

Multimedia Engineering II

02 Client-Server Architektur

Johannes Konert



BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN

University of Applied Sciences



Agenda

- Wiederholung
- Ergebnisse der Umfragen

Client-Server-Architektur

- C-S-Aufbau
- Client: Kommunikation
- LAMP und MEAN
- Server: Architekturkomponenten
- Auswahl-Kriterien für Komponenten
- Das Nadelöhr

- Node.js: Mein erster Server mit JavaScript

- Zusammenfassende Fragen
- Trends
- Ausblick

Wiederholung von der ersten Woche

Fragen

1. Wie wählen Sie per CSS Selektor nur die DIV Elemente aus, welche direkte Kind-Elemente von `<body>` sind?
2. Welche zwei Parameter benötigen Sie, um per JavaScript einen Eventlistener mit `.addEventListener(...)` im DOM zu registrieren? Welchen Typ haben diese?
3. Welche semantischen Tags aus HTML5 kennen Sie?

Aufgabe: Round-Robin-Wiederholung

- Nehmen Sie einen Zettel und schreiben im Querformat 1,2,3 oben drauf
- Tragen Sie zu den Fragen 1,2,3, die Antworten, welche Sie haben oben ein. Lassen Sie Unbeantwortetes leer **(2min)**
- Geben Sie den Zettel in der Reihe weiter, wenn Sie dazu aufgefordert werden
- Prüfen Sie die Antworten auf Ihrem neuen Zettel und ergänzen Sie diese ggf. **(1min)**. Tauschen Sie sich aus.
- Offene Fragen klären wir gemeinsam.

Wiederholung - Lösung

Wie wählen Sie per CSS Selektor nur die DIV Elemente aus, welche direkte Kind-Elemente von <body> sind?

```
body > div { color: red; }
```

Wiederholung - Lösung

Welche zwei Parameter benötigen Sie, um per JavaScript einen Eventlistener mit `.addEventListener(...)` im DOM zu registrieren? Typ?

Syntax

```
target.addEventListener(type, listener[, useCapture]);
```

type

A `string` representing the `event type` to listen for.

listener

The `object` that receives a notification when an event of the specified type occurs. This must be an object implementing the `EventListener` interface, or simply a JavaScript `function`.

useCapture

Optional

Wiederholung - Lösung

Welche zwei Parameter benötigen Sie, um per JavaScript einen Eventlistener mit `.addEventListener(...)` im DOM zu registrieren? Typ?

Syntax

```
target.addEventListener(type, listener[, useCapture]);
```

```
// add event listener to table  
var el = document.getElementById("outside");  
el.addEventListener("click", modifyText, false);
```

```
<table id="outside">  
  <tr><td id="t1">one</td></tr>  
  <tr><td id="t2">two</td></tr>  
</table>
```

Quelle: MozillaDeveloperNetwork
<https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener>



Wiederholung - Lösung

Welche semantischen Tags aus HTML5 kennen Sie?

1. <thead>
2. <a>
3. <aside>
4. <main>
5. <nav>
6. <section>
7. <header>
8. <footer>

1. Tabellenkopfzeilen
2. Verlinkung (Hypertext)
3. Zusatzinformation mit Bezug zur Umgebung
4. Hauptteil des Dokumentes (Einzigartig pro URL)
5. Navigationselemente der Seite/App
6. Abschnitt der Seite (Geschachtelt, ggf. mit <article>)
7. Kopfteil eines <body>, <section> oder <article>
8. Fussteil eines <body>, <section> oder <article>

..und weitere...

Semantik: Lehre von der Beziehung und Bedeutung von Objekten
...in HTML typischerweise die Auszeichnungen,
die keine Layout/UI-Aspekte ausdrücken, sondern ausschließlich Bedeutung und Bezug

Agenda

- Wiederholung
- Ergebnisse der Umfragen

Client-Server-Architektur

- C-S-Aufbau
- Client: Kommunikation
- LAMP und MEAN
- Server: Architekturkomponenten
- Auswahl-Kriterien für Komponenten
- Das Nadelöhr
- Node.js: Mein erster Server mit JavaScript
- Zusammenfassende Fragen
- Trends
- Ausblick

Ergebnisse zur Frage „Was möchten Sie lernen in ME2“?

- **95 Antworten**
- **Kategorien:**
 - **JavaScript allgemein**
 - **Server: Backend, Technologie, REST**
 - **Client: Dynamische Webseiten, SPA, Frameworks, AngularJS**
 - **Sonstiges**

- **Server: Backend, Technologie, REST**



Ergebnisse zur Frage „Was möchten Sie lernen in ME2“?

- Client: Dynamische Webseiten, SPA, Frameworks, AngularJS



Ergebnisse zur Frage „Was möchten Sie lernen in ME2“?

- Sonstiges (davon 9x JS, 7x der Rest)

AWS
GuterCode

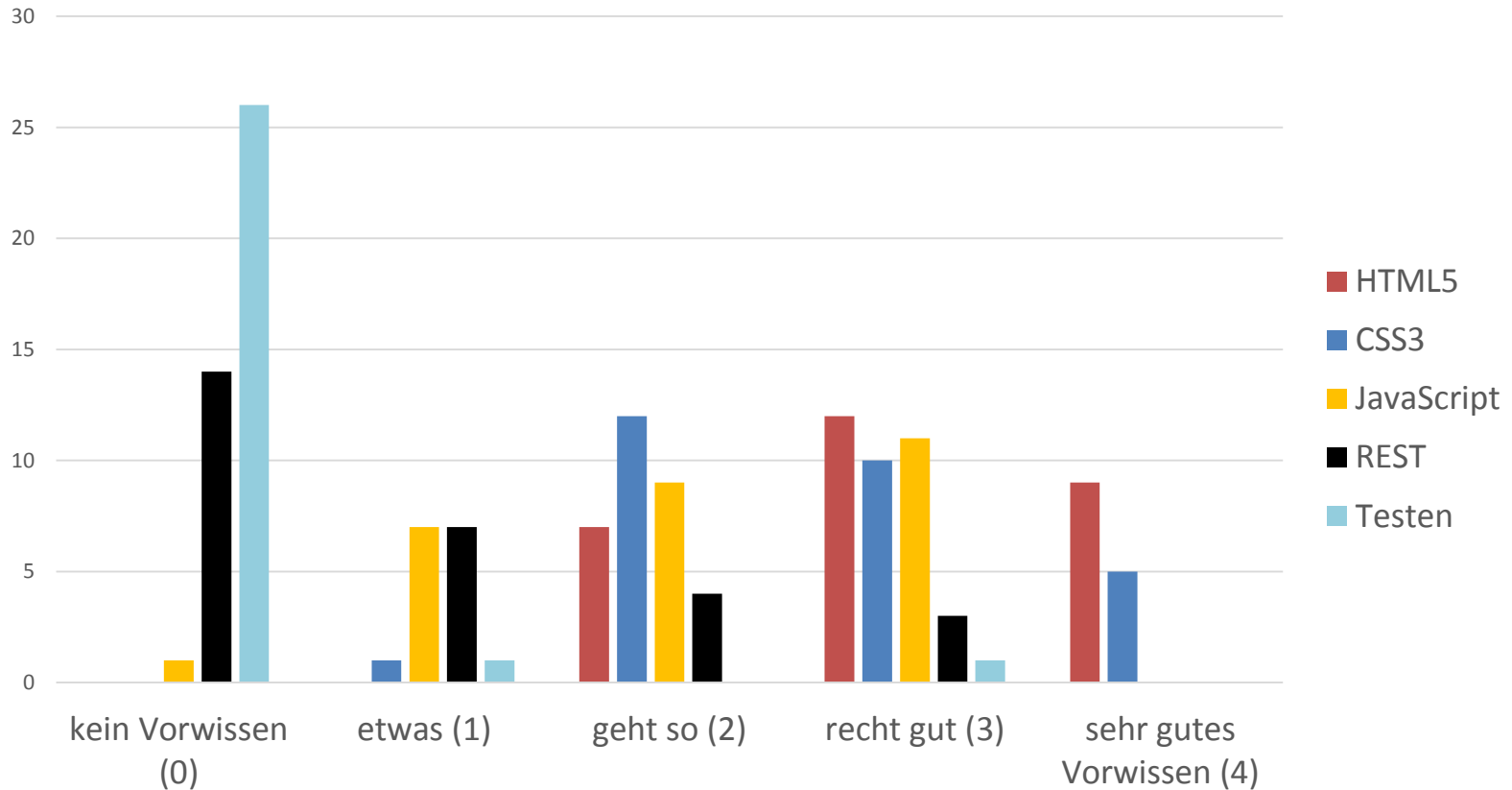
Sytanx Tools

Semantik
Login
Vertiefung

JavaScript

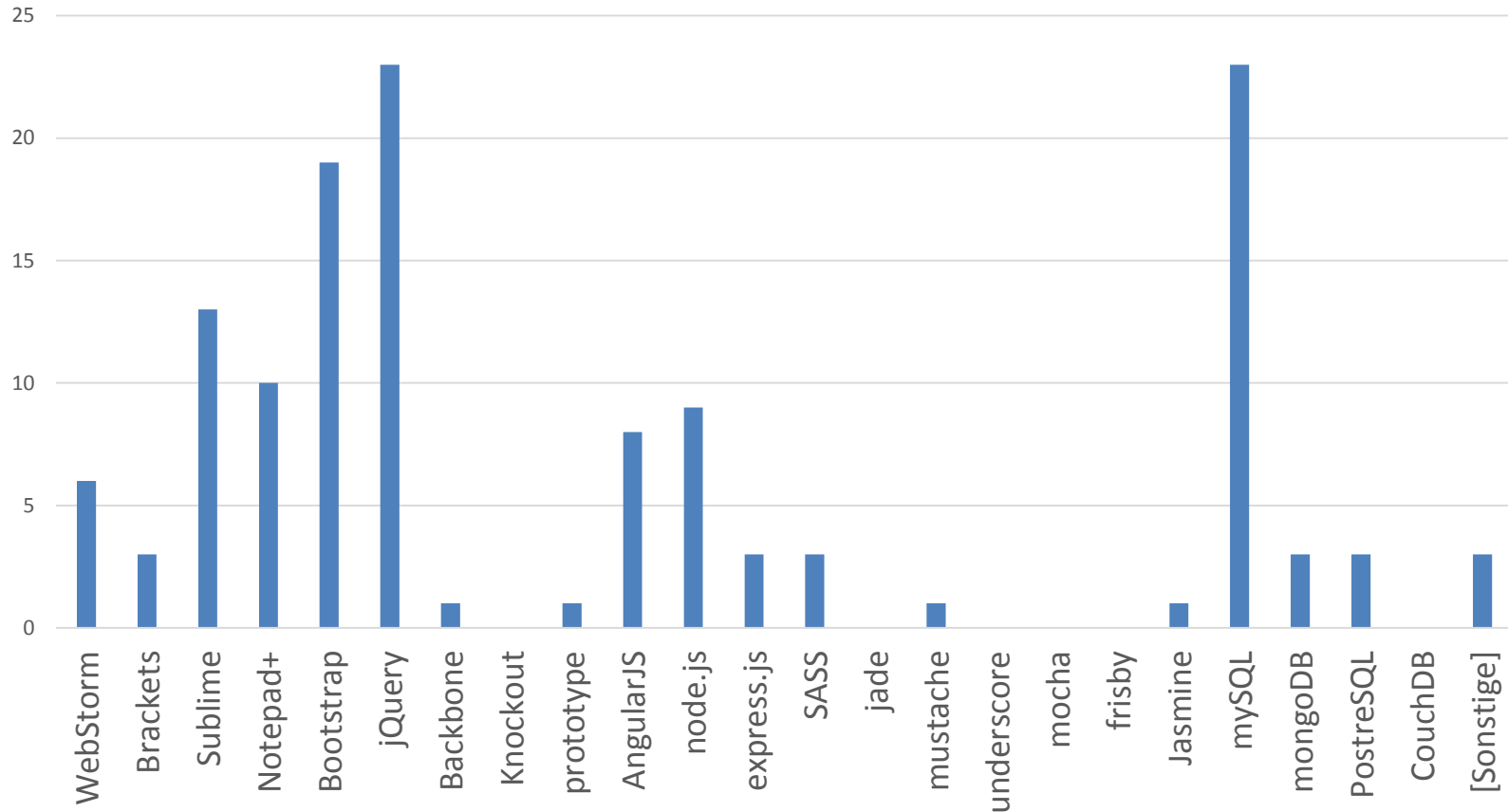
Ergebnisse der Moodle Umfrage

Wie gut schätzen Sie ihr Vorwissen ein?



Ergebnisse der Moodle Umfrage

Welche der Werkzeuge/Tools haben Sie bereits eingesetzt?

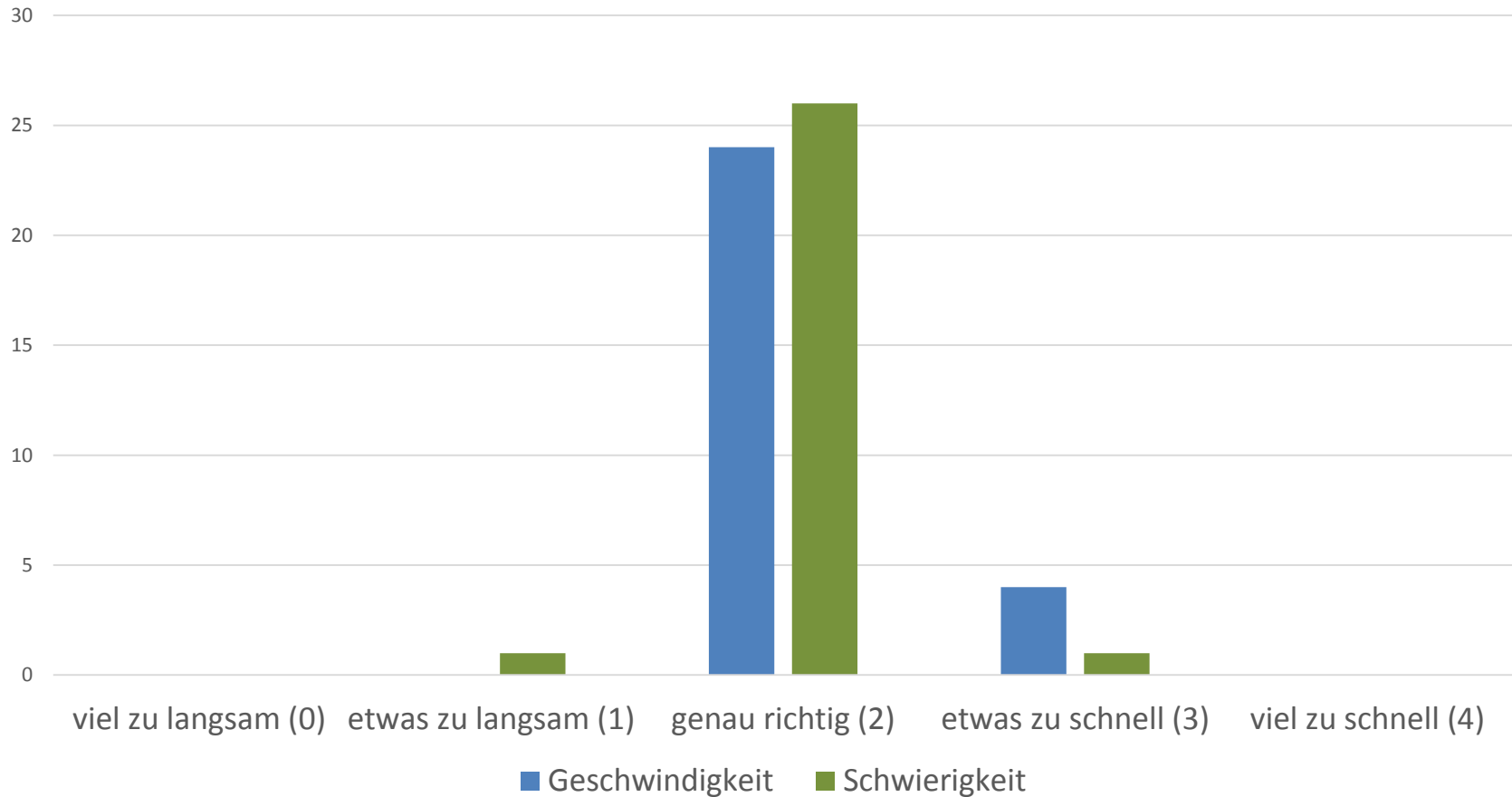


Ergebnisse der Moodle Umfrage

- **Welche der Werkzeuge/Tools haben Sie bereits eingesetzt? (Sonstige)**
 - Eclipse mit Webplugins
 - Browser-Werkzeuge wie die Konsole, Inspektor, etc.
 - **mySQL-Statements via PHP**
 - createjs , three.js, Greensock
 - SQL
 - LESS
 - Gulp
 - PHP Storm, Slim PHP Framework, Netbeans
 - Postman fürs Testen von REST Schnittstellen

Ergebnisse der Moodle Umfrage

Wie empfanden Sie .. im ersten Unterricht?



