Web Engineering

Dr. Michael Lesniak
mlesniak@micromata.de

29. April 2019

MICROMATA

Wer steht da eigentlich vorne?

Michael

Worum geht‘s hier?

- Web Engineering
  - Entwicklung (Deployment, Wartung, …) von Software-Anwendungen
  - Visualisierung der Daten im Browser
  - Erreichbarkeit des technischen Backends
- Lernziele und andere Ziele
  - Engineering ist mehr als reine Technik
  - Methoden-Kompetenz
- Wie sieht es in der Praxis aus?

Praxis > Theorie

Buzzword Bingo!

REST

Security

JWT

CSS

Docker

Performance

HTML

Debugging

GitFlow(s)

React

Java

JavaScript

GraphQL

NPM

Maven

GitHub

Spring Boot

Tools

Git

Heroku

HTTP

Parcel

JSON

Architektur

Logging

OWASP

OAuth

Voraussetzungen

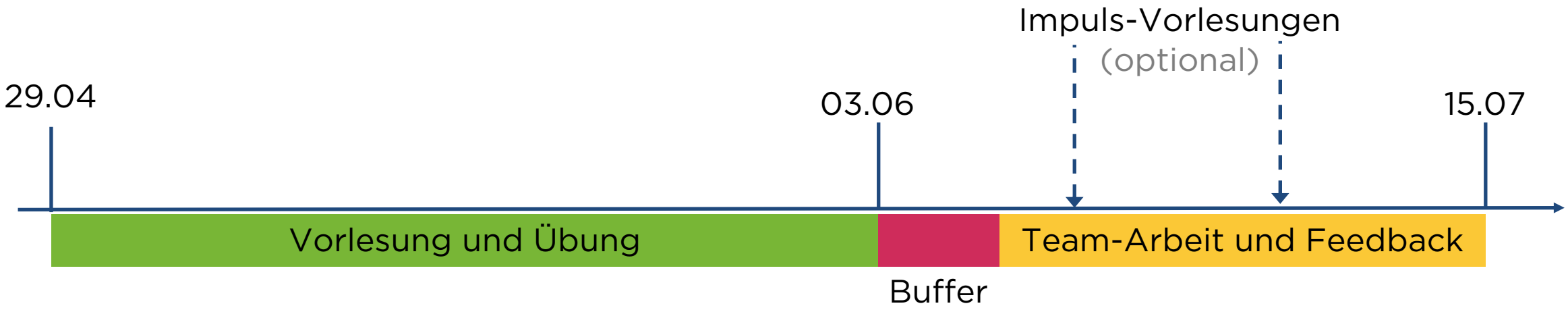
Selbstverantwortung

- **Keine** Vorlesungen, in der ich euch aus der Dokumentation für Frameworks und Programmiersprachen vorlese.
- IDEs, Java, HTML, CSS, JavaScript – Basics, Unix-Basics, Git, GitHub, …
- Hilfreich, aber **keine Pflicht**:
  - Teilnahme an Übungen
  - Nicht gut: Übungen nicht besuchen, und später nachfragen…
- Vorteil: Ihr habt die Chance, unglaublich viel zu Lernen und aktiv auszuprobieren

Meine Standardfrage bei Problemen: Was hast Du bereits ausprobiert?

Sommersemester 2019

- 11 Vorlesungen
- ca. 6 - 8 klassische Vorlesungen
- anschließend: Team-Arbeit mit wöchentlichen Reviews mit Michael



Ziele

- Technologien kennenlernen und anwenden
- Tools und Vorgehen aus der Praxis kennen und anwenden
- Tradeoffs erkennen und bewerten
- Feedback-Schleifen und Experimente
- Team-Arbeit praktizieren
- ein eigenes **Real-World-Projekt einer Web-Anwendung**
  - Anwendung online
  - Quellcode online
  - Optional: Open-Source

Schein, Schein, Schein …

- Umfangreiches Projekt im Team aus **N** Leuten
- Basis-Anforderungen kommen von mir als Kunden
- Extra-Features können (müssen aber nicht) vom Projekt-Team implementiert werden

