Web Entwicklung 2

Michael Johann

Fachbereich Wirtschaft - Fachhochschule Münster Bachelor of Science Wirtschaftsinformatik Wintersemester 2014/2015

Agenda

- Organisatorisches
  - Veranstaltungsdaten
  - Vorstellung des Dozenten
  - Inhaltsübersicht
- Kapitel 0: Einleitung

Organisatorisches

Verlauf der Veranstaltung

- Vorlesungen und Übungen
  - Zeit Vorlesung: Montags 12:30 Uhr bis 14:00 Uhr
  - Zeiten Übungen: Gruppe I: Montags 14:15 Uhr bis 15:45 Uhr
    Gruppe II: Montags 16:00 Uhr bis 17:30 Uhr

Leistungsnachweis / Klausur

- Leistungsnachweis: Fallstudie
- Ziel: Entwicklung einer Ruby-on-Rails-Anwendung
- Gruppenarbeit: bis 4 Personen
- Themen:
  - Auswahl eines vorgegebenen Themas oder eigener Vorschlag eines eigenen Themas
  - Vergabe erfolgt ca. Mitte des Semesters
- Abgabe:
  - Quellcodes
  - „Angemessene“ Dokumentation
- Abgabetermin:
  - Nach Ende der Klausurenphase (genauer Termin wird noch bekannt gegeben)

Kontext der Veranstaltung

Grundstufe		Aufbaustufe			Erweiterungsstufe		
1. Semester (Winter)	2. Semester (Sommer)	3. Semester (Winter)	4. Semester (Sommer)		5. Semester (Winter)	6. Semester (Sommer)	
Grundlagen Wirtschaftsinformatik	Datenbanken	Betriebliche Anwendungssysteme I	Betriebliche Anwendungssysteme II	Praxisphase	Projektmanagement	Praxisphase	IT-Management
Grundlagen Programmierung	Software-Entwicklung I	Software-Entwicklung II	Software Engineering		Business Engineering		Projekt
Betriebssysteme und Rechnerarchitekturen	Web-Entwicklung I	Web-Entwicklung II	Web Engineering		E-Services		Thesis
Grundlagen Betriebswirtschaftslehre	Netzwerke	Betriebswirtschaftliche Primärprozesse	Wirtschaftsrecht		Business Intelligence		Kolloquium
Volkswirtschafts-lehre	Finanzwirtschaftl. Betriebswirtschaftslehre	Statistik	Quantitative Methoden		Schlüssel-kompetenzen III		
Wirtschafts-mathematik	Mathematik für Informatik	Englisch	Schlüssel-kompetenzen II		Wahlpflicht I		
Schlüssel-kompetenzen I			Transfermodul		Wahlpflicht II		

Lernziele

- Lernen, wie Programmierung im "Großen" geht
- Entwicklung eines grundlegenden Verständnis in Web- und Software Architekturen
- Lernen, wie man Projekte angeht und wie man Ideen online bekommt
  - Von der Idee ...
  - ... über Entwicklung ...
  - ... über Deployment ...
  - ... bis zur Maintenance

Programmierung im "Großen"

- Generell: Komplexe datenbankgestützte Webanwendungen mit Hilfe von serverseitigen Frameworks implementieren können
- Speziell:
  - Die Programmiersprache Ruby beherrschen
  - Das serverseitige Framework Ruby-on-Rails einsetzen können
  - Die Kenntnisse der clientseitgen Technologien HTML, CSS und JavaScript vertiefen
  - Aspekte der Usability bei der Entwicklung von Webanwendungen berücksichtigen können
  - Deployment von Webanwendungen
  - Aspekte der Sicherheit und Stabilität bei der Entwicklung von Webanwendungen berücksitigen können

Didaktisches Konzept

- Vorlesung
  - Vorstellung von Programmierkonzepten und Herangehensweisen
  - Zunächst: Vermittlung von „theoretischem“ Wissen
  - Später: Entwicklung eines komplexen Projektes (Step-by-Step)
- Übung
  - Vertiefung von Inhalten der Vorlesung
  - Praktische Anwendung des Wissens
- Begleitmaterial
  - Orientierung an Railsguides
  - Bei Bedarf Bereitstellung weiterer Materialien