Agenda

- Wiederholung
- Ergebnisse der Umfragen

Client-Server-Architektur

- C-S-Aufbau
- Client: Kommunikation
- LAMP und MEAN
- Server: Architekturkomponenten
- Auswahl-Kriterien für Komponenten
- Das Nadelöhr

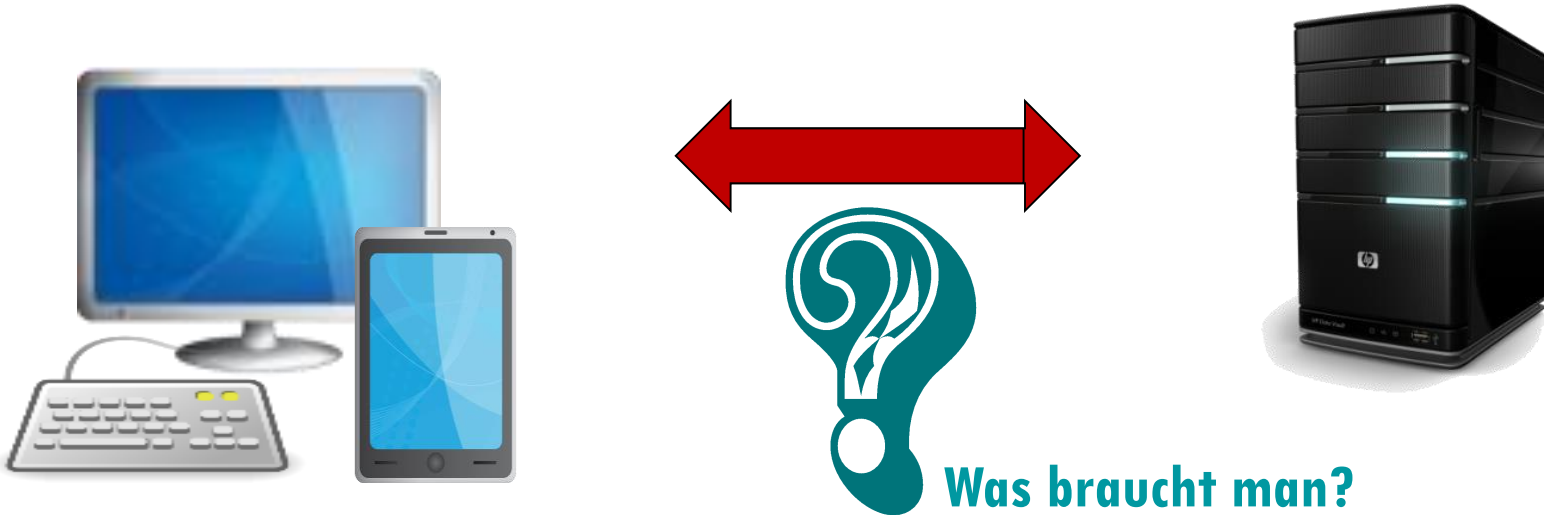
- Node.js: Mein erster Server mit JavaScript

- Zusammenfassende Fragen
- Trends
- Ausblick

■ Semesterplan (vorläufig)

	Datum	Thema	Übung
1	05.04.2016		Ü1: Client-Website
2	12.04.2016	Einführung, Ziele, Ablauf, Benotung	Ü1
3	19.04.2016	Client-Server Architektur	Ü2: Server mit node.js
4	26.04.2016	REST-APIs	Ü2
5	03.05.2016	REST in node.js	- Feiertag - (5.5.)
6	10.05.2016	Debugging und Testen	Ü3: API mit node.js
7	17.05.2016	Strukturierung, Modularisierung	Ü3
8	24.05.2016	Vertiefung einzelner Themen	Ü4: Umfangreiche REST API
9	31.05.2016	Datenhaltung, SQL, NoSQL, primär mit MongoDB	Ü4
10	07.06.2016	backbone.js als Gegenpart zu REST/node	Ü4
11	14.06.2016	Authentifizierung und Patterns	Ü5: mongoDB-Anbindung
12	21.06.2016	Mobile Development/Cross-Plattform-Development	Ü5
13	28.06.2016	Gastdozent(en)	Ü6: Backbone.js
14	05.07.2016	Klausurvorbereitung	Ü6
15	12.07.2016	Klausur PZR1 (Di, 12.07. 12:15 Uhr, Ingeborg-Meising-S.)	-
16	19.07.2016	Klausureinsicht	-
	21.09.2016	Klausur PZR2 (Mi, 21.09. 12:15 Uhr, Ingeborg-Meising-S.)	-

Client-Server Architekturen



- **Physikalische Verbindung (Wifi/Kabel)**
- **Client-Anwendung (Browser)**
- **Protokolle (unser Fokus: Anwendungsprotokolle)**
- **Speicher/Persistenz**
- **Webserver-Deamon**
- **Schnittstelle(n) / APIs**
- **Programmiersprache auf Serverseite**
- **Programmiersprache auf Clientseite**

Agenda

- Wiederholung
- Ergebnisse der Umfragen

Client-Server-Architektur

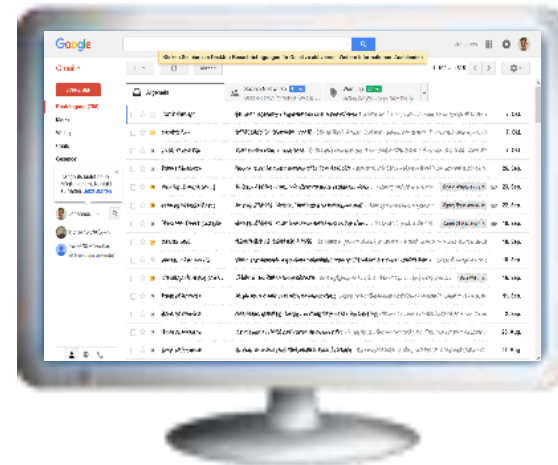
- C-S-Aufbau
- Client: Kommunikation
- LAMP und MEAN
- Server: Architekturkomponenten
- Auswahl-Kriterien für Komponenten
- Das Nadelöhr
- Node.js: Mein erster Server mit JavaScript
- Zusammenfassende Fragen
- Trends
- Ausblick

Client-Server: 2 Szenarien

- **Eher statische Website (Beuth Website)**

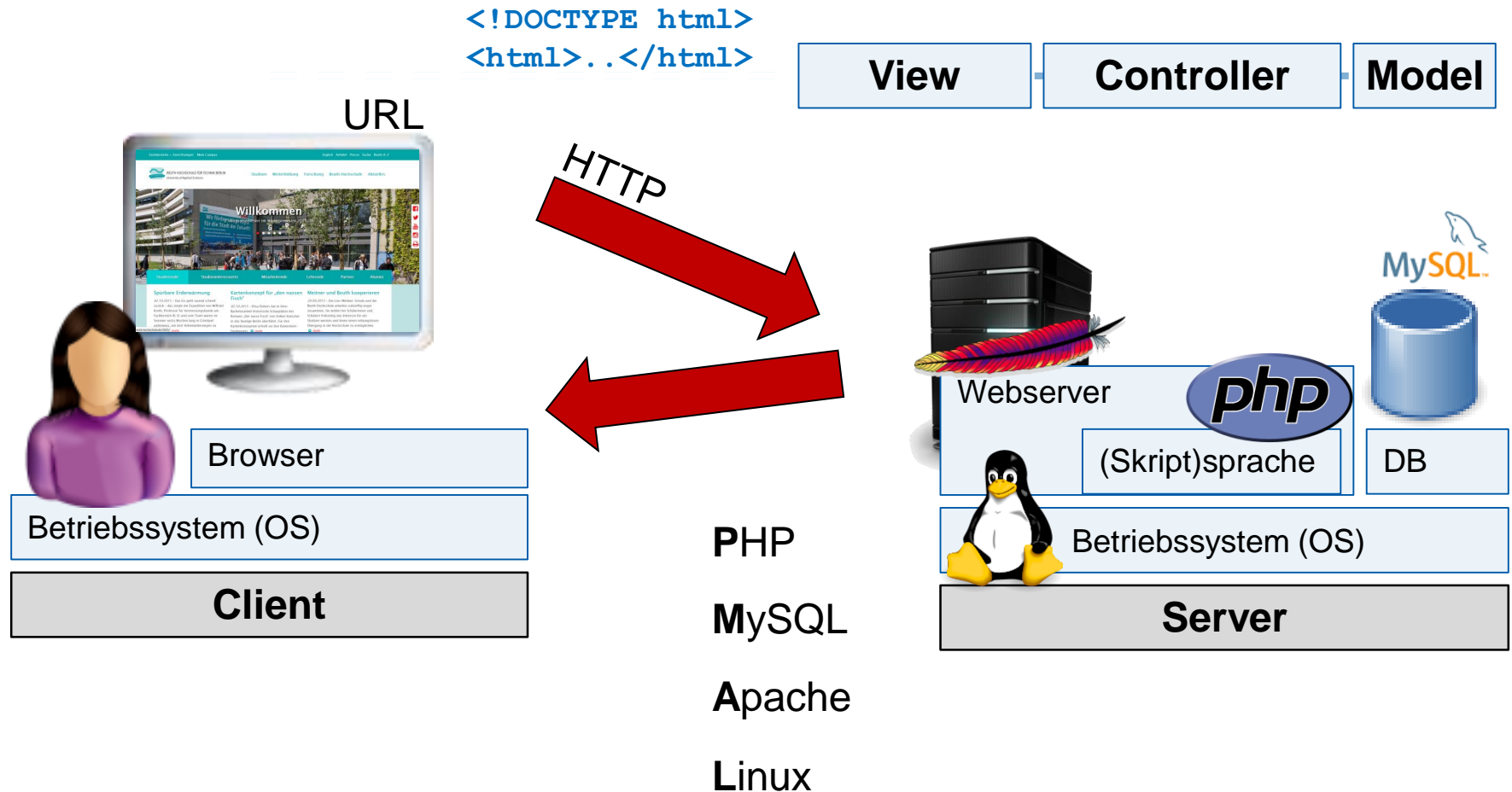


- **SinglePageApp (Gmail)**



Client-Server Architekturen

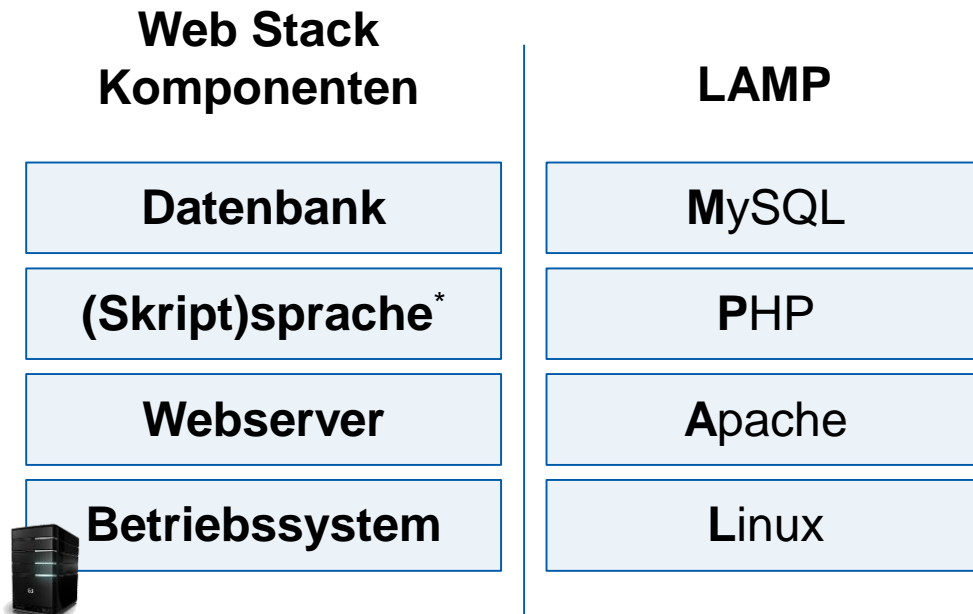
■ Thin-Client-Prinzip (Beispiel: Szenario der statischen Webseite)



Client-Server Architekturen: LAMP

Der Web Application Solution Stack (kurz: Web Stack)

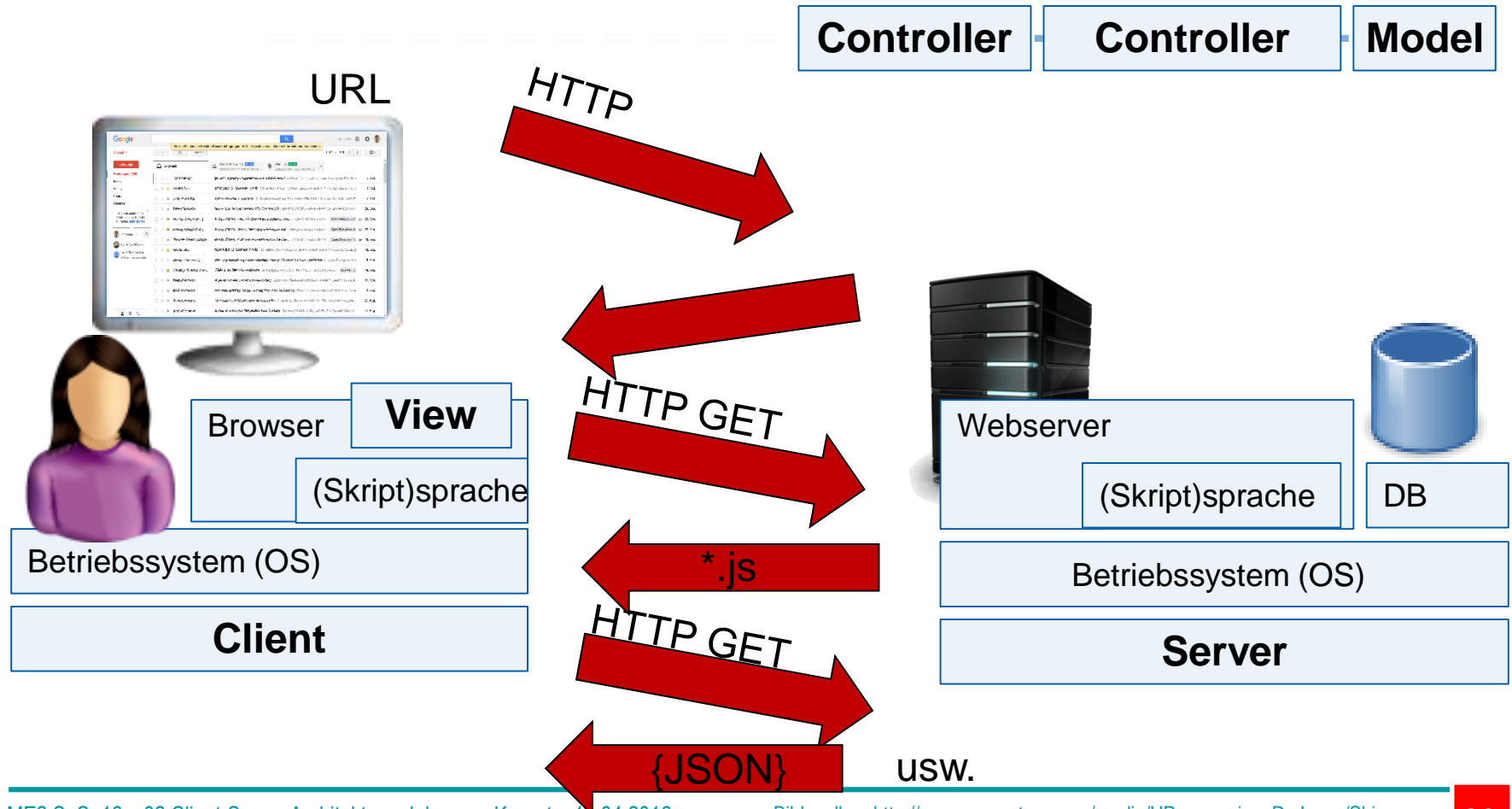
- ..alle Softwarekomponenten, die zum Betreiben der Web Anwendung notwendig sind



Client-Server Architekturen

- **Rich-Client-Prinzip** (Beispiel: Szenario der SinglePageApp wie Gmail)

```
<!DOCTYPE html>
<html>..<script>..</script>..</html>
```