- Team-Arbeit – Zusammenstellung ggf. durch mich
- Wöchentliche Treffen (Vor Ort und/oder Remote) mit den einzelnen Teams
  - Was lief gut?
  - Was ist geplant?
  - Wo gibt es Probleme?
  - Feedback und Tipps von mir
  - Gemeine Reflektionsfragen …

Slides & Code // Online

https://github.com/micromata/webengineering-2019

Kommunikation // Online

slack

- Ein Channel pro Vorlesung
- Ein Channel pro (zukünftigem) Projekt-Team für alle Fragen um das Projekt
- (Mail geht natürlich auch …)

- Anonymes Feedback (auch in der Vorlesung) mit `/abot #lecture-2019-04-29 Hello, world!`
- Ab und zu anonyme Abstimmungen von mir …
- Gerne auch Fragen zur Praxis

https://bit.ly/2PD6Pao

Übung: #Slack

- Account registrieren
- @michael eine Nachricht mit Deiner Emailadresse schreiben
- Ein GIF deiner Wahl in #random mit /giphy posten

https://bit.ly/2PD6Pao

Kommunikation // Offline

Eure Aufgabe in der Vorlesung

- Mitdenken
- **Fragen** (Risiko: Impulsvorträge (von mir) ;-))
- Diskutieren
- Reinrufen
- Widersprechen
- ...

... nicht rumdösen ...

Michael weiß auch nicht alles (besser).
Technische Entscheidungen sind oft einfach nur subjektive Meinungen des Dozenten.

Feedback gewünscht!

- Möglichkeit, die Vorlesung aktiv mitzugestalten
  - Zukünftige Themen
  - Stil, Art, …
  - …
- Eure Chance – nutzt sie aktiv.

Love it, leave it, … or give at least feedback and try to change it ;-)

Vorlesung vs. Übung

- Nahtloser Übergang (mit kleiner Pause)
- Kein Freund von Pflichtveranstaltungen – wer gehen will, geht (…leise)
- Herumspielen, Experimentieren, Lernen, Gegenseitig über die Schulter gucken, helfen, …
- Ich behalte mir vor, mit einem dummen Spruch und wesentlich langsamer zu helfen, wenn jemand später Probleme hat, die in der Übung geklärt werden könnten ;-)

Unsere Anwendung ... heute

- Software Engineering in der echten Welt:
  - Wasserfall?
  - MVP!
    - iterativ
    - Fehler sind erlaubt
    - 2 Schritte vor, ein Schritt zurück

Gedanke: Geht es noch einfacher?

