



Web-Programmierung und Web-Frameworks

Hintergrundvortrag Gruppe 1

Projektseminar Wintersemester 2017/2018
Adrian Abendroth, Jakob Braun, Philipp Hoberg,
Youri Kaminsky, Maximilian Streubel,
Betreuer: Matthias Bauer

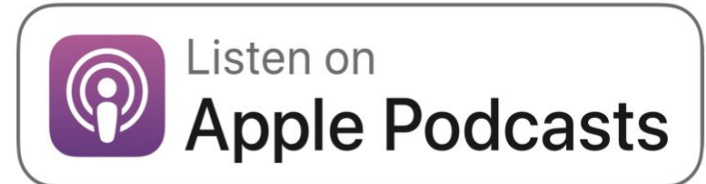
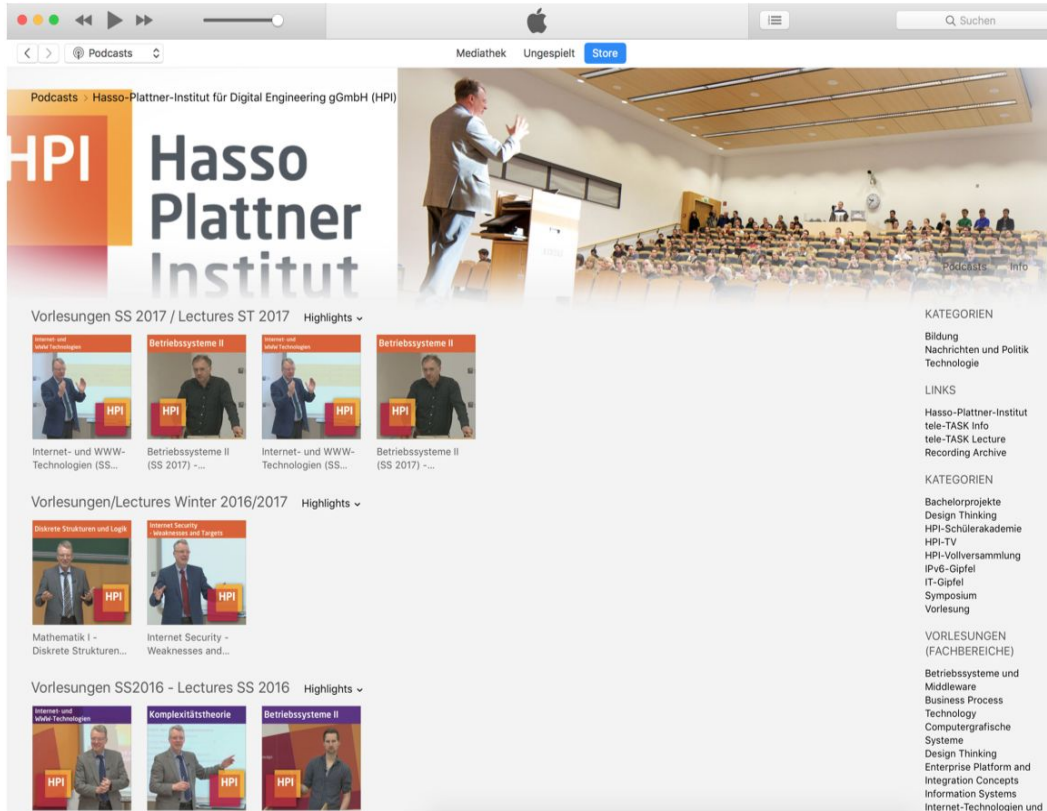
Gliederung

1. Einführung
 - 1.1. Das Projekt
 - 1.2. Anforderungen an das Projekt
2. Kommunikations- und Toolplanung
 - 2.1. Unsere Tools
 - 2.2. Praxisbeispiel Docker
3. Front-End
 - 3.1. Wire-Frames
 - 3.2. Mock-Ups

Gliederung

- 4. Datenvisualisierung
 - 4.1. Warum Datenvisualisierung
 - 4.2. Vorstellung: D3.js
 - 4.3. Vorstellung: Chart.js
 - 4.4. Vor- und Nachteile
 - 4.5. SVG vs. Canvas
 - 4.6. Code-Beispiel
- 5. Back-End
 - 5.1. Node.js
- 6. Momentane Architektur
- 7. Literaturverzeichnis