Anmelden

Konto erstellen

Gmail Funktionen Für Mobilgeräte For work Hilfe

Elements Network Sources Timeline Profiles Resources Audits Console

View: [Icons] Preserve log Disable cache No throttling

Name	Method	Status	Type	Initiator	Size	Time	Timeline – Start Time	40.00 s
maia.js	GET	304	script	about.html:522	0 B	1.94 s		
ga.js	GET	304	script	google.js:7	(from cac...	0 ms		
maia.experimental.css	GET	304	stylesheet	about.html:50	0 B	1.41 s		
lightbox.css	GET	304	stylesheet	about.html:50	0 B	1.41 s		
DXI1ORHCpsQm3Vp6mXoaTegdm0LZdjq...	GET	200	font	ga.js:28	(from cac...	0 ms		
cJZKeOuBrn4kERxqtaUH3VtXRa8TVwTICg...	GET	200	font	ga.js:28	(from cac...	0 ms		
k3k702ZOKiLJc3WWjuplzOgdm0LZdjqr5-o...	GET	200	font	ga.js:28	(from cac...	0 ms		
cb=gapi.loaded_0	GET	200	script	plusone.js:7	(from cac...	0 ms		
cb=gapi.loaded_1	GET	200	script	plusone.js:7	(from cac...	0 ms		
fastbutton?usegapi=1&origin=https%3A...	GET	200	document	plusone.js:23	10.9 KB	1.85 s		
postmessageRelay?parent=https%3A%2F...	GET	200	document	cb=gapi.loaded_1...	(from cac...	1 ms		
api.js	GET	304	script	postmessageRela...	(from cac...	0 ms		
core:rpc:shindig.random:shindig.sha1.js?c...	GET	200	script	postmessageRela...	(from cac...	0 ms		
2408744513-postmessengerelay.js	GET	200	script	postmessageRela...	(from cac...	0 ms		
css?family=Open+Sans	GET	200	stylesheet	about.html:516	(from cac...	0 ms		
data:image/png;base...	GET	(data)	png	gmail.min.js:38	(from cac...	0 ms		
about-carousel-3.jpg	GET	200	jpeg	gmail.min.js:38	1.1 MB	44.00 s		
about-carousel-2.jpg	GET	200	jpeg	gmail.min.js:38	1.1 MB	44.08 s		
about-carousel-1.jpg	GET	200	jpeg	gmail.min.js:38	1.1 MB	44.20 s		
googlemail-16.png	GET	200	png	gmail.min.js:38	(from cac...	0 ms		
cleardot.gif	GET	200	gif	gmail.min.js:37	(from cac...	0 ms		
activity?src=2542116;type=gmail862;cat=...	GET	200	document	doubletrack.js:14	177 B	1.34 s		

45 requests | 3.4 MB transferred | Finish: 51.66 s | DOMContentLoaded: 3.54 s | Load: 47.67 s



Einmal anmelden. Alle Google Produkte nutzen

Anmelden, um zu Gmail zu gelangen

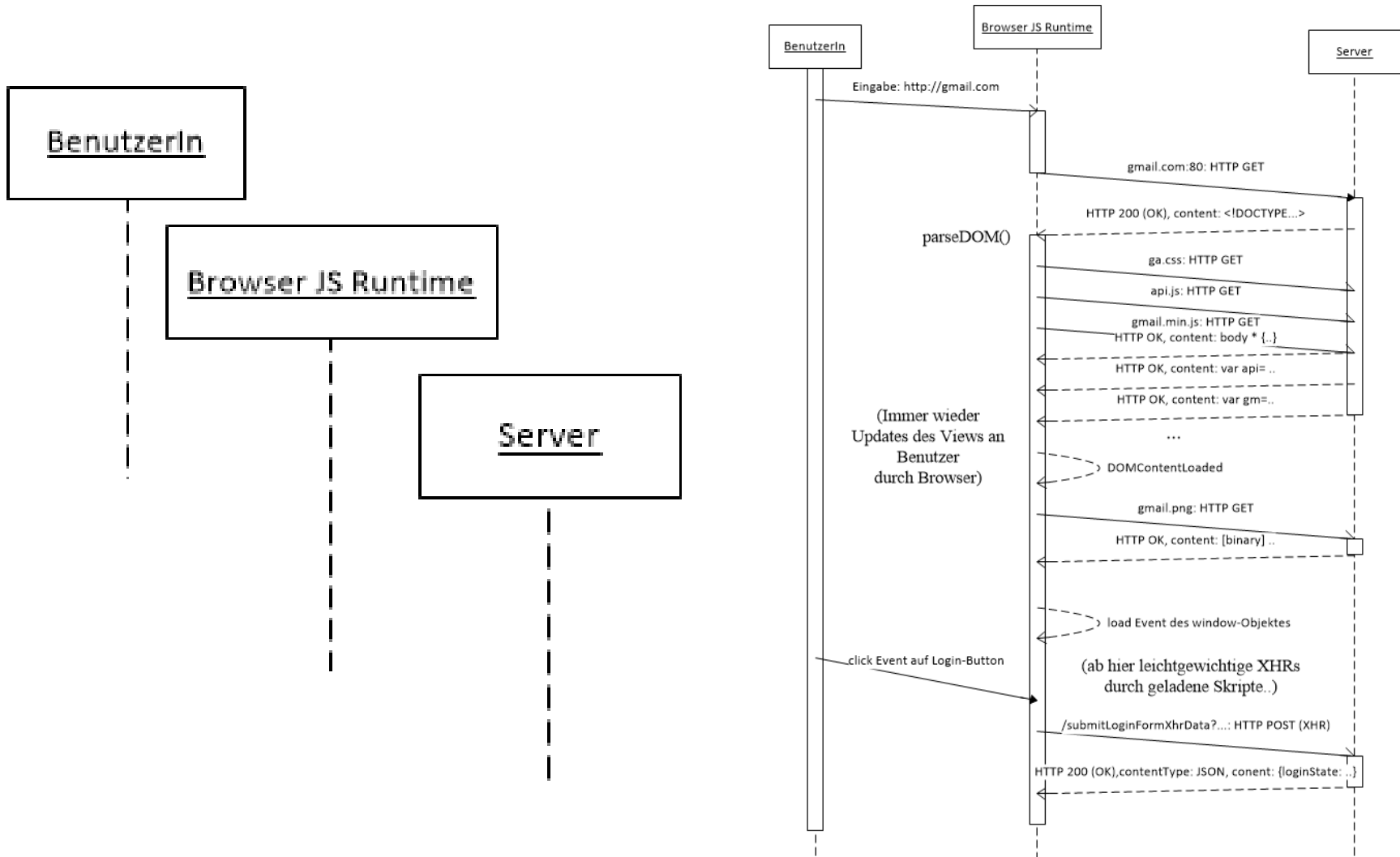
@gmail.com

Anmelden

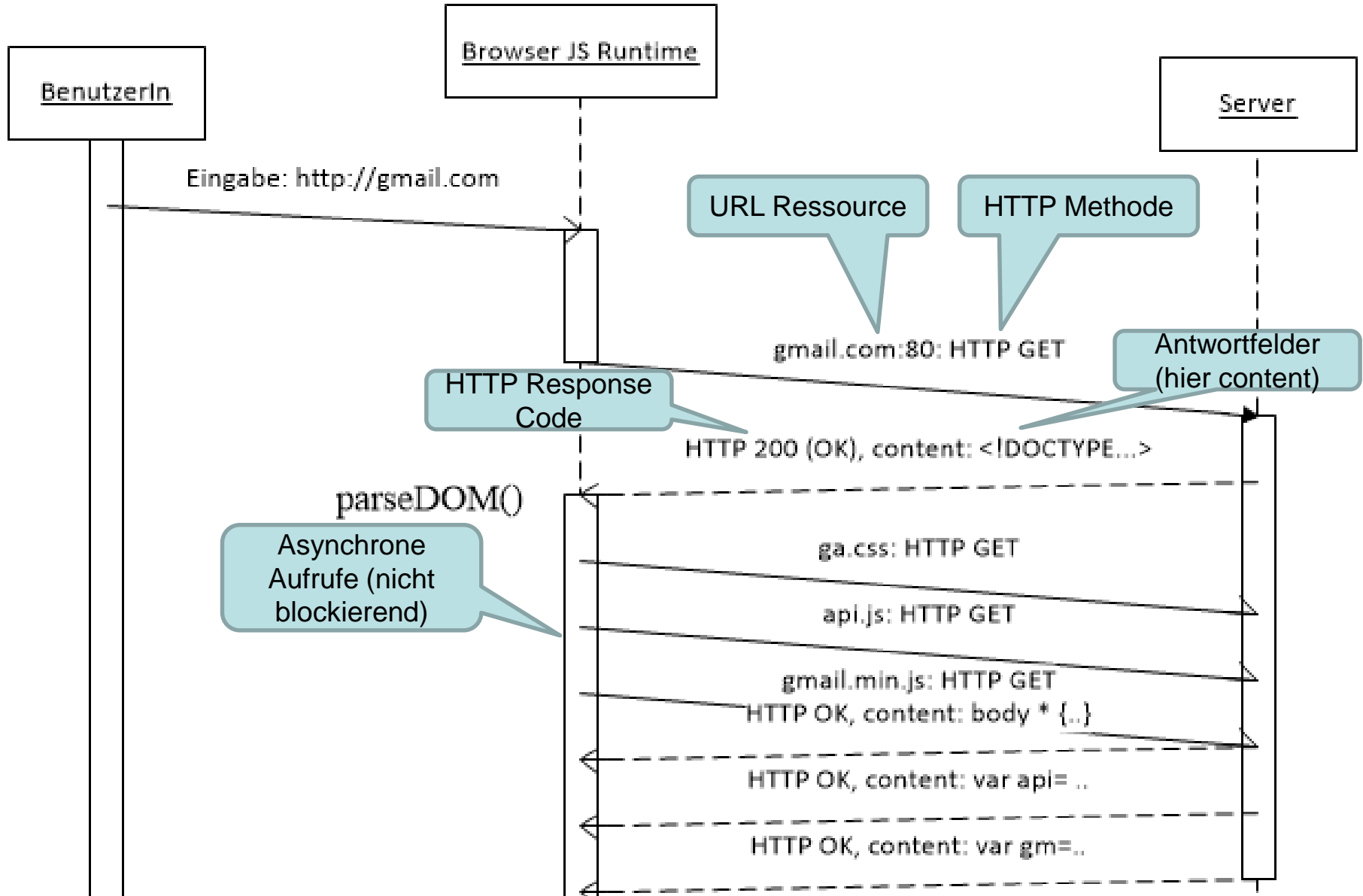
☒ Angemeldet bleiben
 [Passwort vergessen?](#)

<div> <div> <div>🔍</div> <div>📱</div> </div> <div> <div>Elements</div> <div>Network</div> <div>Sources</div> <div>Timeline</div> <div>Profiles</div> <div>Resources</div> <div>Audits</div> <div>Console</div> </div> </div>							
<div> <div> <div>🔴</div> <div>🚫</div> </div> <div> <div>📺</div> <div>🔍</div> </div> <div>View: <div>📄</div> <div>📊</div></div> <div> <input type="checkbox"/> Preserve log <input type="checkbox"/> Disable cache </div> <div>No throttling ▼</div> </div>							
Name	Method	Status	Type	Initiator	Size	Time	Timeline – S
<input type="checkbox"/> accountLoginInfoXhr	POST	200	xhr	ServiceLogin?serv...	284 B	1.03 s	
<input checked="" type="checkbox"/> checkmark.png	GET	200	png	ServiceLogin?serv...	239 B	554 ms	
photo.jpg?sz=96	GET	200	png	Other	1.4 KB	3.02 s	
gxl?email= %40gmail.co...	GET	204	text/plain	Other	0 B	1.28 s	

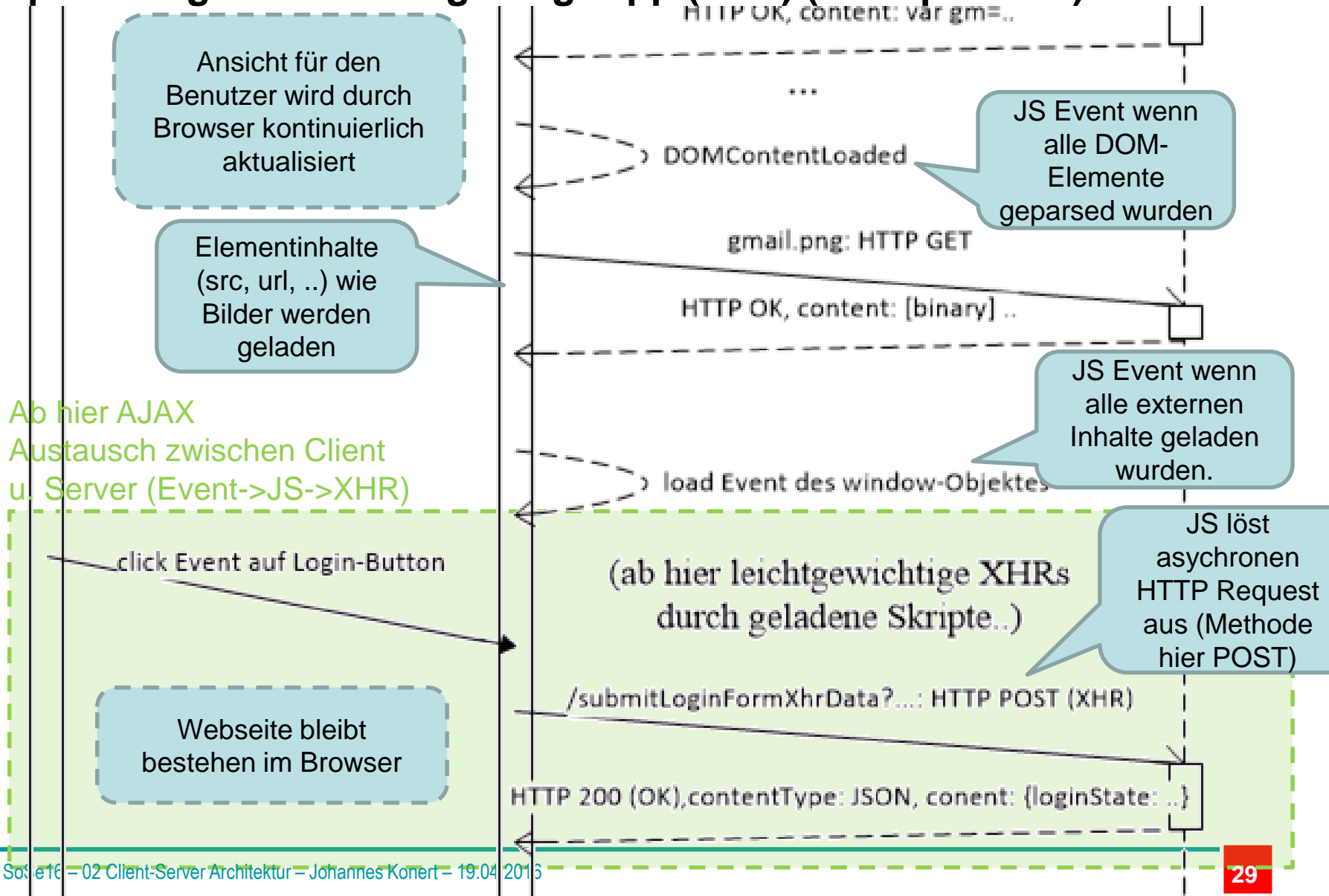
Sequenzdiagramm zu SinglePageApp (SPA) (exemplarisch)



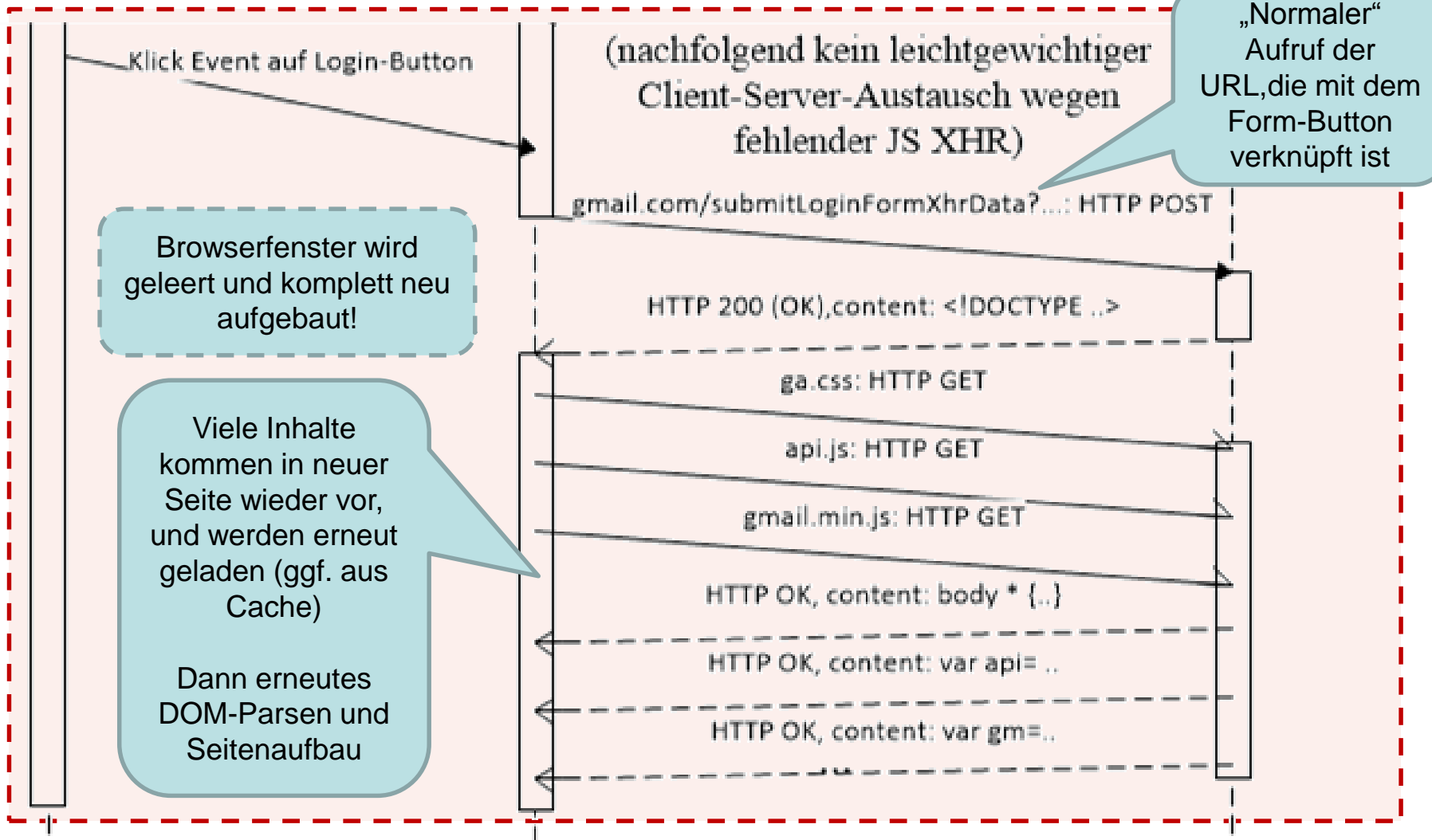
Sequenzdiagramm zu SinglePageApp (SPA) (exemplarisch)



