Moderne Webentwicklung

Eine Einführung in Javascript

Organisatorisches

Code Beispiele findet ihr auf Github (https://github.com/wasdJens/workshop-javascript-introduction)

Die Präsentation ist auch auf Github

Anregung? Feedback? Bitte per Mail an mich:

Jens.Reiner@plan-b-gmbh.com

Etwas ist Unklar? Fragt einfach

Organisatorisches

Um die Beispiele Lokal auszuführen, empfiehlt es sich folgende Tools zu installieren

VSCode https://code.visualstudio.com/

Node https://nodejs.org/en/

Postman https://www.postman.com/

Ziele

Grundlagen von Javascript

REST Grundlagen

Schnittstelle definieren

Frontend Konzepte

Was ist nicht Teil?

Datenbank Konzepte & Anbindung

Vertiefung von Javascript Modulen (ES Module VS CommonJS z.B.)

Authentifierzung

Themen

Background (5 Minuten)

Javascript Überblick (20 Minuten)

REST Überblick (5 Minuten)

Create a Beer API with Express (15 Minuten)

Themen

Frontend Anwendung (15 Minuten)

Frontend Konzepte (5 Minuten)

Ausblick (5 Minuten)

Background

Warum eigentlich Webentwicklung?

Warum Webentwicklung

Immer mehr Digitale Produkte basieren auf Webanwendung

Der Browser bietet eine Platform für die meisten Endgeräte

Viele fertige Frameworks für Alltags Probleme

Einfach als Einstieg

Warum Javascript?

Einfache Programmiersprache zum Einstieg

Scripting Sprache erlaubt den Einsatz auch außerhalb von der klassischen Webentwicklung

Unterstützt verschiedene Programmierstile (Object Orientiert, Imperativ und Funktionale Programmierung)

Frontend & Backend

In der Webentwicklung unterscheidet man meistens zwischen Frontend und Backend

Frontend – User Interface Entwicklung (Presentation layer)

Backend – Daten Anbindung und Schnittstellen Entwicklung (Data Access Layer)

Kommt aus dem Client-Server Model

Javascript Überblick

Javascript Basics schnell und einfach erklärt

Quellcode Beispiele

Basics/index.js

Module, Packages und NPM

Javascript Code kann in Module unterteilt werden

Module können Klassen, Funktionen, Variablen (etc.) beinhalten

Module können auf andere Module zugreifen (Import und Export)

Packages

Packages sind eine Sammlung an Modulen die weiterverteilt werden können

Packages können auch wieder Abhängigkeiten haben

Damit man nicht Packete selbst Verwalten muss braucht man einen Packet Manager

NPM

Der bekannteste Javascript Package Manager

Man findet so gut wie alles auf NPM (Leider...)

https://www.npmjs.com/

NPM schauen wir uns später noch genauer an

BUNDLER

Was ist denn jetzt ein Bundler?

Module müssen irgendwann "zusammengepackt" werden um verteilt zu werden

Wieso?

Weil im Webbereich es Unpraktisch ist 25 Javascript Dateien zu schicken ;)

BUNDLER

Was macht ein Bundler?

Einfach gesagt: Aus all euren Javascript Dateien eine einzelne Javascript Datei (Und vieles mehr)

Muss man Bundler verstehen?

Am Anfang nein – Generell sollten immer fertige Bundler Konfigurationen verwendet werden (Code Rein  und fertig)

NODEJS

Normalerweise wird Javascript direkt im Browser ausgeführt

NodeJS ist eine Runtime die auch außerhalb vom Browser Javascript ausführen kann

REST Grundlagen

Schnittstellen für das Web definieren

REST

Eine Architektur für verteilte Systeme

REST beschreibt immer Ressourcen, die über eine URI abgerufen werden können

HTTP kommt in den meisten Fällen zum Einsatz für REST Ressourcen

Unterschiedliche Clients können auf die selben Ressourcen zugreifen

HTTP Methoden (Vereinfacht)

Für den Zugriff auf REST Ressourcen gibt es folgende Methoden

GET – Frage eine Ressource an

POST – Lege eine neue Ressource an

PUT – Update eine bestehende Ressource

HTTP Methoden (Vereinfacht)

Für den Zugriff auf REST Ressourcen gibt es folgende Methoden

DELETE - Lösche eine Ressource

Warum REST?

REST als API Schnittstelle zwischen Client und Server (Front- und Backend)

REST Client Unabhängig - d.h. auf die selben Ressourcen kann auch mit anderen Technologien zugegriffen werden

Warum APIs?

Kommunikation zwischen verteilten Systemen

Datenbank Anbindungen, Daten verarbeiten etc. Erfolgt auf Server seite (heavy operations)

Clients (Frontends) können beliebig ausgetauscht werden

Create your first API

Eine Schnittstelle mit NodeJS, ExpressJS und Javascript erstellen

Ziel:

Eine REST Schnittstelle mit Javascript erstellen

Die Schnittstelle soll später in unserem Frontend eingesetzt werden

Schnittstellen Design

Als erstes sollte man immer definieren wie die Schnittstelle aussehen soll

Welche Daten werden benötigt?

Welche Funktionalität soll unterstützt werden?

Was sollte der Server übernehmen und was der Client?

Schnittstellen Design

Wir möchten eine Bier API erstellen. Mit folgenden Funktionen:

- Ein neues Bier anlegen
- Jedes Bier soll einen Geschmack haben
- Jedes Bier hat einen Namen
- Doppelte Biere sollen nicht angelegt werden
- Biere können später geändert werden (Geschmack)
- Biere sollen in einer Liste dargestellt werden
- Biere können auch wieder gelöscht werden

Schnittstellen Design

Anforderung auf HTTP & REST Schnittstellen Mappen

GET beers – Alle Biere laden

POST beers – Ein neues Bier erstellen

PUT beers/:id – Ein Bier updaten

DELETE beers/:id – Ein Bier entfernen

Schnittstelle mit Node umsetzen

Wir erstellen ein neues Node Projekt

Wir installieren das Express Framework

Wir definieren unsere Schnittstellen

Schnittstelle mit Node umsetzen

Live Coding Beispiel

API ausprobieren

Live Beispiel mit Postman

Frontend Development

Eine React Anwendung erstellen und die Bier API konsumieren

Background

Frontend Entwicklung hat in den letzten Jahren immer mehr an Bedeutung gewonnen

Stand heute werden die meisten Frontends mit fertigen Frontend Frameworks entwickelt

Im Folgenden betrachten wir nur Single Page Applications (SPA)

Warum Frontends?

Darstellung von Daten

Interaktion mit Daten

Grafische Oberflächen für den Endanwender

Frontend Frameworks Übersicht

Angular

React

Vue

Komponenten

Jedes moderne Framework ist Component basiert

Components sind einzelne UI Bausteine die unabhängig voneinander verwendet werden können

Components können wiederverwendet werden

Props

Components können properties übergeben bekommen (Props)

Props können Components beeinflussen

Props ermöglichen es einfache re-usable components zu bauen

Components können z.B. User Input als Prop bekommen, aber immer dasselbe UI anzeigen

Frontend Beispiel

React Anwendung

Ausblick

& Next Steps

Ausblick

Frameworks wie Express bieten eine super Möglichkeit für schnelle Prototypen

NodeJS eignet sich vor allem aufgrund derselben Programmiersprache als Backend

Im Javascript Universium gibt es viele fertige Frameworks, um schnell vollständige Anwendungen zu bauen

Ausblick

Selber ausprobieren und mit den Frameworks etwas eigenes bauen

RESTful APIs lesen können und zu definieren sind wichtige Fähigkeiten in der Webentwicklung