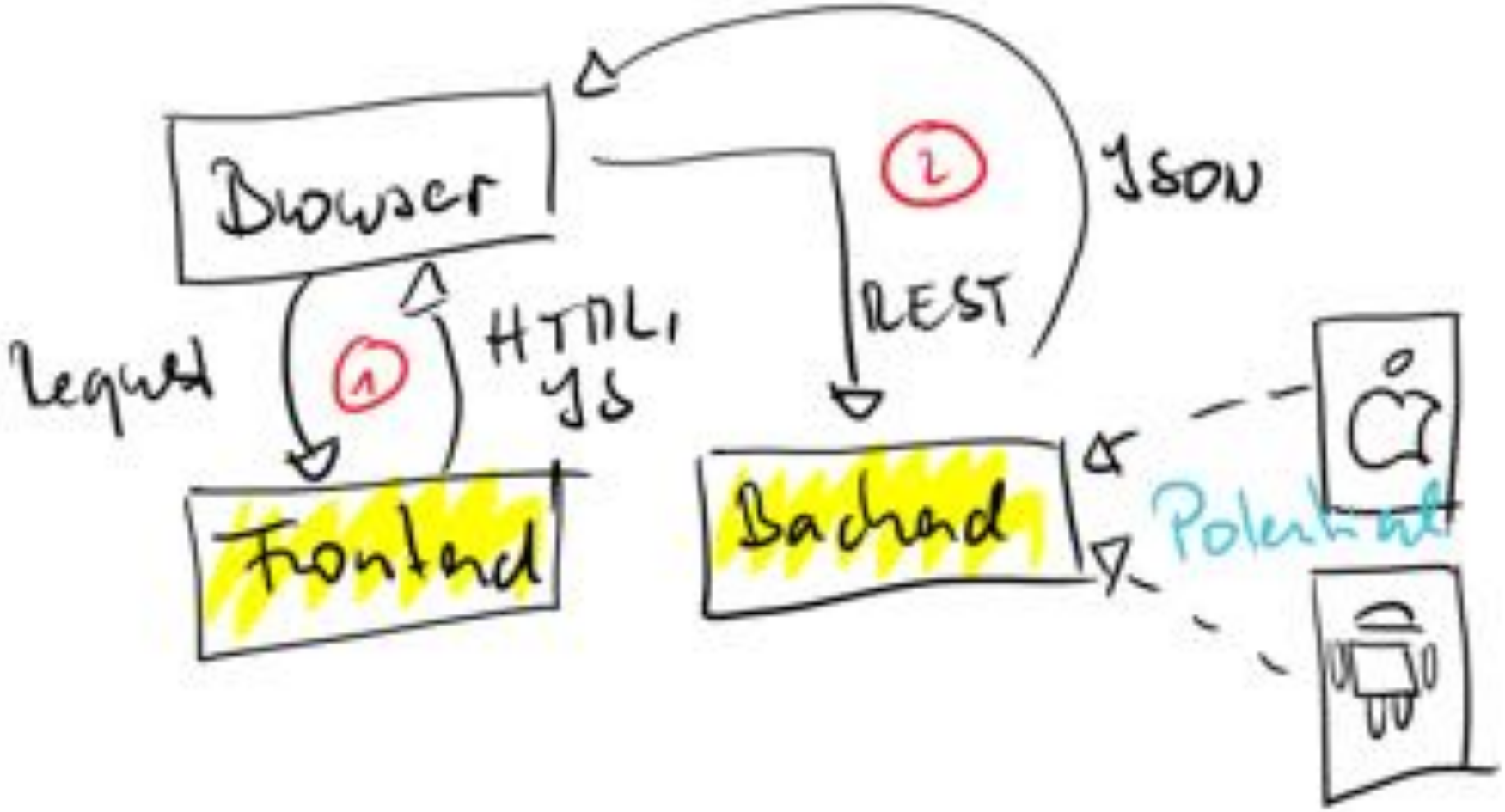
- Zielanwendung in der Vorlesung: https://news.ycombinator.com
- Was ist der Unterschied zwischen einer stylischen und nicht stylischen Web-Anwendung?
  - CSS und fancy stuff -> Design nur am Rande der der Vorlesung
  - Hier: Full-Stack Engineering

Unser Ziel ... heute

- Technischer Durchstich
- Happy Flow
- Basisarchitektur
- Input/Output (Übung)

- Nicht-Ziele:
  - Deployments
  - Tests
  - Schönheitspreis
  - Coole JavaScript Frameworks