Sequenzdiagramm zu SinglePageApp (SPA) (exemplarisch)



Sequenzdiagramm: Zum Vergleich ohne JS (Thin client exemplarisch)



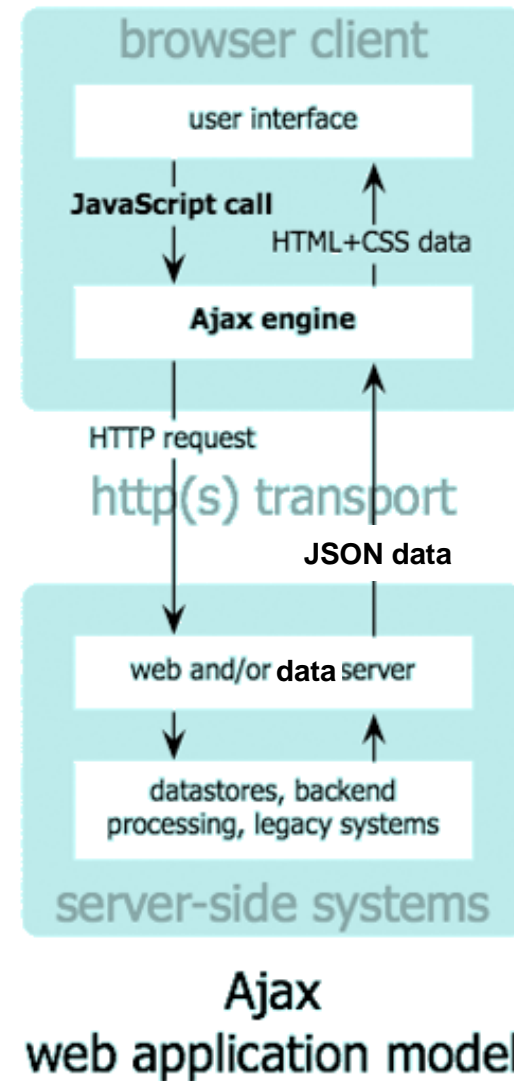
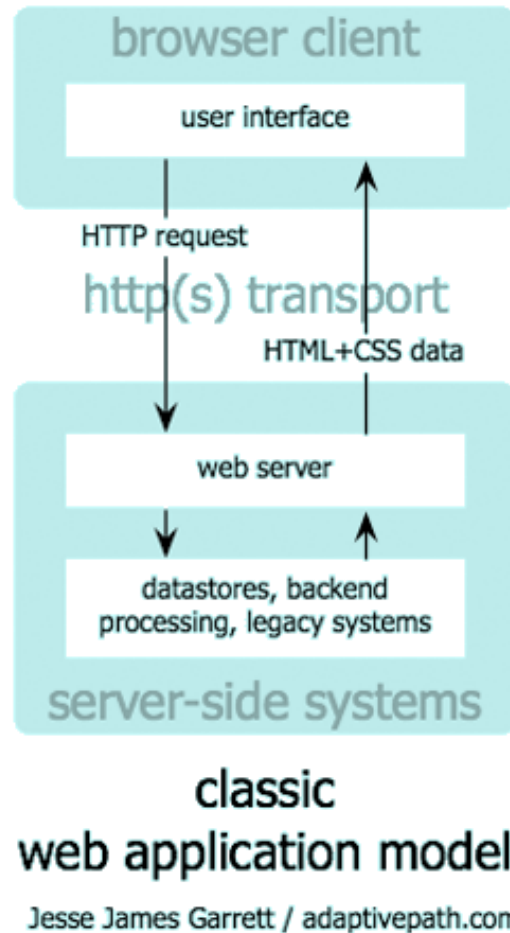
AJAX

Asynchronous JavaScript And XML

- Asynchroner Austausch von Daten zwischen Browser und Server

(heute eher AJAJ: Asynchronous JavaScript And JSON)

AJAX



AJAX Beispiele

```
// Kompatibel zu with IE7+, Firefox, Chrome, Opera, Safari

var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState === 4 && xmlhttp.status === 200) {
        alert(xmlhttp.responseText);
    }
}

xmlhttp.open("GET", "test.php", true); // 3. Parameter → Async
xmlhttp.send();
```

AJAX Beispiele

■ jQuery:

```
$.ajax({  
    url: "test.php",  
    success: function(response) {  
        alert(response);  
    }  
});
```

■ Noch kürzer:

```
$.get("test.php", function(response) {  
    alert(response);  
});
```

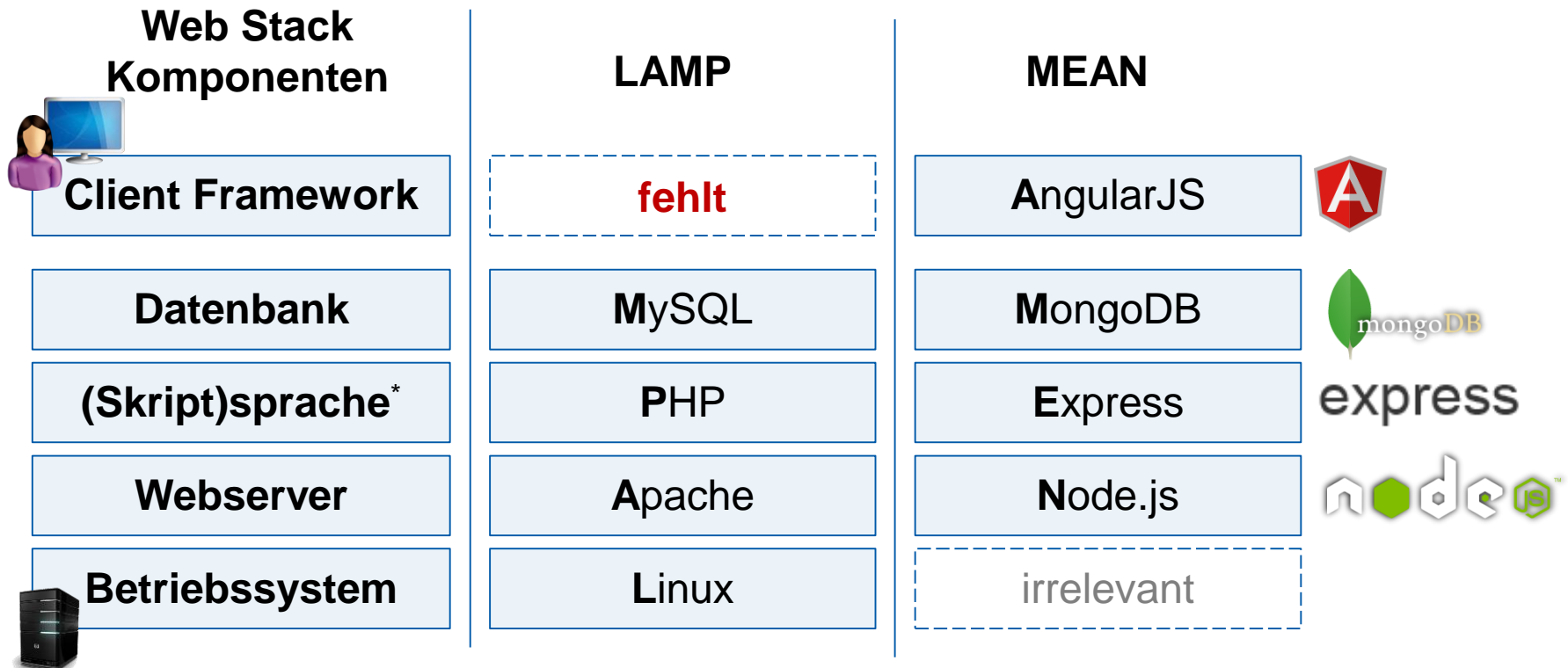
Agenda

- Wiederholung
- Ergebnisse der Umfragen

Client-Server-Architektur

- C-S-Aufbau
- Client: Kommunikation
- LAMP und MEAN
- Server: Architekturkomponenten
- Auswahl-Kriterien für Komponenten
- Das Nadelöhr
- Node.js: Mein erster Server mit JavaScript
- Zusammenfassende Fragen
- Trends
- Ausblick

Client-Server-Architekturen





Agenda

- Wiederholung
- Ergebnisse der Umfragen

Client-Server-Architektur

- C-S-Aufbau
- Client: Kommunikation
- LAMP und MEAN
- **Server: Architekturkomponenten**
- Auswahl-Kriterien für Komponenten
- Das Nadelöhr
- Node.js: Mein erster Server mit JavaScript
- Zusammenfassende Fragen
- Trends
- Ausblick

Backend (BE) und Frontend (FE)

	Nähe zum Benutzenden	Primäre Aufgabe	Benutzerorientierung
 Frontend	nah	Benutzerschnittstelle (View, Feedback)	1 Benutzer, Personalisierung
 Backend	fern	API-Schnittstelle, Aufgabenverarbeitung, Datenhaltung	<i>n</i> Benutzer, Verwaltung, Authentifizierung

Web Stack: Vielfalt dank Modularität



Datenbank (~50)	Webserver (~30)	Server: Web App Framework (~130)	Template Engine (~90)	Client: Web App Framework (JS: ~40)
SQLite HyperSQL MySQL PostgreSQL Cassandra MongoDB	(Microsoft IIS) Apache HTTP Apache Tomcat Jetty Boa NginX Mongoose WS lighttpd (Node.js)	<div>Full solution frameworks: ASP.NET MVC, GWT, ..</div> PHP (CakePHP, Zend) Ruby (on Rails) Python (Django, Pyramid) Java Servlets (Spring, JSF, Struts) ExpressJS (..sails)	PHP / Smarty Genshi Cheetah Mustache JSP Jade	Dojo MochiKit script.aculo.us ExtJS YUI Qooxdoo jQuery backbone.js Ember.js AngularJS REACT

Agenda

- Wiederholung
- Ergebnisse der Umfragen

Client-Server-Architektur

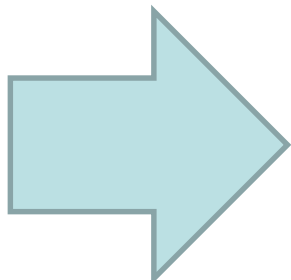
- C-S-Aufbau
- Client: Kommunikation
- LAMP und MEAN
- Server: Architekturkomponenten
- Auswahl-Kriterien für Komponenten
- Das Nadelöhr
- Node.js: Mein erster Server mit JavaScript
- Zusammenfassende Fragen
- Trends
- Ausblick

Kriterien für die Komponentenauswahl




Datenbank (~50)	Webserver (~30)	Server: Web App Framework (~130)	Template Engine (~90)	Client: Web App Framework (JS: ~40)
		Full solution frameworks		

- (Web)Server? Datenbank? Full-Solution-Framework oder Skriptsprache+TemplateEngine+Client-Framework?




**Entscheidung hängt vom
Anwendungsszenario ab!**

Kriterien für die Komponentenauswahl



Datenbank	Webserver	Full Solution Framework	Server: Web App Framework	Template Engine	Client: Web App Framework





Aufgabe:

- Nehmen Sie Ihr Szenario (10sec)
 - (1) Thin-Client-Anwendung (Beispiel Beuth-Webseite)
 - (2) Rich-Client-Anwendung (Beispiel Gmail)
- Diskutieren Sie im kleinen Team, (3min)
 - Welche Anforderungen muss ihr **Webserver** erfüllen (Welche Daten liefert er hauptsächlich) ?
 - Welche Anforderungen haben Sie an Ihr **Server-Framework** und die **Programmiersprache**?
 - Brauchen Sie ein **Client-Framework** und wenn ja, was muss es können (Templates, Interaktion, Datenaustausch, Responsive...) ?
- Tafelsammlung: Jedes Team kommt mal dran



Kriterien für die Komponentenauswahl



Datenbank	Webserver	Full Solution Framework	Server: Web App Framework (+Sprache)	Template Engine	Client: Web App Framework 
	Performance <ul style="list-style-type: none"> ▪ statische Dateien ▪ Streaming ▪ (Skript)-sprachen 		<ul style="list-style-type: none"> ▪ Entwicklungsgeschwindigkeit ▪ Modularisierung ▪ Knowhow eigener Entwickler/in 		<ul style="list-style-type: none"> ▪ Größe (kb) ▪ Binding ▪ Modularisierung ▪ Kompatibilität

■ Generelle Kriterien für alle SW-Produkte

- **Speicherverbrauch**
- **Stabilität**
- **Sicherheit**
- **Geschwindigkeit**
- **Dokumentation**
- **Weiterentwicklung/Community**

Agenda

- Wiederholung
- Ergebnisse der Umfragen

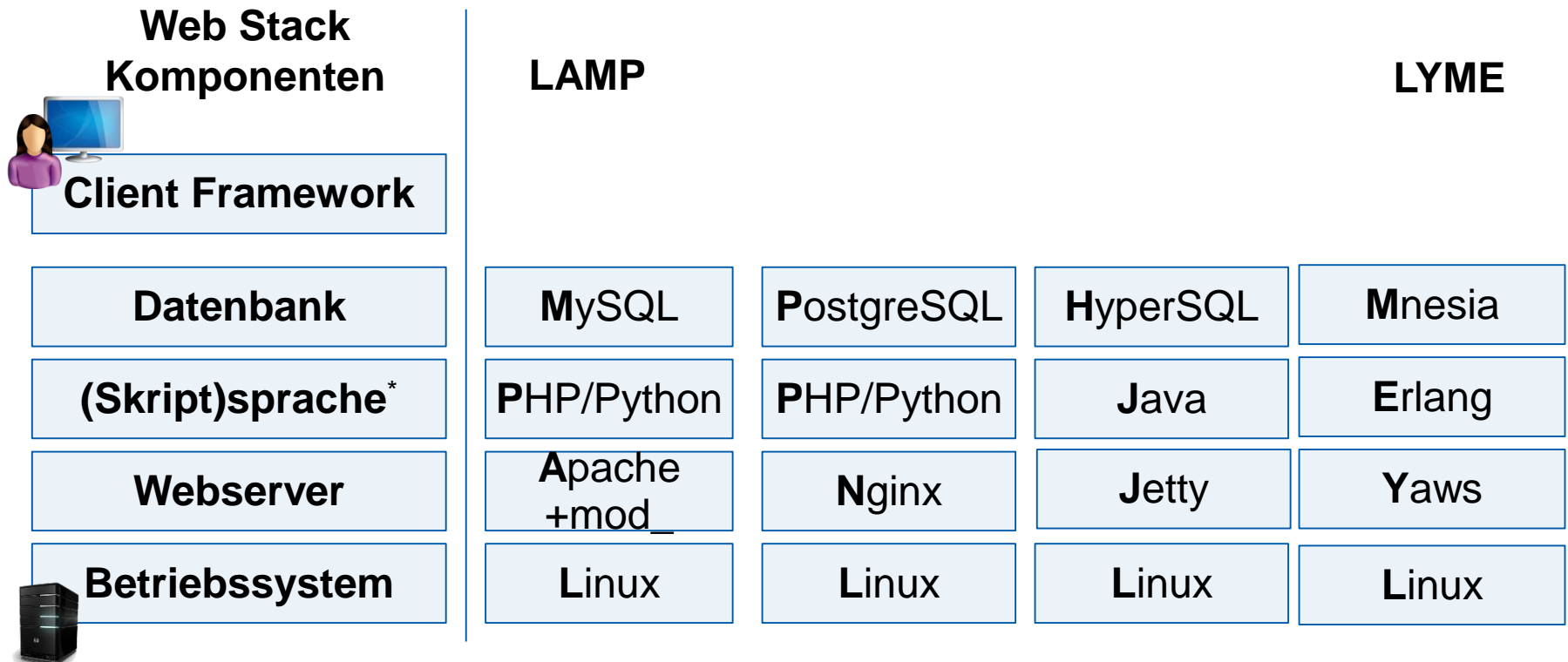
Client-Server-Architektur

- C-S-Aufbau
- Client: Kommunikation
- LAMP und MEAN
- Server: Architekturkomponenten
- Auswahl-Kriterien für Komponenten
- Das Nadelöhr

Web Stack Beispiele

- Node.js: Mein erster Server mit JavaScript
- Zusammenfassende Fragen
- Trends
- Ausblick