Einführung



Das Projekt

tele-TASK @ iTunes Podcasts

Ziel: Auswertung und Anzeigen der Zugriffs-Statistiken der HPI Podcasts auf iTunes als WebApp (ggf. integriert in tele-TASK-Portal)

Backend

- Upload der CSV-Datei mit den Auswertungen per Webformular und per Ablage im Suchordner
- Parsen der Auswertung
- Eintragen der Ergebnisse in Datenbank

Frontend

- Auswertung und Bereitstellung der Statistiken (Zahlen, Berichte, Visualisierung)

Anforderungen an das Projekt

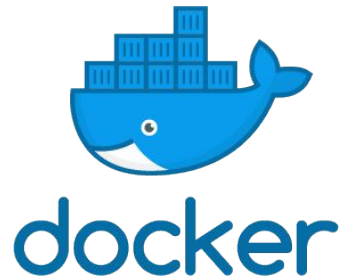
Must-Have:

- Authentifizierung
- Rechtevergabe
- Visualisierung der Daten
- Auswählen von Zeiträumen
- Automatische Erstellung eines Reportes
- Software-Test

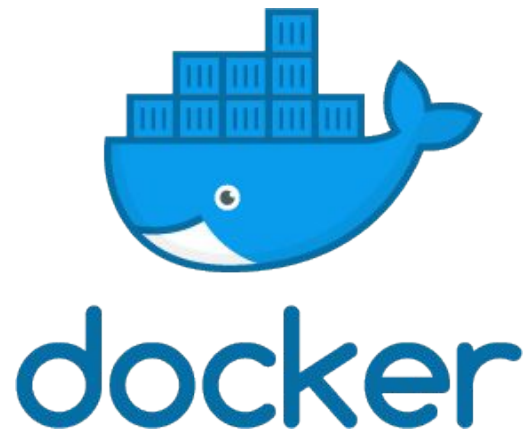
Nice-to-Have:

- HPI-Open-ID Integration
- TeleTask Integration

Kommunikations- und Tool-Planung



Docker



Docker - Unser Setup



Docker - Vor- und Nachteile

Vorteile:

- Einfaches Setup
- Offline Arbeiten
- Parallel Arbeiten
- Aktueller Stand im GIT
- Softwaretests gut möglich

Nachteile:

- Overhead
- Docker-Setup nötig

CSS



HTML



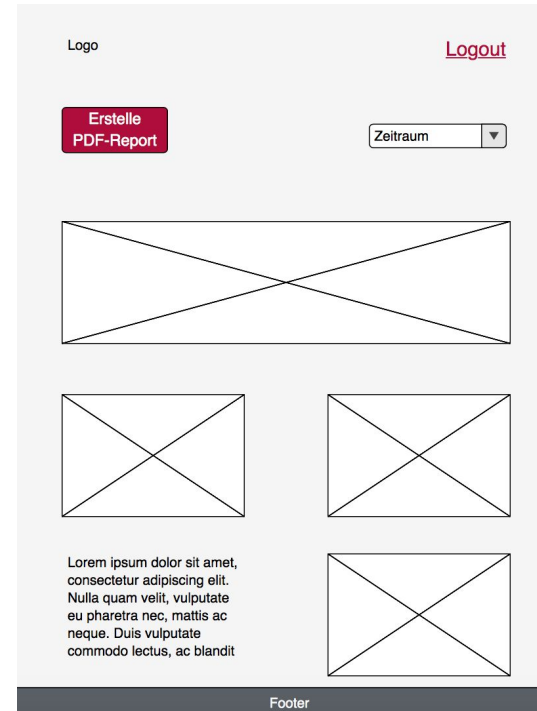
JS



Wireframes

Wireframes:

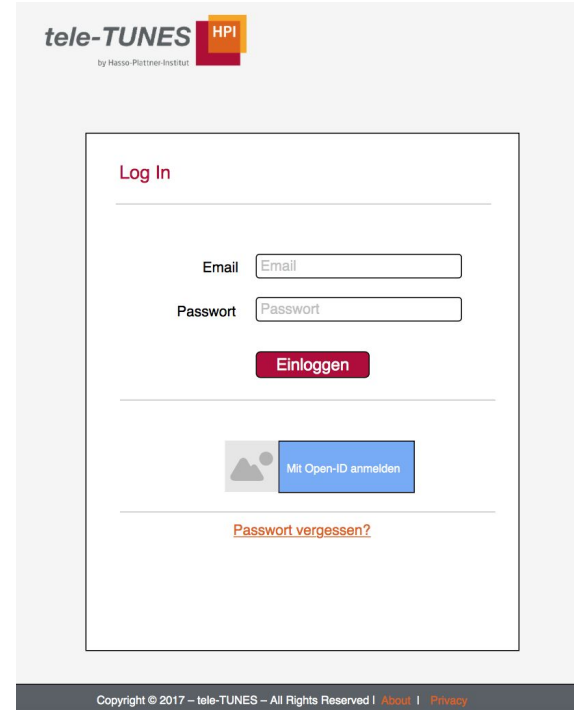
- Konzeptioneller Entwurf einer Applikation während der Planungsphase
- Nur die nötigsten Elemente einer Seite werden dargestellt
- User-Interaktion und Features abstimmen




Mock-Ups

Mock-Ups:

- Enthält die Funktionen sowie das Layout der Applikation
- Farbgebung und Bilder sind bekannt



The mock-up shows a login interface for 'tele-TUNES' by Hasso Plattner Institut. At the top left is the 'tele-TUNES' logo with 'by Hasso Plattner Institut' underneath, and to its right is the HPI logo. The main content area is enclosed in a light gray border and contains the following elements: a 'Log In' heading in red, followed by two input fields labeled 'Email' and 'Passwort' (both with placeholder text), a red 'Einloggen' button, a section for OpenID login featuring a gray icon of two people and a blue button labeled 'Mit Open-ID anmelden', and a red link for 'Passwort vergessen?'. The footer is a dark gray bar with the text 'Copyright © 2017 – tele-TUNES – All Rights Reserved' and links for 'About' and 'Privacy'.


tele-TUNES 
by Hasso Plattner Institut

Log In

Email

Passwort

Einloggen

 Mit Open-ID anmelden

[Passwort vergessen?](#)

Copyright © 2017 – tele-TUNES – All Rights Reserved | [About](#) | [Privacy](#)

Vor- und Nachteile

Vorteile:

- Zeitersparnis
- Absprache mit dem Kunden
- Absegnung des weiteren Vorhabens

Nachteile:

- Einschränkung des Denkens in vorgefertigte Bahnen

Warum Datenvisualisierung?

- Das HPI und Herr Bauer wollen weg von den unübersichtlichen Excel-Tabellen
 - Ziel:
 - Manueller Aufwand zur Erstellung von Reports soll vermieden werden
 - Informationen sollen effektiv (zielgerichtet) und effizient (wirtschaftlich) eingesetzt werden
 - Muster, Trends und Korrelationen sollen sich einfacher mit Datenvisualisierung entdecken lassen

Chart.js & D3.js

- Gemeinsamkeiten:
 - JavaScript-Bibliotheken
 - Dynamische Darstellung von Daten
 - Open Source
 - Werden kontinuierlich erweitert, optimiert
 - Kostenlos

Vorstellung: D3.js

- D3 := Data Driven Documents
- Technologien
 - HTML5, CSS3, SVG-Standards
- Keine 'klassische' Chart-Bibliothek, sondern eine Bibliothek zur Erstellung und Manipulation von Daten
- DOM-Struktur/Manipulation (Dokumenten-Objekt-Modell)
 - Erstellung von Balkendiagrammen oder komplexen Visualisierungen durch Änderung der DOM-Struktur



Vorstellung: Chart.js

- Datenübergabe und Konfiguration der Grafiken mittels JSON
- Technologien
 - HTML5-Element `<canvas>`



Vor- und Nachteile

	Chart.js	D3.js
Vorteile	<ul style="list-style-type: none">■ Einfache Syntax■ Kurz und elegant■ Geeignet für kurze bis mittelfristige Projekte■ Ideal im Zusammenspiel mit NodeJS■ Niedrige Renderzeit bei hoher Anzahl an Objekten/ Charts	<ul style="list-style-type: none">■ Vielfältige Datenvisualisierung und -interaktion■ Unbegrenzte Chart-Varianten■ Unterschiedliche Dateiformate■ SVG gerendert
Nachteile	<ul style="list-style-type: none">■ Komplexe, individuelle Charts schwer realisierbar■ Begrenzte Interaktion - dynamische Elemente dennoch möglich	<ul style="list-style-type: none">■ Komplexität■ Umfang■ Lange Einarbeitungszeit■ Hohe Renderzeit bei hoher Anzahl an Objekten/ Charts