Literaturempfehlungen

- **Agile Web Development with Rails 4** by Sam Ruby, Dave Thomas, David Heinemeier Hansson http://pragprog.com/book/rails4/agile-web-development-with-rails
- **Programming Ruby 1.9 & 2.0 (4th edition): The Pragmatic Programmers' Guide** by Dave Thomas with Chad Fowler and Andy Hunt http://pragprog.com/book/ruby4/programming-ruby-1-9-2-0

Weblinks

- Ruby on Rails Website: http://rubyonrails.org
- Rails Guides: http://guides.rubyonrails.org/
- Gutes Videotutorial: http://railsforzombies.org
- Screencasts zu verschiedenen Rails-Themen: http://railscasts.com

Weitere Weblinks

- http://tryruby.org/levels/1/challenges/0
- http://mislav.uniqpath.com/poignant-guide/book/chapter-1.html
- http://api.rubyonrails.org/
- http://ruby-doc.org/core-2.0.0/

Vorstellung des Dozenten

Michael Johann (45)

- Geboren in Hamm
- Studium Maschinenbau, TFH Bochum, Abschuss Dipl.-Ing. (FH)
- Während des Studiums:
  - Freelancing als Software Entwickler
- https://www.xing.com/profile/Michael_Johann

Motivation (aus Dozentensicht)

- Web Entwicklung lernt man nicht aus Büchern, sondern in der Praxis
- Erfahrung und Wissen weitergeben
- Mehr Leute für moderne Technologien begeistern ("Evangelism")

Inhaltsübersicht

Inhalte der Vorlesung

- Theorie-Teil
  - Einleitung
  - Einführung in Ruby on Rails
- Praxis-Teil
  - Entwicklung eines Beispielprojektes
  - Engineering Aspekte

Kapitel 0: Einleitung

Ein kurzer Blick zurück ...