Beispiele zu Thin-Client-Stacks

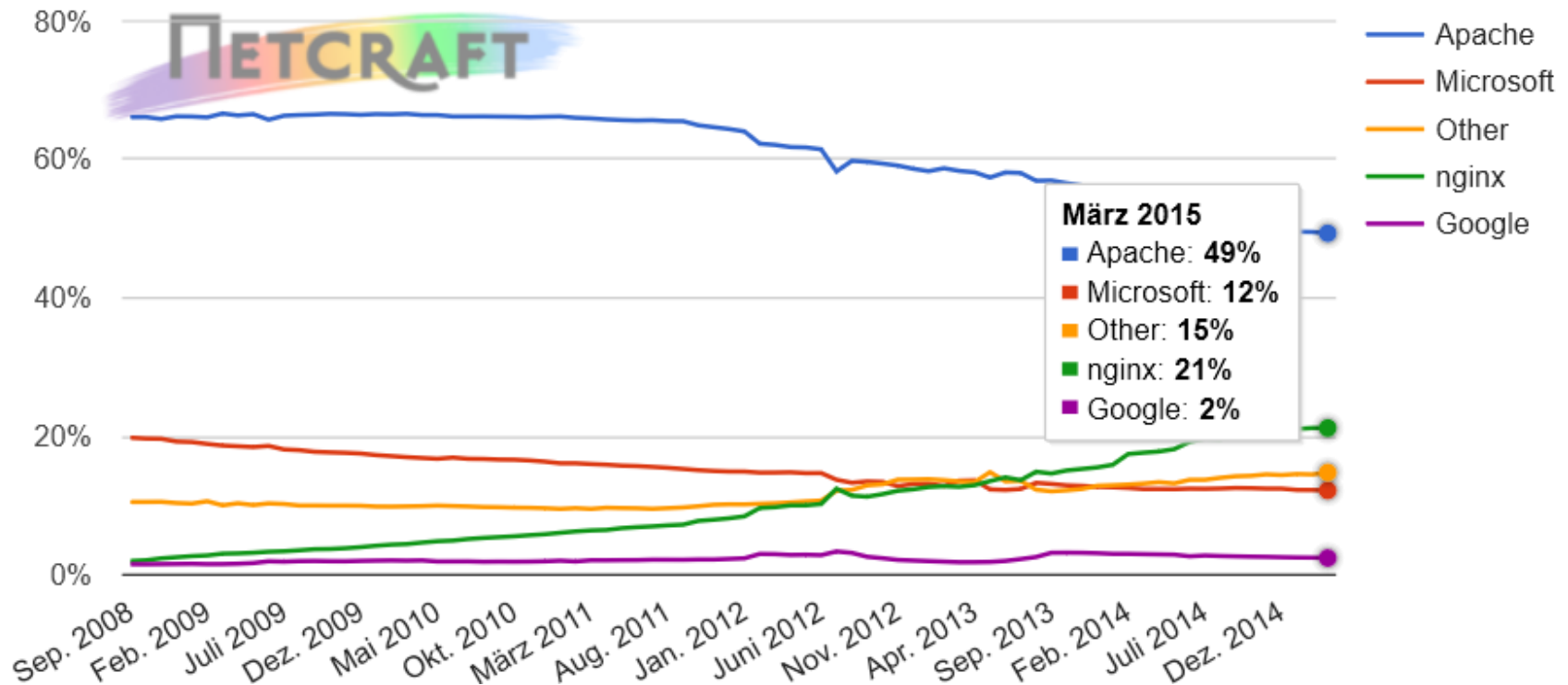


Vergleich der Webserver: Nutzungshäufigkeit

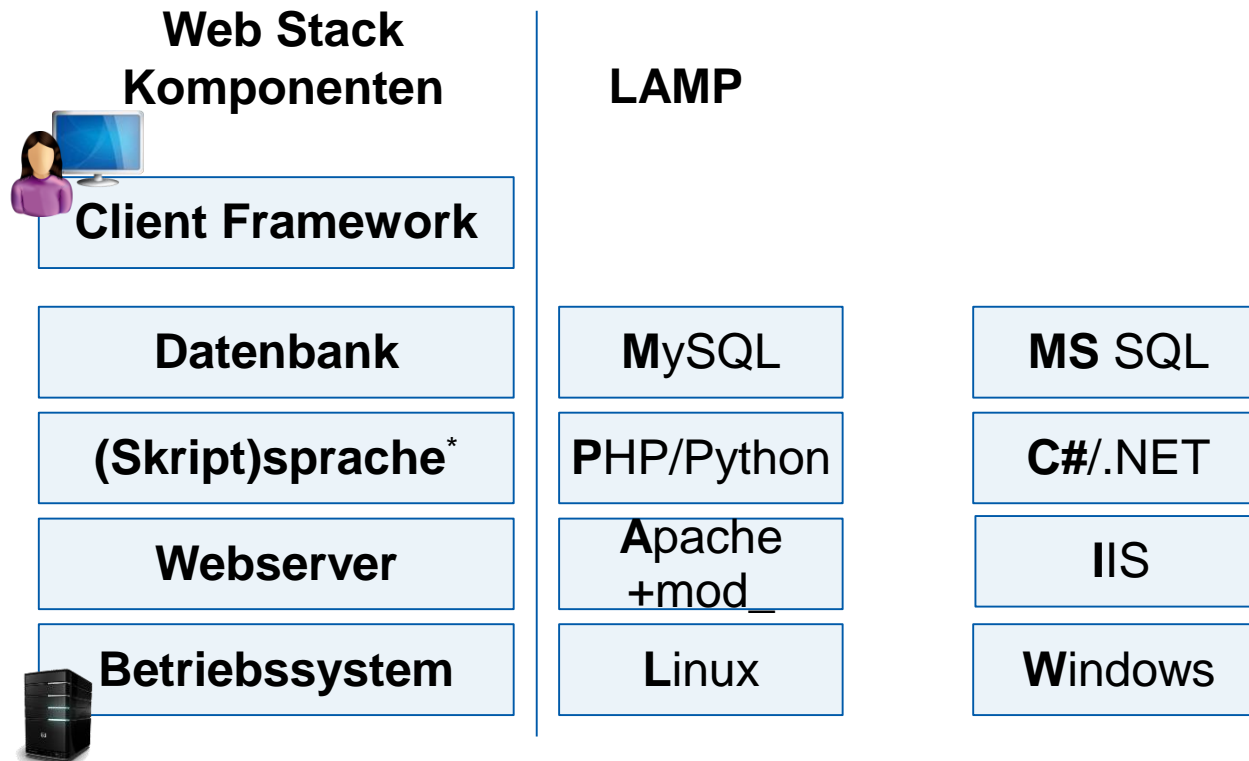
■ Stärken:

- Apache: Sprachanbindungen, 1 Thread per Request
- nginx: statische Files, viele Requests pro Thread
- Microsoft: .NET Support

Web server developers: Market share of the top million busiest sites



Beispiele zu Thin-Client-Stacks



Nutzung von XAMP-ähnlichen Stacks für Thin Client Anwendungen

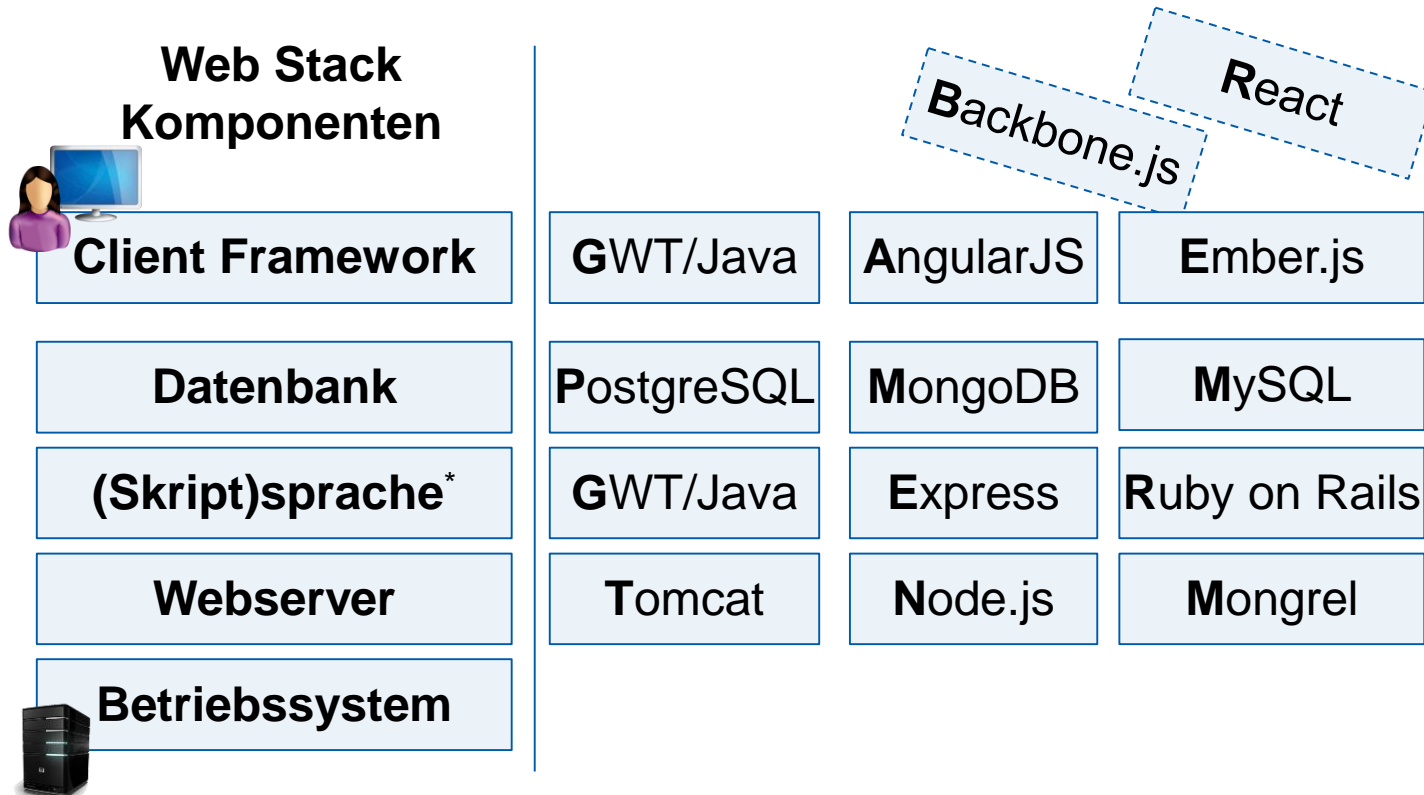
Pro

- Performant für statische Inhalte (Caching, Kompression, Zugriffssteuerung)
- Schwergewichtige Webserver für Sicherheit und Verfügbarkeit
- sehr starke Verbreitung
- Vielzahl fertiger Apps (Blog, Email, ..)

Contra

- Viele Sprachenkenntnisse nötig (Apache-Konfig, PHP/Python, MySQL, JavaScript)
 - → außer bei LYME!
- für kleinere WebApps zu starr/
viel Aufwand, da einfache Client-App-Anbindung im Stack fehlt.

Beispiele zu Rich-Client-Stacks für WebApps



Nutzung von Client JS Frameworks

■ Backbone.js



- 2010 (Community)
- M(VC) einfach, verbreitet

■ AngularJS



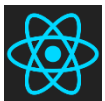
- 2009 von Google
- MVVM 2-Wege-Binding über Direktiven

■ Ember

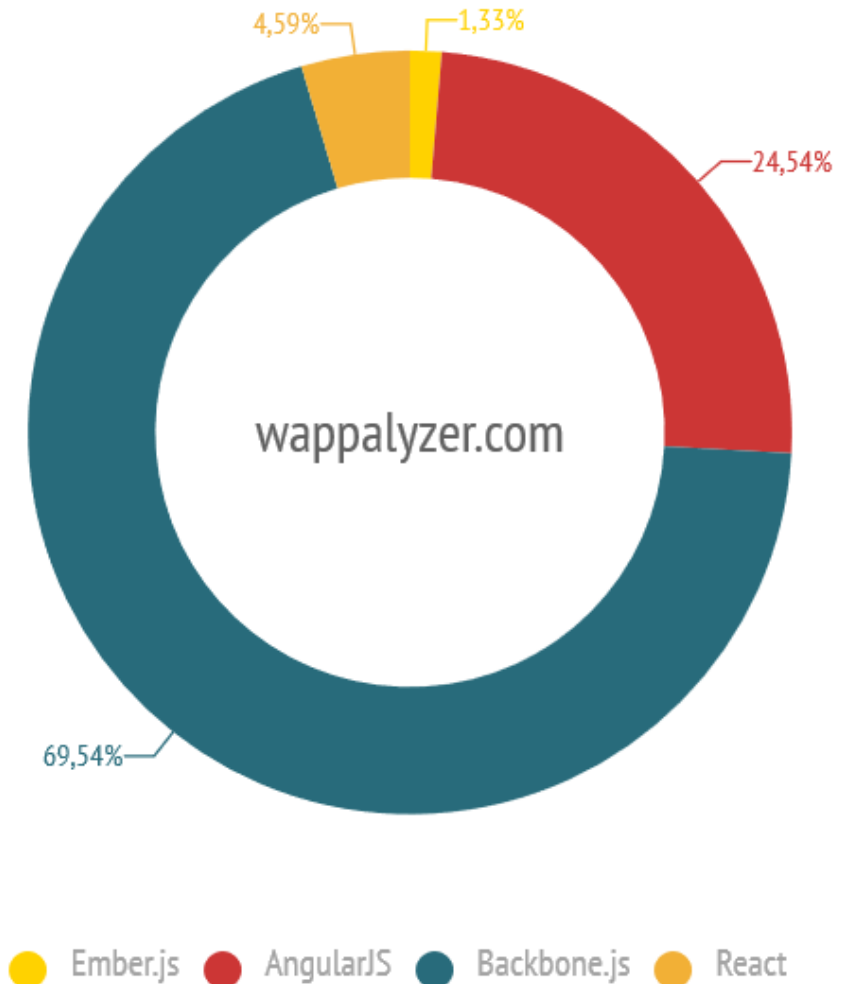


- 2011 von Katz & Dale
- MVC

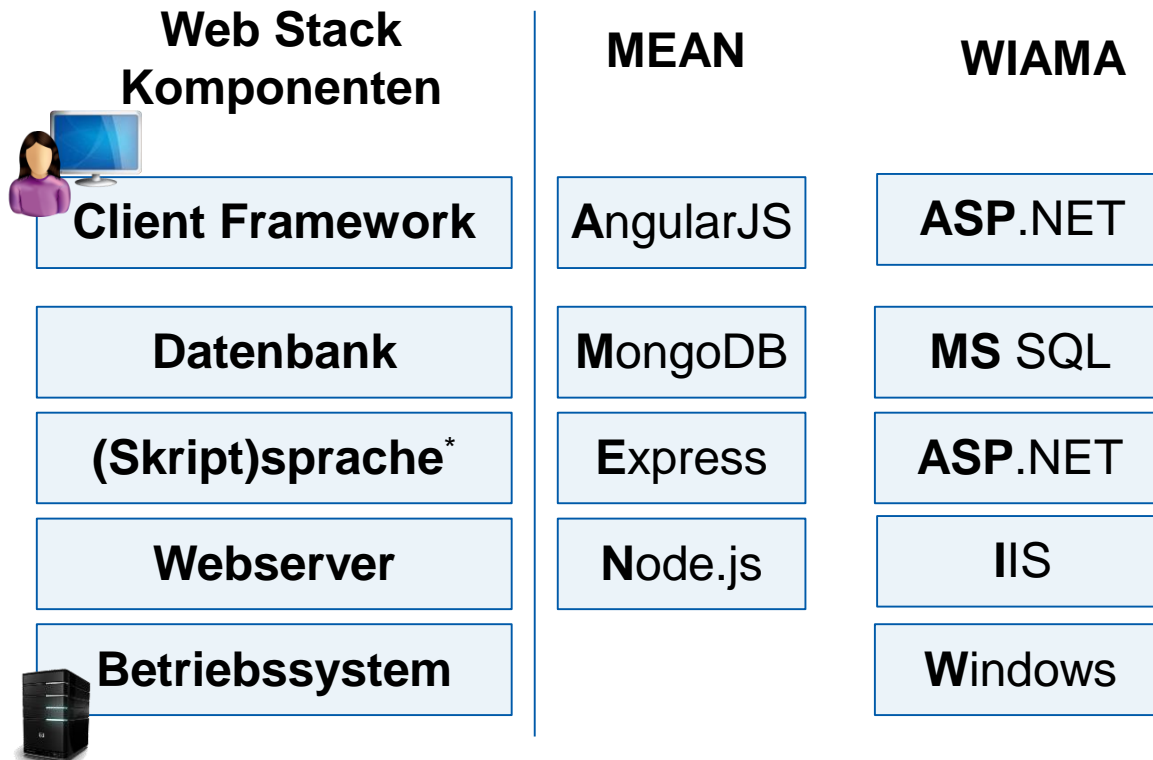
■ React



- 2013 von facebook
- 1-Weg View



Beispiele zu Rich-Client-Stacks für WebApps



Nutzung von MEAN-ähnlichen Stacks für Rich Client Anwendungen

Pro

- Performant für dynamische Inhalte (Event-basiert, nicht-blockierend)
- Leichtgewichtig für Client-Server Datenaustausch
- JavaScript als Basissprache (DB, Server, Client)
- Bridge zwischen Webserver und Skriptsprache entfällt



Contra

- Funktionale Programmierung via Callbacks ist komplexer
- Schwache Typisierung kann zu mehr Laufzeitfehlern führen (bei JS)
- fehlende Rückwärtskompatibilität zu alten Templates/Systemen
- kein Fokus auf Auslieferung statischer Dateien (Zugriff, Listing, ..)