SVG vs. Canvas (1)

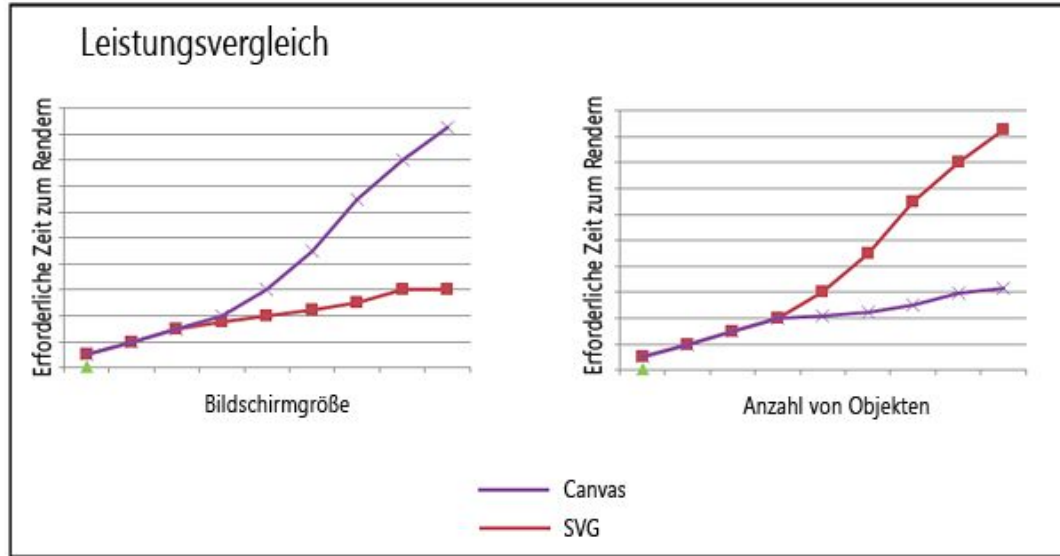
SVG

- Form-basiert mit Speichersystem
- Mehrere grafische Elemente, die Teil des DOM sind
- Erneutes Rendern dank Speichermodus
- Gute Skalierbarkeit

Canvas

- Pixelbasiert (dynamische PNG)
- Einzelnes HTML-Element
 - `<canvas>` - Tag
- Rendern im unmittelbaren Modus
- Schnellere Darstellung

SVG vs. Canvas (2)



Canvas

- Zunehmende Bildschirmgröße führt zur Verringerung der Leistung

SVG

- Zunehmende Anzahl von Objekten führt zur Verringerung der Leistung

→ In Abhängigkeit von der Implementierung und Plattform

Code-Beispiel

D3.js

```

1 var svg = d3.select("svg"),
2   margin = {top: 20, right: 20, bottom: 30, left: 50},
3   width = +svg.attr("width") - margin.left - margin.right,
4   height = +svg.attr("height") - margin.top - margin.bottom;
5
6 var tooltip = d3.select("body").append("div").attr("class", "tooltip");
7
8 var x = d3.scaleBand().rangeRound([0, width]).padding(0.1),
9   y = d3.scaleLinear().rangeRound([height, 0]);
10
11 var colours = d3.scaleOrdinal()
12   .range(["#6F257F", "#CA0059"]);
13
14 var g = svg.append("g")
15   .attr("transform", "translate(" + margin.left + "," + margin.top + ")");
16
17 d3.json("data.json", function(error, data) {
18   if (error) throw error;
19
20   x.domain(data.map(function(d) { return d.area; }));
21   y.domain([0, d3.max(data, function(d) { return d.value; })]);
22
23   g.append("g")
24     .attr("class", "axis axis--x")
25     .attr("transform", "translate(0," + height + ")")
26     .call(d3.axisBottom(x));
27
28   g.append("g")
29     .attr("class", "axis axis--y")
30     .call(d3.axisLeft(y).ticks(6).tickFormat(function(d) { return parseInt(d / 2000); }));
31   g.append("text")
32     .attr("transform", "rotate(-90)")
33     .attr("y", 6)
34     .attr("dy", "-0.71em")
35     .attr("text-anchor", "end")
36     .attr("fill", "#5D6971");
37
38   g.selectAll(".bar")
39     .data(data)
40     .enter().append("rect")
41     .attr("x", function(d) { return x(d.area); })
42     .attr("y", function(d) { return y(d.value); })
43     .attr("z", function(d) { return y(d.value); })
44     .attr("width", x.bandwidth());