Webentwicklung I

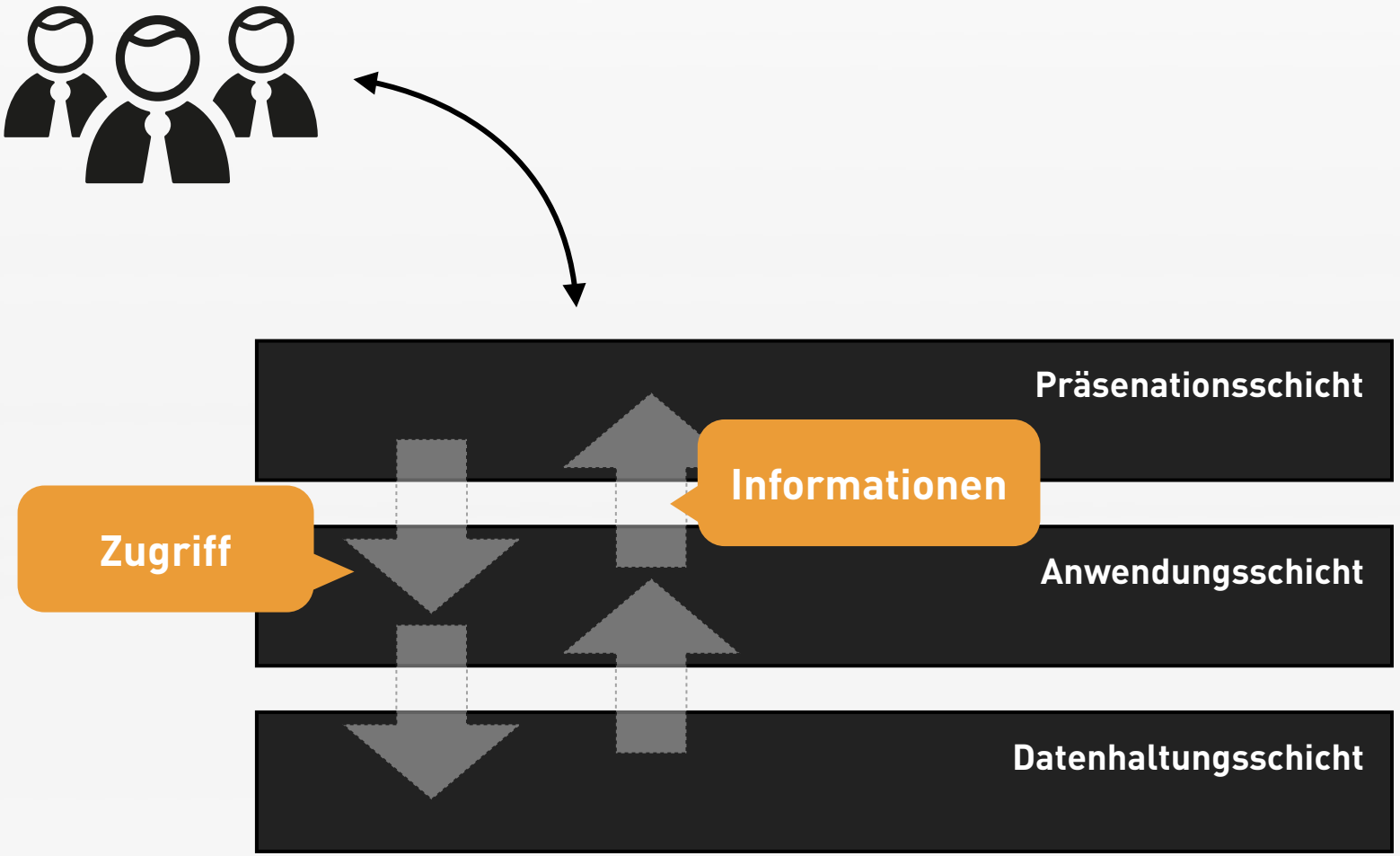
Statische vs. dynamische Webseiten

RECAP

- Ursprunglicher Zweck des WWW:
  - Darstellung von statischen Inhalten
  - Typisch: Server liefert den Inhalt einer auf der Festplatte abgelegten HTML-Datei aus
  - Üblich: Dokumente manuell erstellt
- Technische Moglichkeit:
  - Dynamische Erzeugung des auszuliefernden HTML-Dokumentes
  - Server-Software generiert Dokument „on-the-fly", z.B. aus in einer Datenbank enthaltenen Informationen.
- Konsequente Weiterentwicklung der Idee:
  - Entkopplung der Prasentationsschicht einer Software-Anwendung uber das WWW wird möglich

Schichten Modell

RECAP



Schichten Modell

RECAP

- Präsentationsschicht (engl. Presentation Layer):
  - Präsentation der Daten für den Benutzer
  - Entgegennahme von Benutzereingaben
  - Weitere Synonyme: Front-End, Client-Schicht, Benutzer-Schnittstelle
- Anwendungsschicht (engl. Application Layer):
  - Beinhaltet die eigentliche Geschäftslogik
  - Weitere Synonyme: Logik-Schicht, Middle-Tier, Enterprise-Tier, Businessschicht
- Datenhaltungsschicht (engl. Data Layer):
  - Verantwortlich für die persistente Speicherung der Daten
  - Weitere Synonyme: Back-End, Database-Tier