Agenda

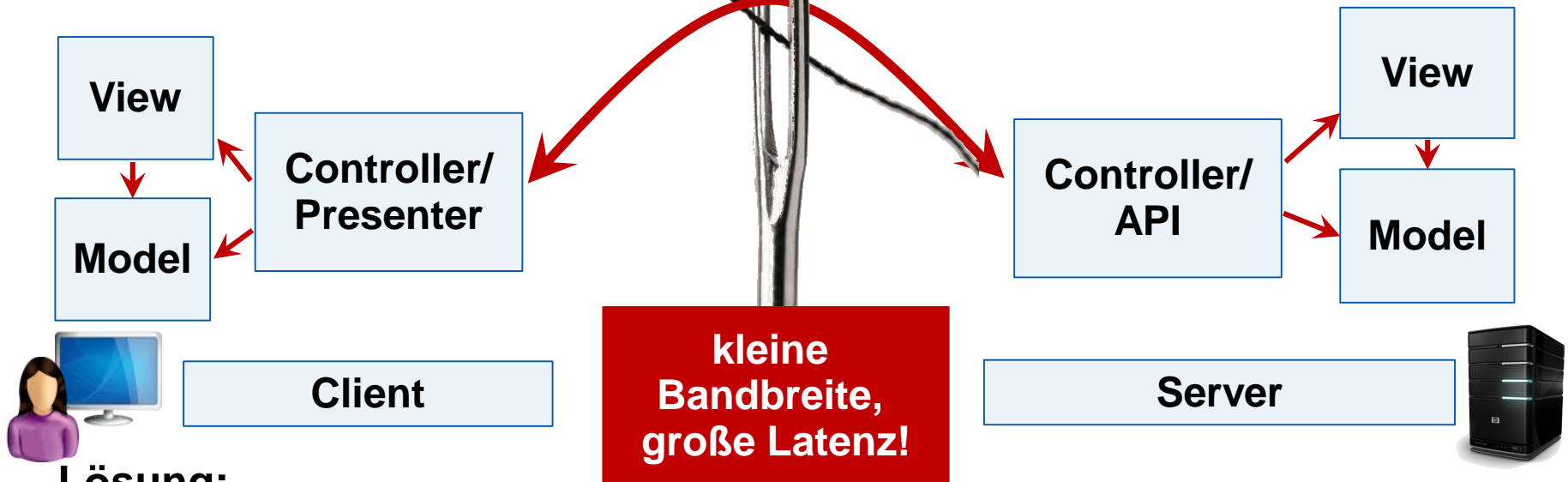
- Wiederholung
- Ergebnisse der Umfragen

Client-Server-Architektur

- C-S-Aufbau
 - Client: Kommunikation
 - LAMP und MEAN
 - Server: Architekturkomponenten
 - Auswahl-Kriterien für Komponenten
 - Das Nadelöhr
-
- Node.js: Mein erster Server mit JavaScript
 - Zusammenfassende Fragen
 - Trends
 - Ausblick

Das Client-Server-Nadelöhr

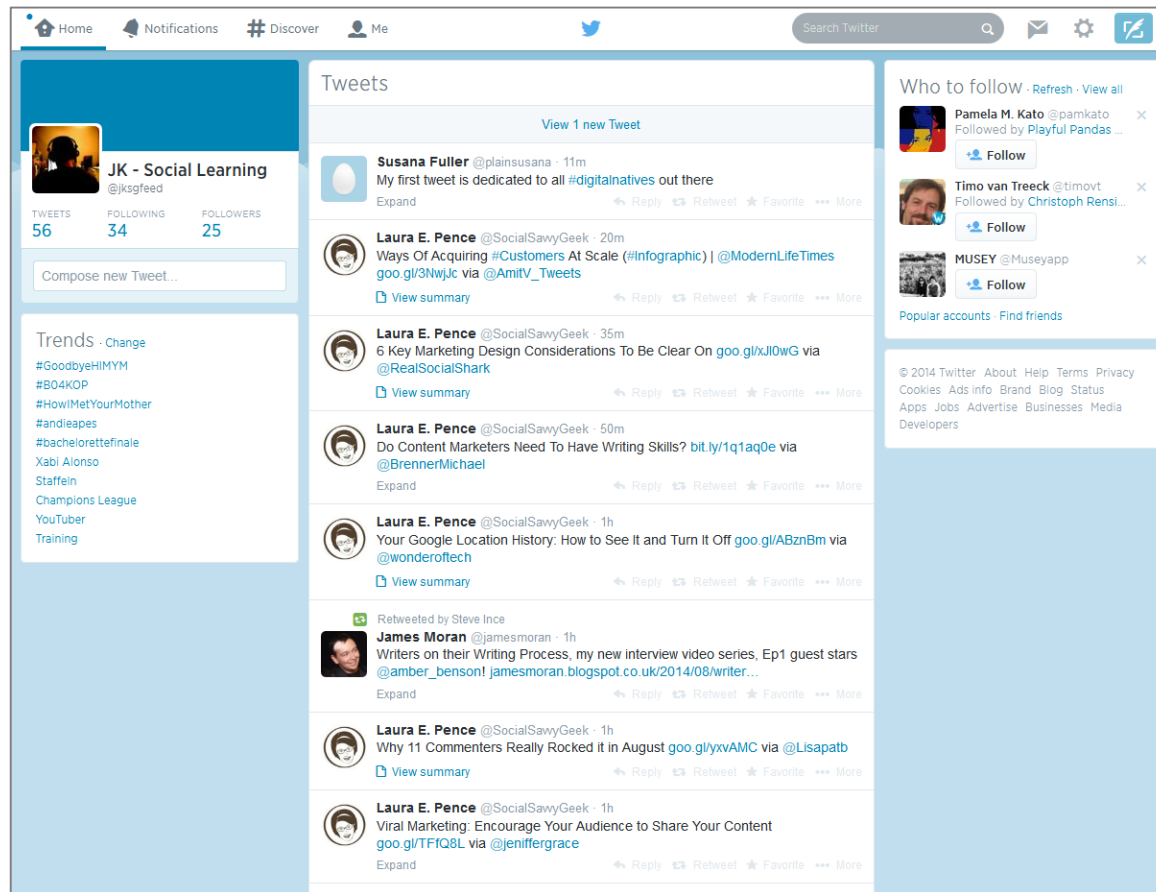
HTML, CSS, JPG, PNG
JS, JSON



Lösung:

1. Frameworks, welche Datenaustausch kapseln (Full-Stack), oder
2. Asynchrones Laden und Speichern per JS
 - REST API auf dem Server
 - JSON Objekte zur Kommunikation

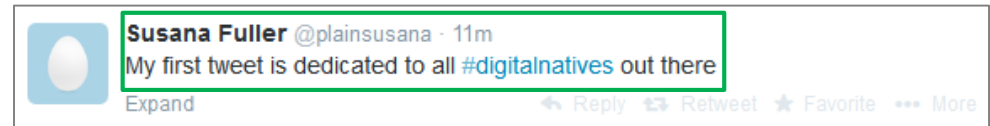
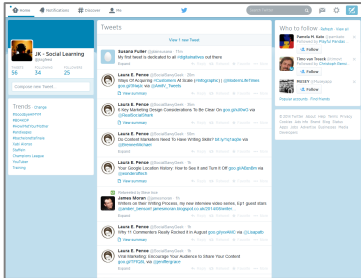
Vergleich: Twitter Webseite mit LAMP (ohne JS) vs. AJAX-Nachladen bspw. MEAN (mit JS)



Vergleich: Twitter Webseite mit LAMP (ohne JS) vs. AJAX-Nachladen bspw. MEAN (mit JS)

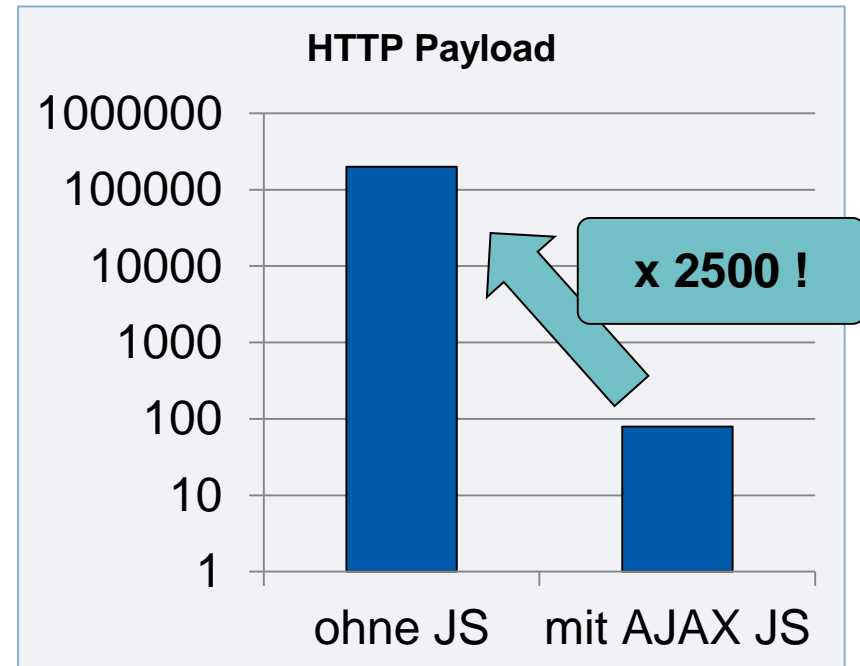
Ohne JavaScript (LAMP):

- **Gesamte Twitterseite**
>200.000 Zeichen
- >195 kB



Mit JavaScript:

- @ID:
- Timestamp:
- Nachricht:
- = Byte



Agenda

- Wiederholung
- Ergebnisse der Umfragen

Client-Server-Architektur

- C-S-Aufbau
- Client: Kommunikation
- LAMP und MEAN
- Server: Architekturkomponenten
- Auswahl-Kriterien für Komponenten
- Das Nadelöhr

■ Node.js: Mein erster Server mit JavaScript

- Zusammenfassende Fragen
- Trends
- Ausblick

node



TM

node.js

- **serverseitige Plattform**
- **JavaScript: Basiert auf Googles V8 JavaScript Engine**
- **Skalierbare Netzwerkanwendungen**
- **Event-gesteuert**
- **Große Anzahl an parallelen Verbindungen (kein Thread Switch sondern blockierungsfrei über Callbacks in einem Thread)**