```

Chart.js

```

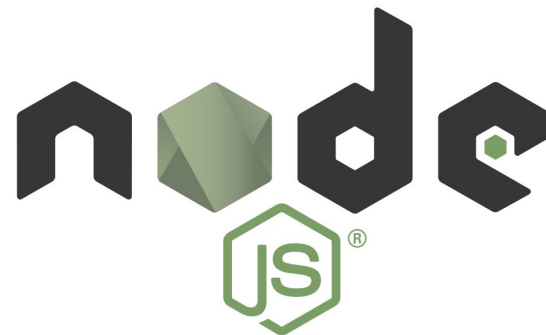
8 // Data with datasets options
9 var data = {
10   labels: ["Vanilla", "Chocolate", "Strawberry"],
11   datasets: [
12     {
13       label: "Ice Cream Prices ",
14       fill: true,
15       backgroundColor: [
16         'moccasin',
17         'saddlebrown',
18         'lightpink'],
19       data: [11, 9, 4]
20     }
21   ]
22 };
23
24 var options = {
25   tooltips: {
26     callbacks: {
27       label: function(tooltipItem) {
28         return "$" + Number(tooltipItem.yLabel);
29       }
30     }
31   },
32   title: {
33     display: true,
34     text: 'Ice Cream Truck',
35     position: 'bottom'
36   },
37   scales: {
38     yAxes: [{
39       ticks: {
40         beginAtZero: true
41       }
42     }]
43   }
44 };
45
46 // Chart declaration
47 var myBarChart = new Chart(ctx, {
48   type: 'bar',
49   data: data,
50   options: options
51 });

```

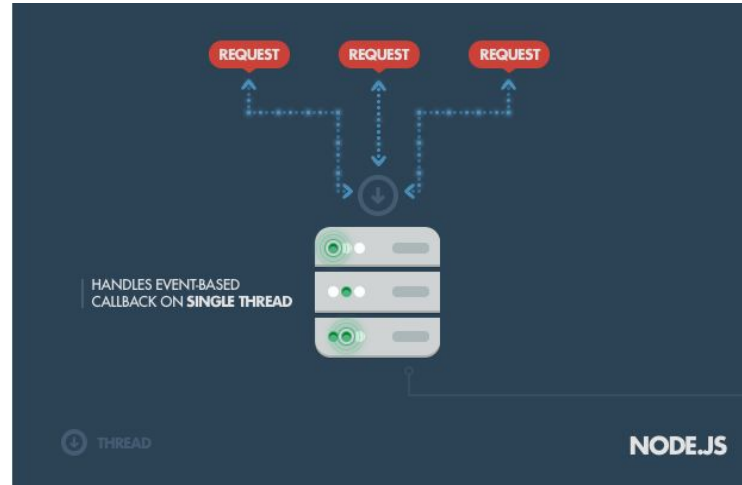
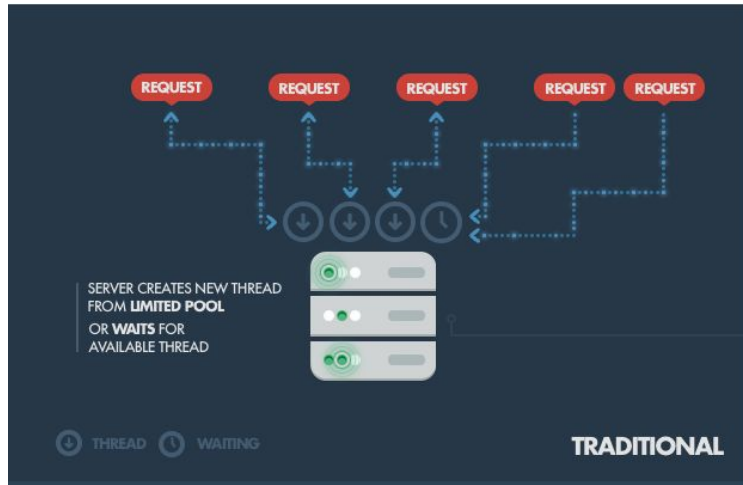
- Zur Erstellung eines simplen Balkendiagramms mit Tooltip