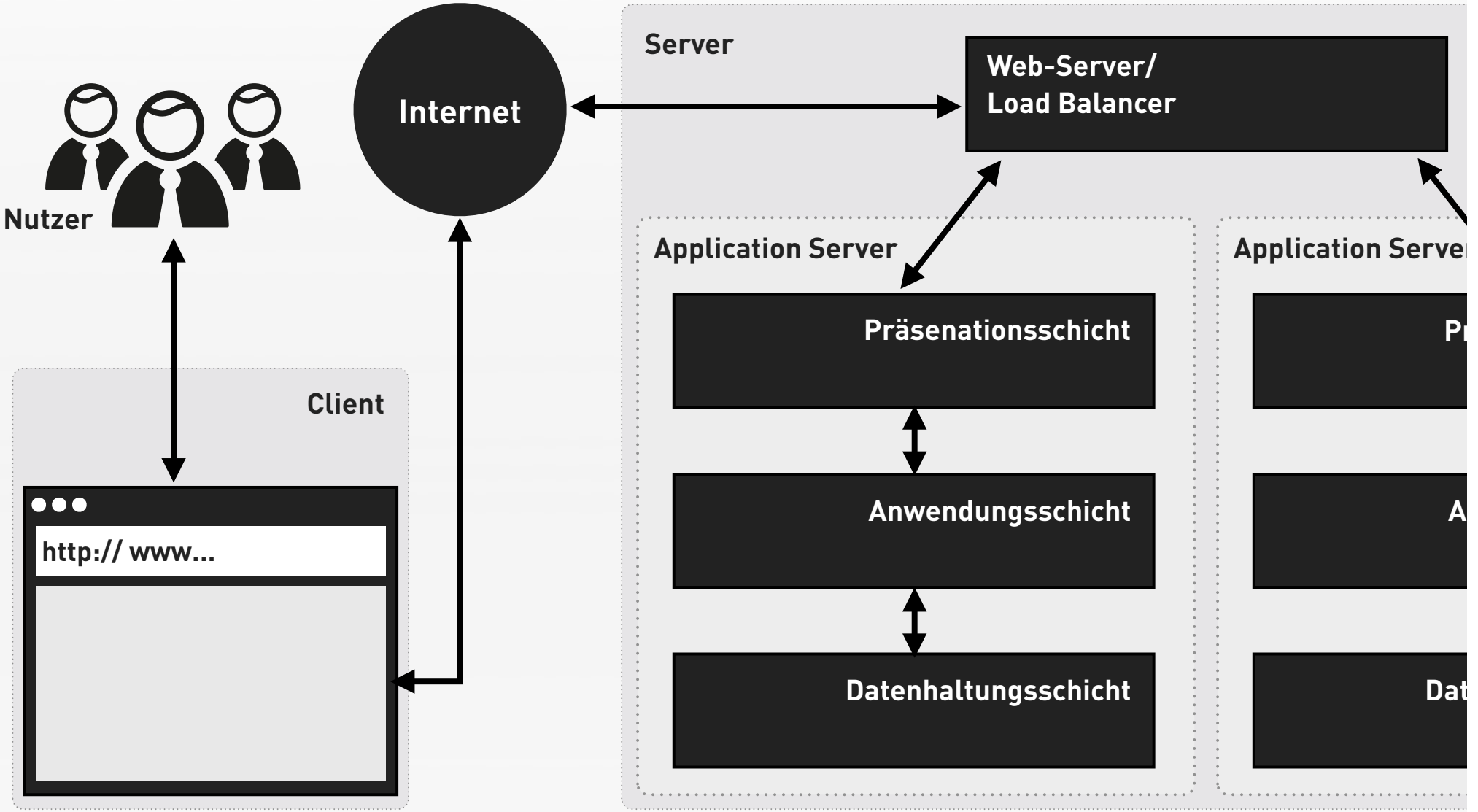
Client-Server

RECAP

- Standard-Modell zur Verteilung von Aufgaben im Netzwerk
- Server:
  - Software-Anwendung, die **Dienst** (engl. Service) anbietet
  - Bereitstellung einer Netzwerk-Schnittstelle, über die der Dienst in Anspruch genommen werden kann
  - Festlegung eines **Protokolls**, welches den konkreten Ablauf und das Format des Datenaustausches deniert
  - **Passive Komponente**: Server wartet auf eingehende Anfrage (engl. **Request**), bearbeitet diese und sendet eine Antwort (engl. **Response**) zurück
- Client:
  - Software-Anwendung, die einen bestimmten Dienst nutzt
  - **Aktive Komponente**: Client stellt über das Netzwerk Anfrage an Server und wartet auf Antwort
  - Varianten: synchron vs. asynchron, Meldung vs. Auftrag