node.js - Installation

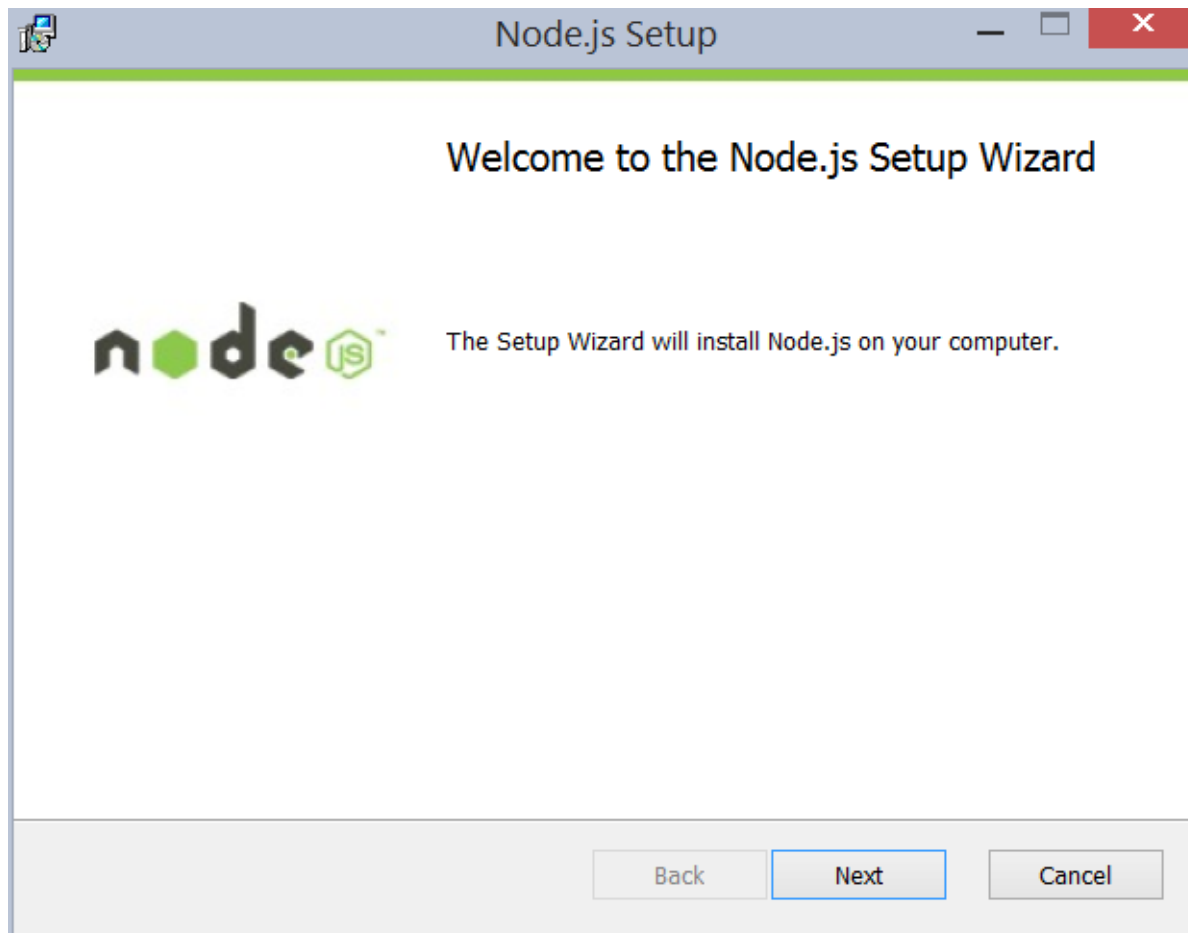
- Windows

- <http://nodejs.org/>



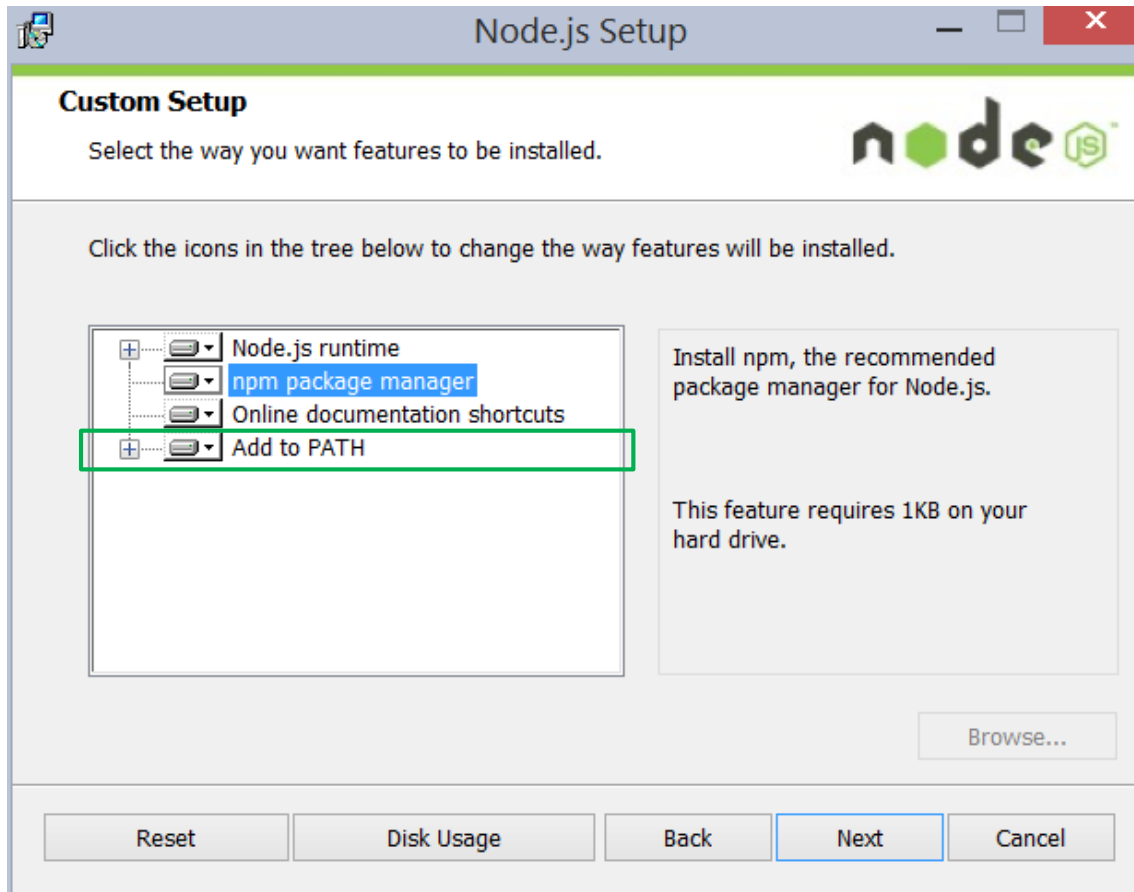
node.js - Installation

- Windows



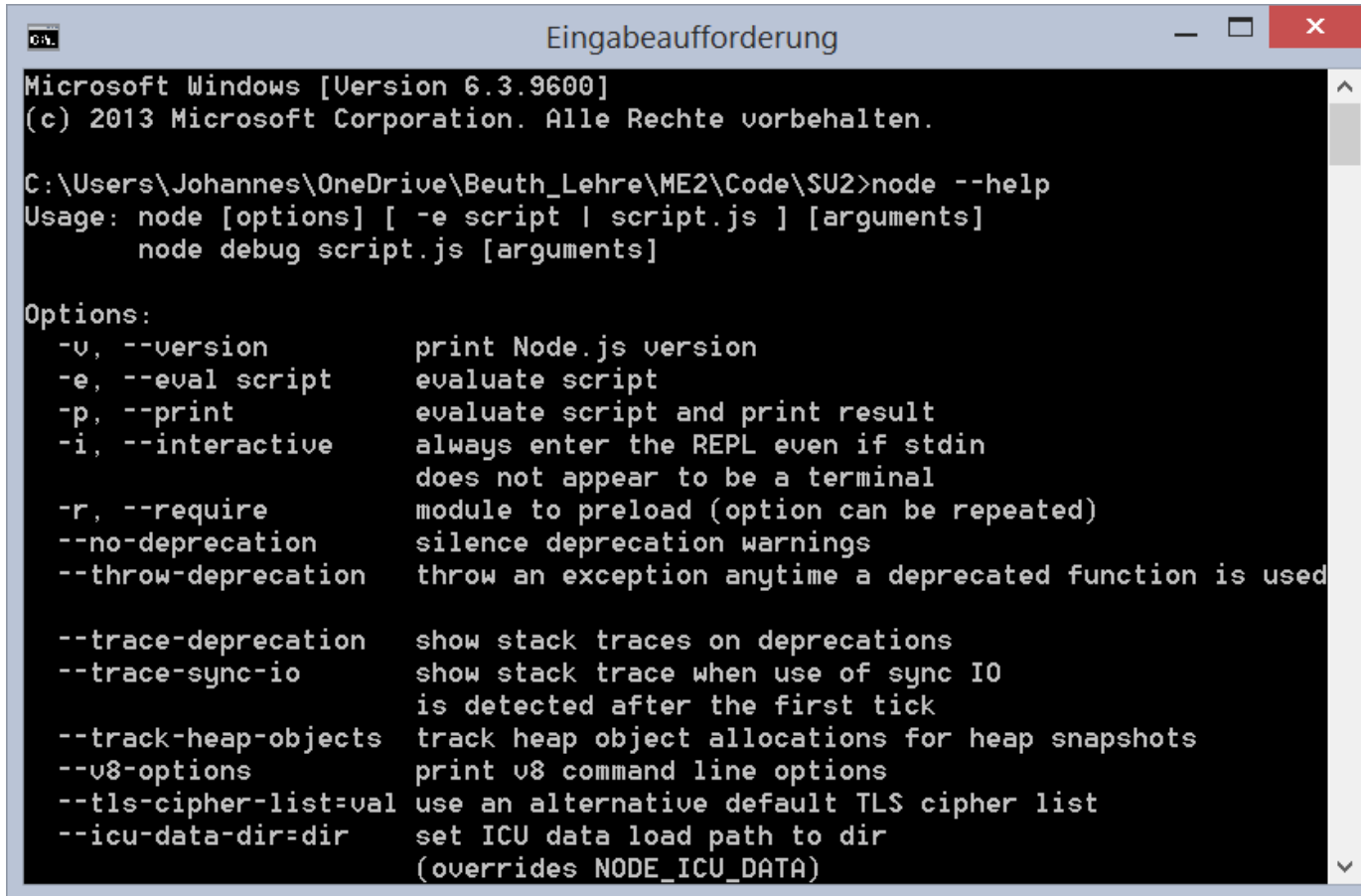
node.js - Installation

- Automatische PATH-Verknüpfung
- npm – Node.js Package Manager → behandelt Abhängigkeiten



node.js - Installation

- node und npm nun in der Eingabeaufforderung verfügbar



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Johannes\OneDrive\Beuth_Lehre\ME2\Code\SU2>node --help
Usage: node [options] [ -e script | script.js ] [arguments]
       node debug script.js [arguments]

Options:
  -v, --version           print Node.js version
  -e, --eval script       evaluate script
  -p, --print             evaluate script and print result
  -i, --interactive       always enter the REPL even if stdin
                        does not appear to be a terminal
  -r, --require module    module to preload (option can be repeated)
  --no-deprecation        silence deprecation warnings
  --throw-deprecation      throw an exception anytime a deprecated function is used
  --trace-deprecation      show stack traces on deprecations
  --trace-sync-io          show stack trace when use of sync I/O
                        is detected after the first tick
  --track-heap-objects     track heap object allocations for heap snapshots
  --v8-options            print v8 command line options
  --tls-cipher-list=val    use an alternative default TLS cipher list
  --icu-data-dir=dir       set ICU data load path to dir
                        (overrides NODE_ICU_DATA)
```

node.js – Installation

■ Linux

- `curl --silent --location
https://deb.nodesource.com/setup_0.12 | sudo bash
-`

- `apt-get install nodejs`

(opt.) Symlink anlegen für node = nodejs

- `sudo ln -s /usr/bin/nodejs /usr/bin/node`

node.js – Installation

■ node und npm nun im Terminal verfügbar

```
root@vagrant-ubuntu-trusty-64:/home/vagrant# node --help
Usage: node [options] [ -e script | script.js ] [arguments]
       node debug script.js [arguments]

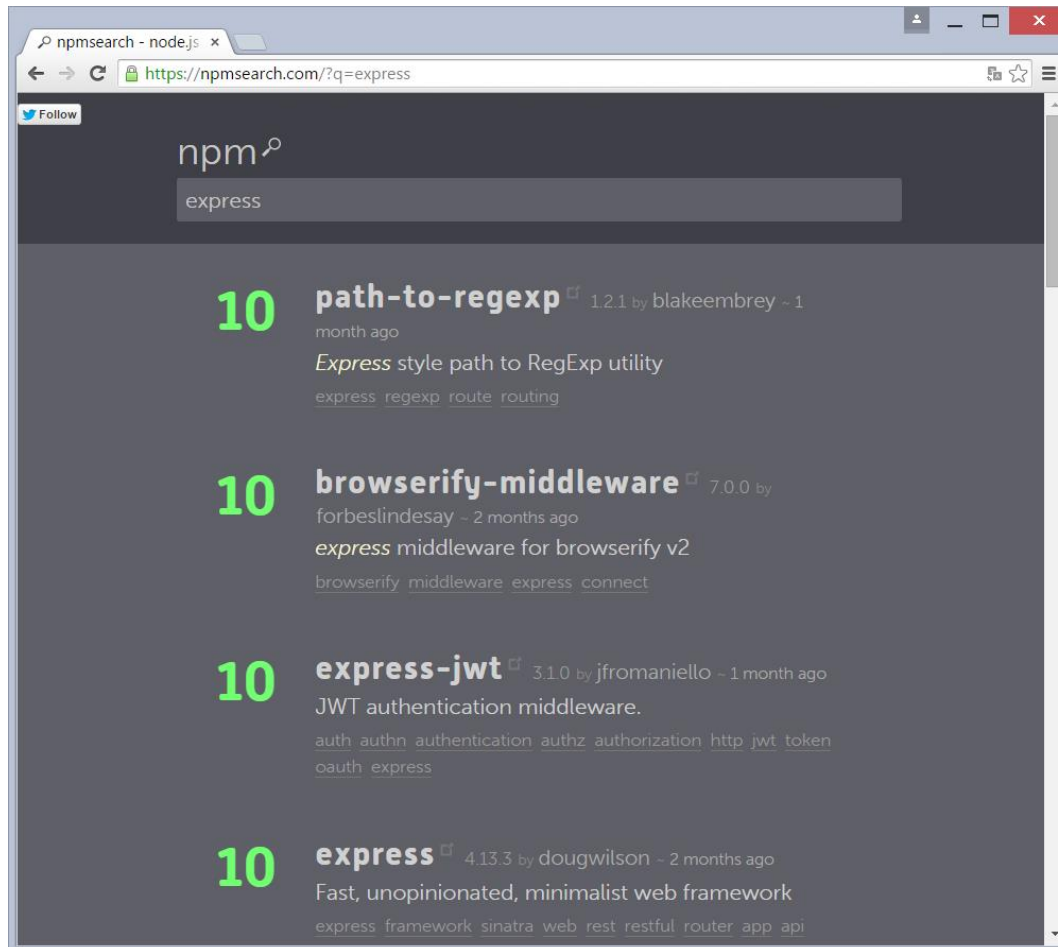
Options:
  -v, --version           print node's version
  -e, --eval script       evaluate script
  -p, --print             evaluate script and print result
  -i, --interactive       always enter the REPL even if stdin
                          does not appear to be a terminal
  --no-deprecation        silence deprecation warnings
  --trace-deprecation     show stack traces on deprecations
  --v8-options            print v8 command line options
  --max-stack-size=val   set max v8 stack size (bytes)

Environment variables:
NODE_PATH                ':'-separated list of directories
                          prefixed to the module search path.
NODE_MODULE_CONTEXTS     Set to 1 to load modules in their own
                          global contexts.
NODE_DISABLE_COLORS      Set to 1 to disable colors in the REPL

Documentation can be found at http://nodejs.org/
root@vagrant-ubuntu-trusty-64:/home/vagrant#
```

Node Package Manager nutzen

- **Umfangreichste Paket-Bibliothek der ganzen OpenSource-Gemeinde!**



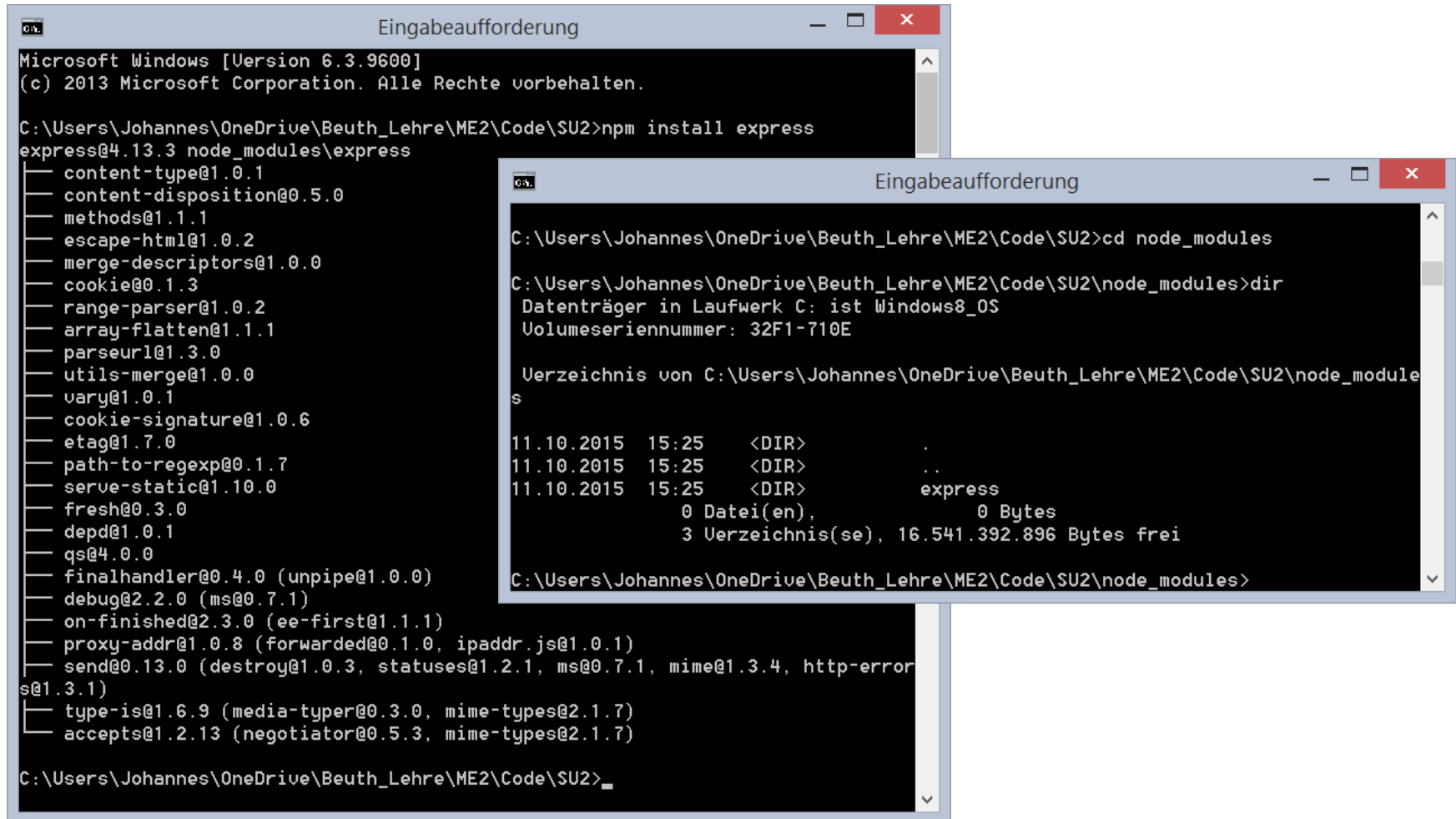
Node Package Manager nutzen

- **Express installieren**
 - **Express ist eine Erweiterung mit vielen nützlichen Funktionen um später Server zu entwickeln**

```
npm install express
```

Node Package Manager nutzen

- Express installiert im lokalen Ordner node_modules



```
Eingabeaufforderung
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Johannes\OneDrive\Beuth_Lehre\ME2\Code\SU2>npm install express
express@4.13.3 node_modules\express
├── content-type@1.0.1
├── content-disposition@0.5.0
├── methods@1.1.1
├── escape-html@1.0.2
├── merge-descriptors@1.0.0
├── cookie@0.1.3
├── range-parser@1.0.2
├── array-flatten@1.1.1
├── parseurl@1.3.0
├── utils-merge@1.0.0
├── vary@1.0.1
├── cookie-signature@1.0.6
├── etag@1.7.0
├── path-to-regexp@0.1.7
├── serve-static@1.10.0
├── fresh@0.3.0
├── depd@1.0.1
├── qs@4.0.0
├── finalhandler@0.4.0 (unpipe@1.0.0)
├── debug@2.2.0 (ms@0.7.1)
├── on-finished@2.3.0 (ee-first@1.1.1)
├── proxy-addr@1.0.8 (forwarded@0.1.0, ipaddr.js@1.0.1)
├── send@0.13.0 (destroy@1.0.3, statuses@1.2.1, ms@0.7.1, mime@1.3.4, http-errors@1.3.1)
├── type-is@1.6.9 (media-typer@0.3.0, mime-types@2.1.7)
└── accepts@1.2.13 (negotiator@0.5.3, mime-types@2.1.7)

C:\Users\Johannes\OneDrive\Beuth_Lehre\ME2\Code\SU2>

Eingabeaufforderung
C:\Users\Johannes\OneDrive\Beuth_Lehre\ME2\Code\SU2>cd node_modules

C:\Users\Johannes\OneDrive\Beuth_Lehre\ME2\Code\SU2\node_modules>dir
Datenträger in Laufwerk C: ist Windows8_OS
Volumennummer: 32F1-710E

Verzeichnis von C:\Users\Johannes\OneDrive\Beuth_Lehre\ME2\Code\SU2\node_modules
.
..
express
0 Datei(en), 0 Bytes
3 Verzeichnis(se), 16.541.392.896 Bytes frei

C:\Users\Johannes\OneDrive\Beuth_Lehre\ME2\Code\SU2\node_modules>
```

Node Package Manager

- **npm verwaltet ~alle~ Pakete und Abhängigkeiten**
- **ist serverseitig !**
- für Webprojekte mehrfach `npm install <xy>` ist lästig
 - daher kann man eine `package.json` Datei anlegen, die `npm` dann nutzt und verwaltet.
- Anlegen eines neuen Projektes/neue `package.json` mit
 - `npm init` im neuen Verzeichnis des Projektes

Node Package Manager: package.json



```
npm
About to write to C:\Users\Johannes\OneDrive\Beuth_Lehre\ME2\Code\SU2\helloworld\package.json:
{
  "name": "helloworld",
  "version": "1.0.0",
  "description": "A small Hello World example with express",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Johannes Konert",
  "license": "MIT",
  "dependencies": {
    "express": "^4.13.3"
  },
  "devDependencies": {}
}
Is this ok? (yes)
```

- Automatisches Aktualisieren der `package.json` bei Installation weiterer Pakete: `npm install <xy> --save`

node.js – mein erstes WebProjekt

- **Aufgabe:** Geben Sie bei Abruf der Route „/“ etwas HTML zurück mit Hello World....

- **Lösung:** app.js