Basisarchitektur



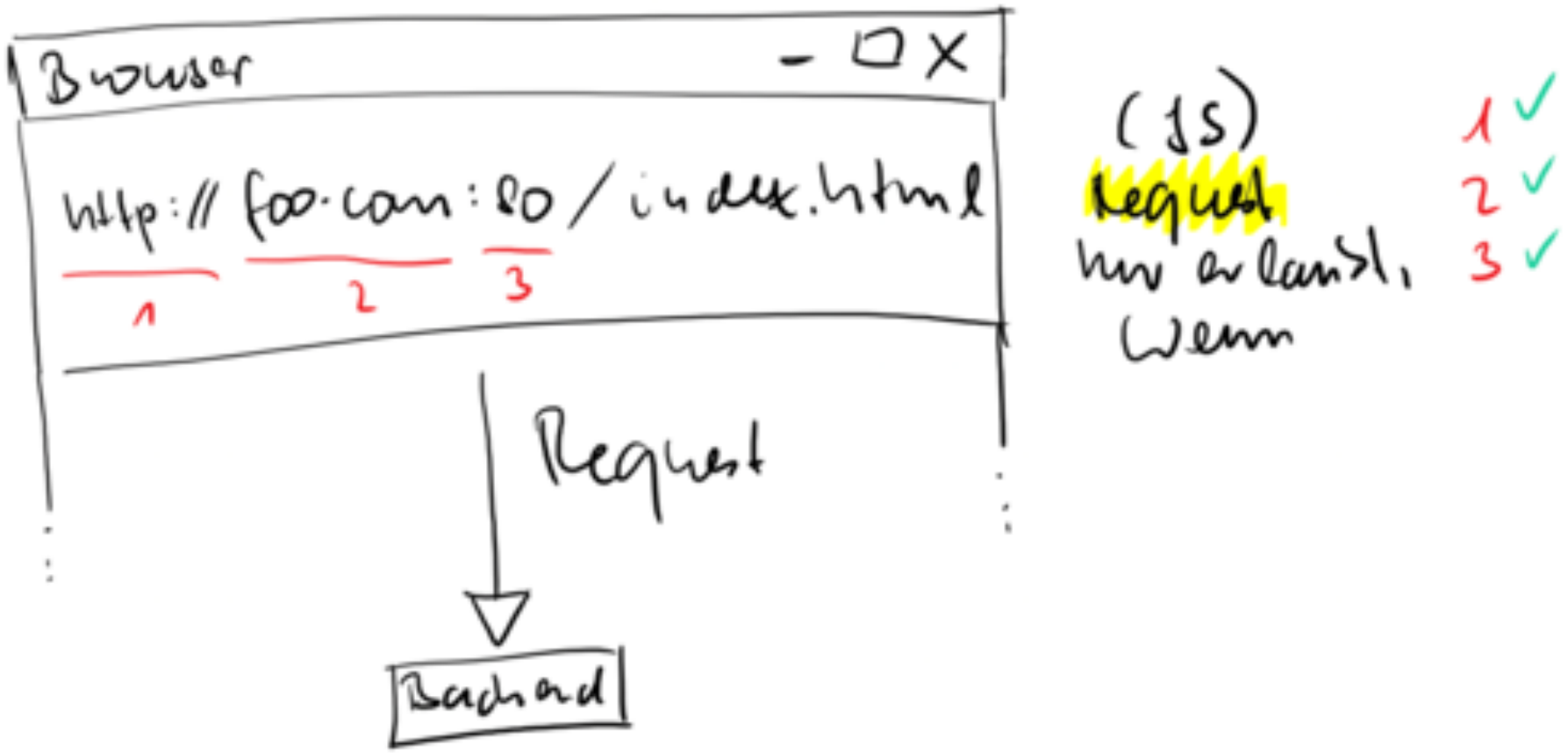
Talk is cheap. Show me the code.

- Linus Torvalds (2000-08-25)

SOP und CORS :-|

Simplified!