Architektur einer Web-Anwendung

RECAP



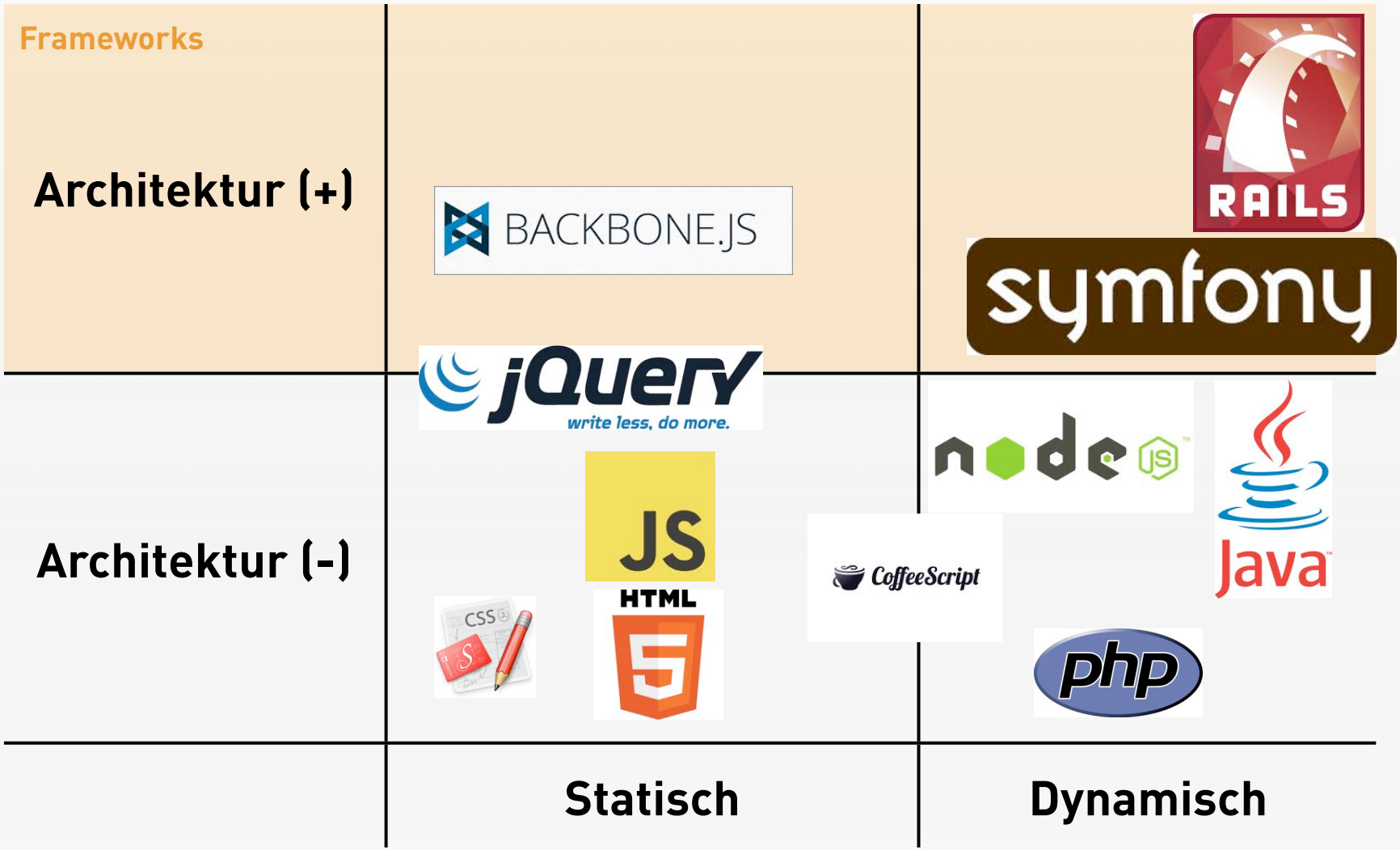
HTTP Request Methods

RECAP

- Werden von Representational State Transfer (REST) genutzt, um die Art der Anfrage zu beschreiben
- **GET**: fordert die angegebene Ressource vom Server an
- **POST**: legt eine neue Ressource auf dem Server an
- **PUT**: aktualisiert eine bestehende Ressource auf dem Server
- **DELETE**: löscht eine Ressource vom Server
- Weitere HTTP Request Methoden
  - PATCH, HEAD, OPTIONS

Web Technologien (Auswahl!)