Vorstellung: Node JS

- Entwickelt von Ryan Dahl im Jahr 2009
- Serverseitige JavaScript-Laufzeitumgebung
- Chrome V8 basierend
- Größtes Packet Ecosystem: NPM
- Open-Source
- Genutzt von vielen Unternehmen



Single Thread / Async



Single Thread / Async

Single Threading

- Spart RAM und Overhead
- Gegensatz zu traditionellen Serverimplementierungen

Asynchrone Architektur

- IO anfragen blockieren Backend nicht
- Event-Driven -> Callback System



Vor- und Nachteile

Vorteile	Nachteile
<ul style="list-style-type: none">■ einfache und bekannte Programmiersprache■ einfacher Einstieg■ gute Performance■ asynchroner Code■ Behandlung gleichzeitiger Anfragen■ große Community■ Node package manager (npm)■ Kommunikation Server - Client	<ul style="list-style-type: none">■ Single Threaded■ Skalierbarkeit■ berechnungsintensive Aufgaben■ asynchrones Paradigma■ inkonsistente API



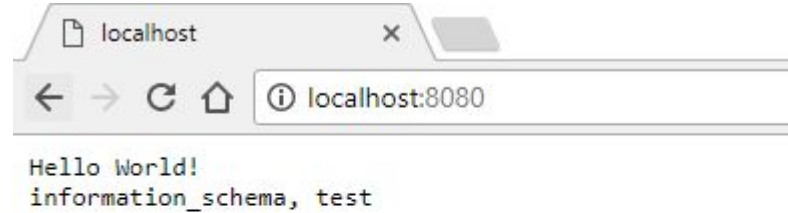
Einstieg

```
1 "use strict";
2
3 const express = require("express");
4
5 // Constants
6 const PORT = 8080;
7 const HOST = "0.0.0.0";
8
9 // App
10 const app = express();
11
12 app.get("/", (req, res) => {
13   res.send("Hello World!\n");
14 });
15
16 app.listen(PORT, HOST);
17 console.log(`Running on http://${HOST}:${PORT}`);
```



Einstieg - Callbacks

```
1 "use strict";
2
3 const express = require("express");
4 const mysql = require("mysql");
5
6 // Constants
7 const PORT = 8080;
8 const HOST = "0.0.0.0";
9
10 // App
11 const app = express();
12
13 app.get("/", (req, res) => {
14   getDatabases(result => {
15     res.write(result.join(", "));
16     res.end();
17   });
18   res.write("Hello World!\n");
19 });
20
21 app.listen(PORT, HOST);
22 console.log(`Running on http://${HOST}:${PORT}`);
23
24 function getDatabases(callback) {
25   var connection = mysql.createConnection({
26     host: "localhost",
27     user: "user",
28     password: ""
29   });
30
31   connection.connect(err => {
32     if (err) throw err;
33
34     connection.query("SHOW DATABASES", (err, result) => {
35       if (err) throw err;
36
37       var res = [];
38       result.forEach(val => {
39         res.push(val.Database);
40       });
41       callback(res);
42     });
43   });
44 }
```



“Callback Hell”

```
1 function initCB() {  
2   connection.connect(err => {  
3     if (err) throw err;  
4  
5     connection.query("CREATE DATABASE teletunes", (err, result) => {  
6       if (err) throw err;  
7  
8       console.log("Database created");  
9       connection.query("CREATE TABLE test (id INT PRIMARY KEY AUTO_INCREMENT)", (err, result) => {  
10        if (err) throw err;  
11  
12        console.log("Table created");  
13      });  
14    });  
15  });  
16 }
```