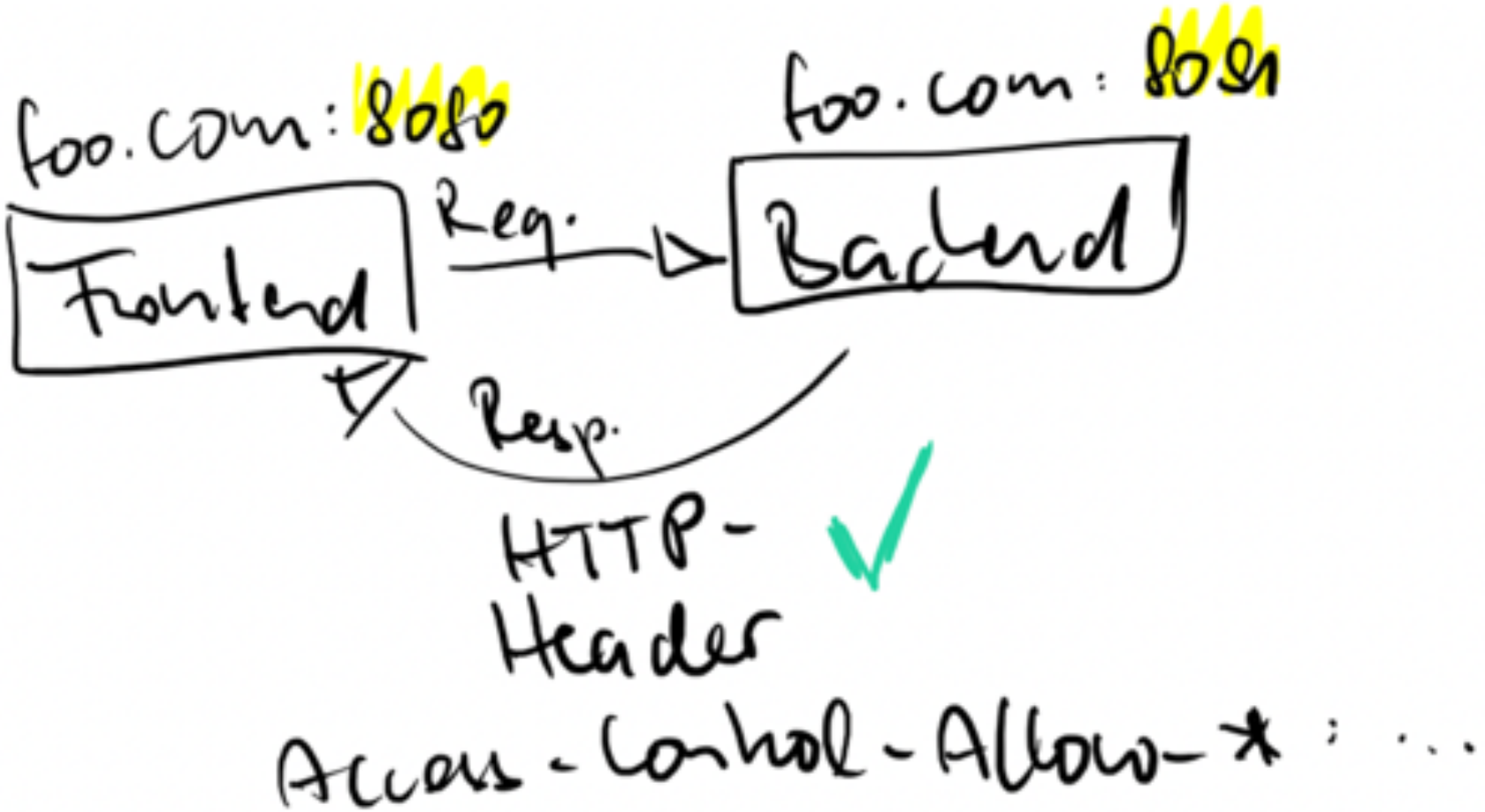
- Same-Origin-Policy



SOP und CORS :-)

Simplified!

- Cross-Origin Resource Sharing



Übung: Code!

Vorlesung selber Nachbauen

- … – nicht einfach herunterkopieren

Übung: Code!

Input/Output

- Backend soll eine Konvertierung eines Strings in GROSSBUCHSTABEN ermöglichen.
- HTTP POST/REST
- `public ... toUppercase(@RequestBody Map<String, String> ...)`
- Frontend soll Eingabe ermöglichen und Ergebnis zeigen (Fehler ignorieren ist ok)

- (Anonymes) Feedback zur Schwierigkeit
  - in `#lecture-2019-04-29` und
  - per Vote im Channel

Übung: Code!

Explorieren und Experimentieren

- Was passiert, wenn ich Dinge weglasse / hinzufüge / ändere?
- Möchte von jedem 3 Dinge, die er weggelassen oder geändert hat und eine entsprechende kleine Erläuterung per #slack-Nachricht
  - Nicht anonymisiert ;-)