine fertige kleine Vue App in dem Subfolder / src.
 - **main.js** ist die Start Routine . Sie definiert die Vue. Instance. Diese Instanz importiert eine erste Komponente **App.vue** (als single file). Diese Komponente könnte man als Root Komponente benutzen die dann weitere Komponenten in subfolder /src/components/ importiert (in diesem Fall **HelloWorld.vue**)
 - Diese Basis kann nun benutzt werden um die eigenen Erweiterungen zu implementieren

Vue.js Framework

- Herausforderungen
 - Wird ein REST API nötig muss gegebenenfalls ein zweiter Node.js aktiviert werden, der explizit diese Requests abarbeitet
 - Der Übergang zur Produktion
 - Durch den npm run build Befehl werden alle Daten (Bundles) in die Directory /dist geschrieben.
 - Diese Directory muss dann für dem Produktion Node.js erreichbar sein.
 - Mit express über folgende Anweisung
 - `app.use(express.static(__dirname + '/dist'));`
 - Der Produktion Node.js Server muss selbst geschrieben werden und kann mit den REST API Server von oben kombiniert werden.