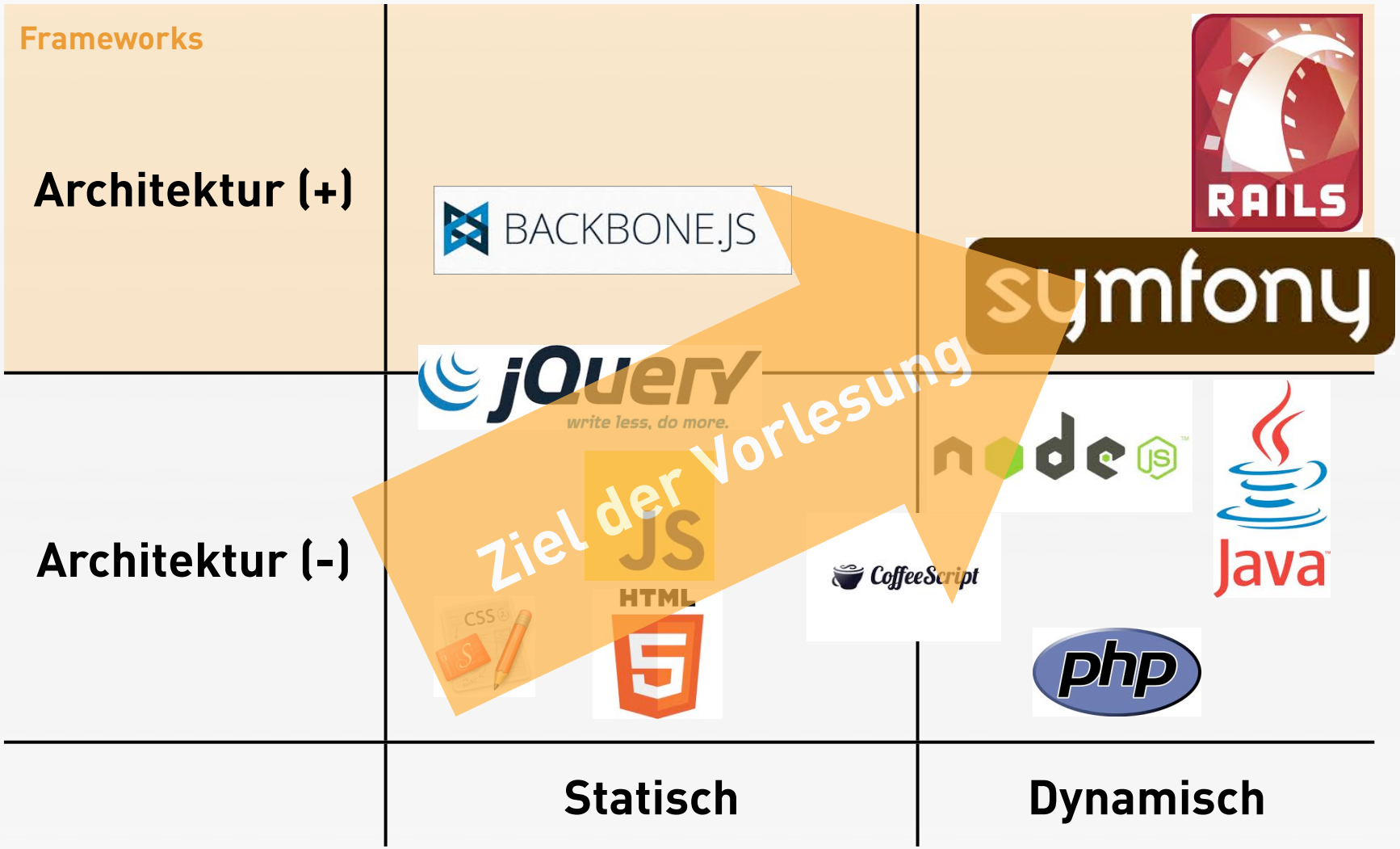
Web Technologien (Auswahl!)



Ziele von Frameworks

- Unterstützung bei Architekturentwurf
- Wiederholende Tätigkeiten vereinfachen
- Wiederverwendung von Code
- Standardisierung von Tätigkeiten („Dokumentation“)
- Steigerung der Entwicklungsgeschwindigkeit
- Nutzung von Best-Practises

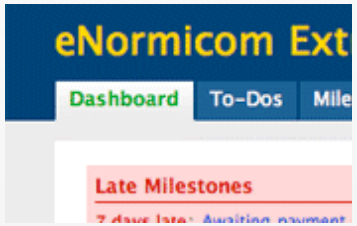
„Lernkurve“

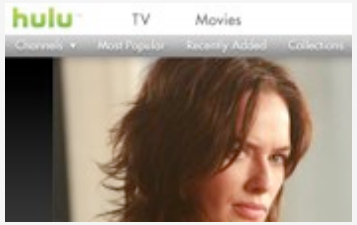

Frameworks
Architektur (+)
Architektur (-)
Statisch
Dynamisch
BACKBONE.JS
RAILS
symfony
jQuery
write less, do more.
node
Java
JS
HTML
CoffeeScript
CSS
php
Ziel der Vorlesung

Warum Ruby on Rails?

- Ruby: Moderne Programmiersprache
- Rails: Framework zur Webentwicklung
- Fokus ausschließlich auf Webanwendungen
- Vorteile Ruby on Rails
  - Große, aktive Community / Ecosystem
  - Viele Ressourcen im Netz
  - Best Practises
  - Steile Lernkurve
- Sehr viele, frei verfügbare Erweiterungen
  - z. B. für Anmeldung, Kommentare, etc.

Ruby on Rails Beispiele





Und viele mehr!

Fazit

- Los geht‘s! :-)