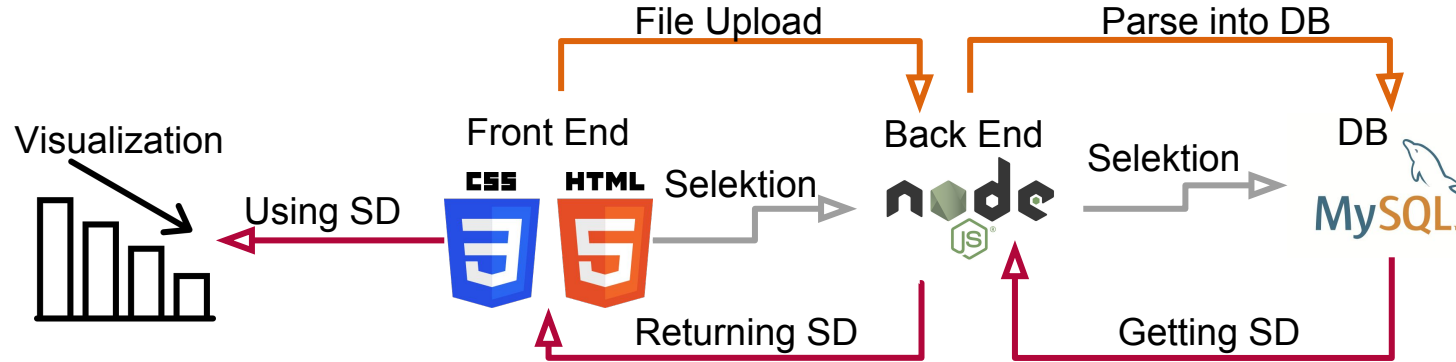
- viele “});” Endungen
- anonyme Funktionen

=> undurchsichtiger Code

“Callback Hell” - gelöst

```
1 function initPromise() {
2   connection.connect(err => {
3     if (err) throw err;
4
5     createDatabase("teletunes").then(() => createTable("test"))
6       .catch(err => console.log(err))
7   });
8 }
9
10 function createDatabase(dbName) {
11   return new Promise((resolve, reject) => {
12     connection.query("CREATE DATABASE " + dbName, (err, result) => {
13       if (err) reject(err);
14       console.log("Database created");
15       resolve();
16     });
17   });
18 }
19
20 function createTable(tableName) {
21   return new Promise((resolve, reject) => {
22     connection.query("CREATE TABLE " + tableName + " (id INT PRIMARY KEY AUTO_INCREMENT)", (err, result) => {
23       if (err) reject(err);
24       console.log("Table created");
25       resolve();
26     });
27   });
28 }
```

Tele-TUNES Architektur



SD: Selected Data
→ Uploading Pipeline
→ Visualizing Pipeline

Literaturverzeichnis

Bildquellen:

Docker Inc.: „Brand Guidelines“, URL:
<https://www.docker.com/brand-guidelines>
(abgerufen am 23.11.2017).

GitHub Inc: „GitHub Logos and Usage“, URL:
<https://github.com/logos>
(abgerufen am 23.11.2017).

Google Inc: „GitHub Logos and Usage“, URL:
Use the Drive Badge and Brand
(abgerufen am 23.11.2017).

Slack Technologie Inc.: „Brand Guidelines“, URL:
<https://slack.com/brand-guidelines>
(abgerufen am 23.11.2017).

Leistungsvergleich, Canvas/SVG:

URL:
[https://msdn.microsoft.com/de-de/library/gg193983\(v=vs.85\).aspx](https://msdn.microsoft.com/de-de/library/gg193983(v=vs.85).aspx) (abgerufen am 24.11.2017).

D3.js: “Logo”, URL:
https://upload.wikimedia.org/wikipedia/en/thumb/1/15/Logo_D3.svg/1079px-Logo_D3.svg.png
(abgerufen am 24.11.2017)

Chart.js: “Logo”, URL:
<http://www.chartjs.org/img/chartjs-logo.svg>
(abgerufen am 24.11.2017)

Literaturverzeichnis

Node.js: „Trademark Policy“, URL:
<https://nodejs.org/en/about/resources/>
(abgerufen am 26.11.2017).

CSS/HTML: „CC 3.0“, URL:
https://commons.wikimedia.org/wiki/File%3ACSS3_and_HTML5_logos_and_wordmarks.svg
(abgerufen am 26.11.2017).

MySQL: „Terms of Logo Use“, URL:
<https://www.mysql.com/about/legal/logos.html>
(abgerufen am 26.11.2017).

Graph: “Flaticon Basic License”
https://www.flaticon.com/free-icon/graph_138350
Author: <https://smashicons.com>
(abgerufen am 16.11.2017)

JS: „Logo.js“, URL:
<https://github.com/voodootikigod/logo.js/blob/master/registry.md>
(abgerufen am 26.11.2017).

Vergleich zw. Canvas und SVG: URL:
[https://msdn.microsoft.com/de-de/library/gg193983\(v=vs.85\).aspx](https://msdn.microsoft.com/de-de/library/gg193983(v=vs.85).aspx)
(abgerufen am 25.11.2017)