```
var express = require('express');  
var app = express();
```

```
// add and configure Route /  
app.get('/', function (req, res) {
```

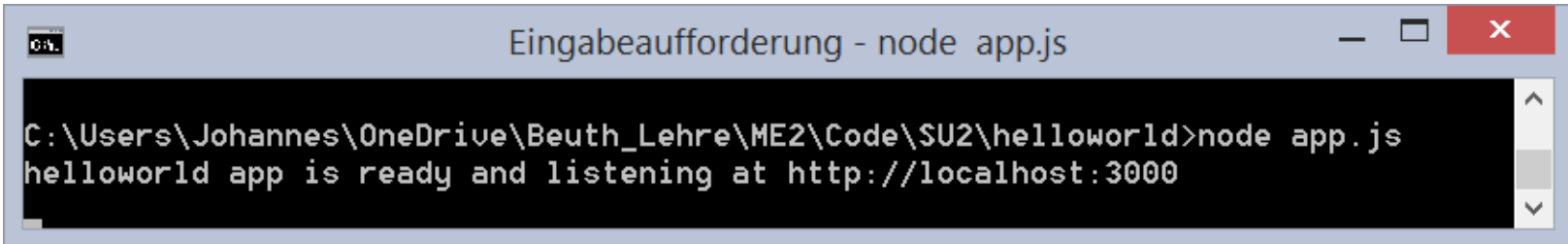
```
    res.send('<!DOCTYPE html>' +  
            '<html lang="en">' +  
            '<head><meta charset="utf-8"></head>' +  
            '<body><h1>Hello World!</h1></body>' +  
            '</html>'
```

```
    );  
});
```

```
var server = app.listen(3000, function () {  
    console.log('helloworld app is ready and listening at http://localhost:3000');  
});
```

node.js – mein erstes WebProjekt

- **Aufgabe:** Geben Sie bei Abruf der Route „/“ etwas HTML zurück mit Hello World....
- **Lösung:** app.js
- **Node-Server starten (bspw. in Konsole oder via IDE)**



```
C:\Users\Johannes\OneDrive\Beuth_Lehre\ME2\Code\SU2\helloworld>node app.js
helloworld app is ready and listening at http://localhost:3000
```


node.js – mein erstes WebProjekt

- **Aufgabe:** Geben Sie bei Abruf der Route „/“ etwas HTML zurück mit Hello World....
- **Lösung:** app.js
- **Node-Server starten** (bspw. in Konsole oder via IDE)



node.js – Serverseitiges JavaScript und seine Besonderheiten

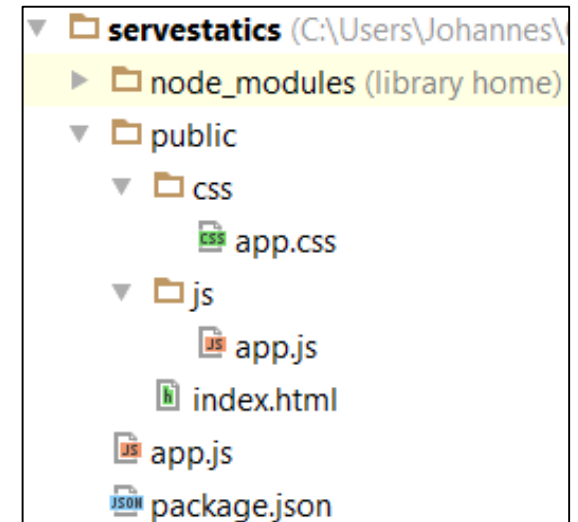
- Statische Dateien ausliefern, *.js, *.css, *.html, *.png usw.

```
"use strict";  
var express = require('express');  
var path = require('path');  
  
var app = express();
```

```
// add route to static files  
app.use(express.static(path.join(__dirname, 'public')));
```

```
// start server
```

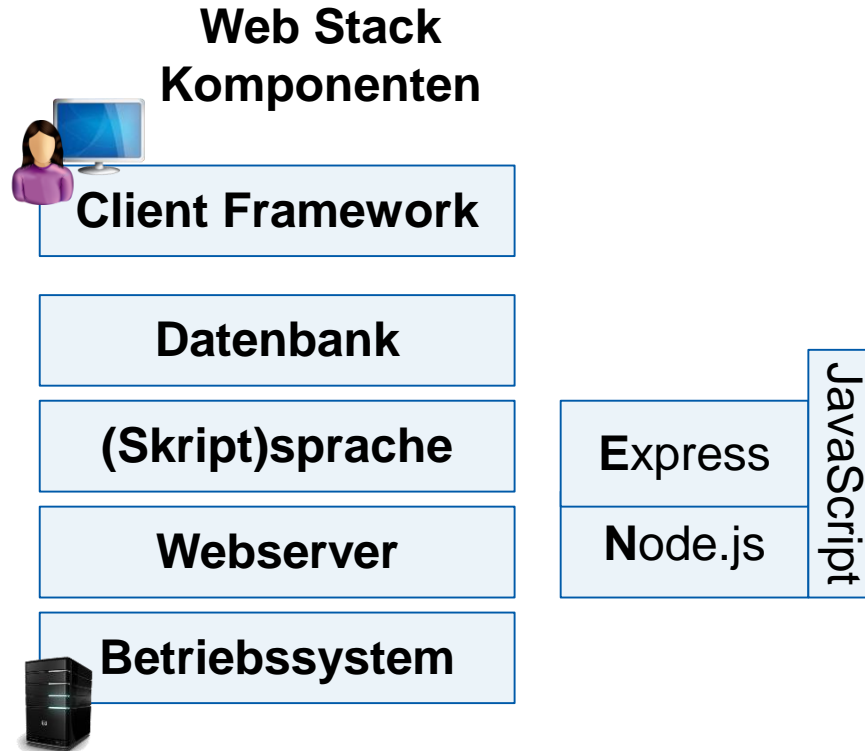
```
var server = app.listen(3000);
```





node.js

- **Wo ist der Webserver**



- **node.js ist Webserver und Laufzeitumgebung für die Skriptsprache in einem!** → weniger Komponenten, mehr Geschwindigkeit

Agenda

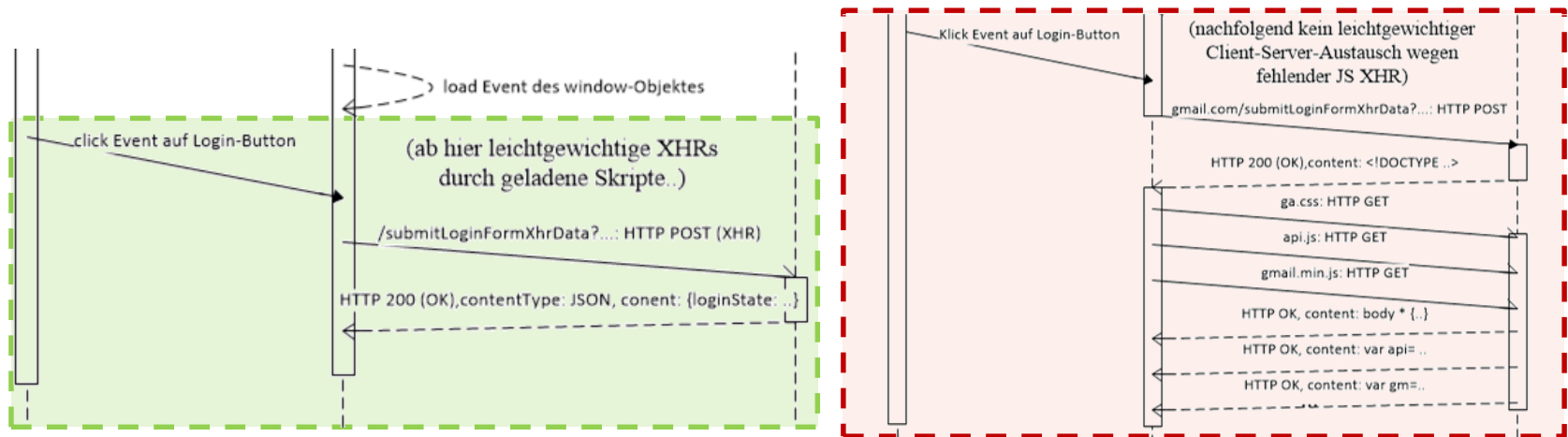
- Wiederholung
- Ergebnisse der Umfragen

Client-Server-Architektur

- C-S-Aufbau
- Client: Kommunikation
- LAMP und MEAN
- Server: Architekturkomponenten
- Auswahl-Kriterien für Komponenten
- Das Nadelöhr
- Node.js: Mein erster Server mit JavaScript
- Zusammenfassende Fragen (3x)
- Trends
- Ausblick

Rückblick / Wiederholung

- Was sind die wesentlichen Unterschiede beim Laden der Webseite zwischen Thin-Client und Rich-Client Prinzip?
 - Klausur: ggf. „Beschriften Sie ein Sequenzdiagramm dazu“

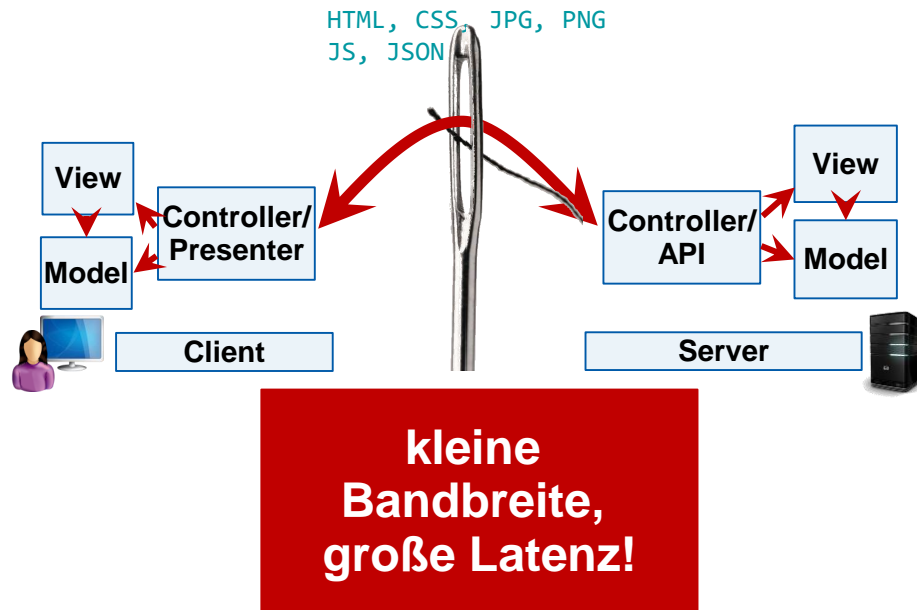


Rückblick / Wiederholung

- **Warum ist für eine SPA auf Server-Seite eine Verwendung verschiedener Komponenten für Webserver und (Skript)-Sprache nachteilig?**
- Auch das initial ausgelieferte Dokument ggf. schon via Skriptsprache dynamisch erstellt wird
- Laufzeitumgebung des Webserver != Laufzeitumgebung der Skriptsprache → mehr Speicherverbrauch und langsamere Reaktion
- Nach dem initialen Laden des Dokumentes (DOM)
 - überwiegend dynamische Inhalte via XHR geladen werden
 - JSON via API (idealerweise REST-API)
 - kleine Templatebausteine

Rückblick / Wiederholung

■ Was ist das Client-Server „Nadelöhr“ – Problem?



Agenda

- Wiederholung
- Ergebnisse der Umfragen

Client-Server-Architektur

- C-S-Aufbau
- Client: Kommunikation
- LAMP und MEAN
- Server: Architekturkomponenten
- Auswahl-Kriterien für Komponenten
- Das Nadelöhr
- Node.js: Mein erster Server mit JavaScript
- Zusammenfassende Fragen (3x)
- Trends
- Ausblick

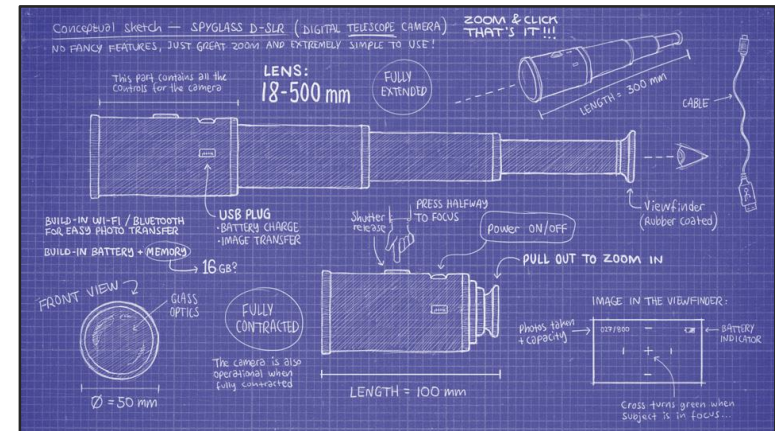
Trends:

Aufkommende Webtechnologien für den Web Stack von morgen

- Facebooks Hack Sprache (Open Source, statische Typisierung, Compiler und Geschwindigkeit für PHP)



- Isomorphes JavaScript (dank node.js) für schnelle Clients, bessere SEO



- JavaScript wird zum neuen “Machine Code”
→ Andere Sprachen werden nach JavaScript kompiliert

- Microsoft TypeScript
(Open Source, fügt statische Typisierung und Klassen-basierte Objektnotation hinzu, JS abwärtskompatibel)



- Google Dart
(Java-ähnliche Sprache, auch statische Typisierung, Kompiliert nach JS, Für Datenaustausch mit JS gibt es Libs)



Interesse?: siehe auch <https://www.paypal-engineering.com/2015/04/27/isomorphic-react-apps-with-react-engine/>
oder <http://nerds.airbnb.com/isomorphic-javascript-future-web-apps/>

Organisatorisches / Einschub

- Die Übungsbelegung hat leider nicht ausgereicht.
- Mit sofortiger Wirkung ist die Übung 3 (2a) um 8 Uhr gestrichen.

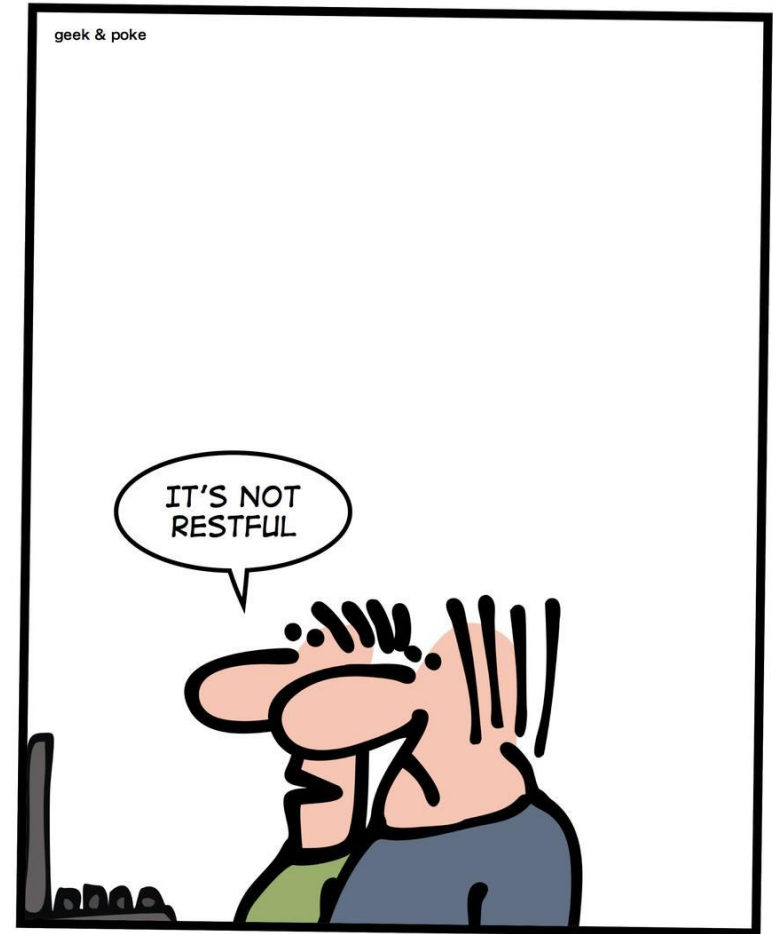
Übung	Uhrzeit (am Do)	SOLL	Status, Fr 15.04.	NEU
3 (2a)	8:00 Uhr	22	7	0
4 (2b)	10:00 Uhr	22	17	21
1 (1a)	12:15 Uhr	22	8	10
2 (1b)	14:15 Uhr	22	13	13

Ausblick auf nächstes Mal

- API Design
- RESTful

**Vielen Dank
und bis
zum nächsten Mal**

HOW TO INSULT A DEVELOPER





Apache



XAMPP

- Apache – Windows/Mac
 - XAMPP
 - <http://www.apachefriends.org/de/xampp.html>
- Inhalt:
 - Apache
 - MySQL
 - PHP
 - phpMyAdmin
 - FileZilla FTP Server
 - Tomcat (with mod_proxy_ajp as connector)
 - Strawberry Perl Portable
 - XAMPP Control Panel

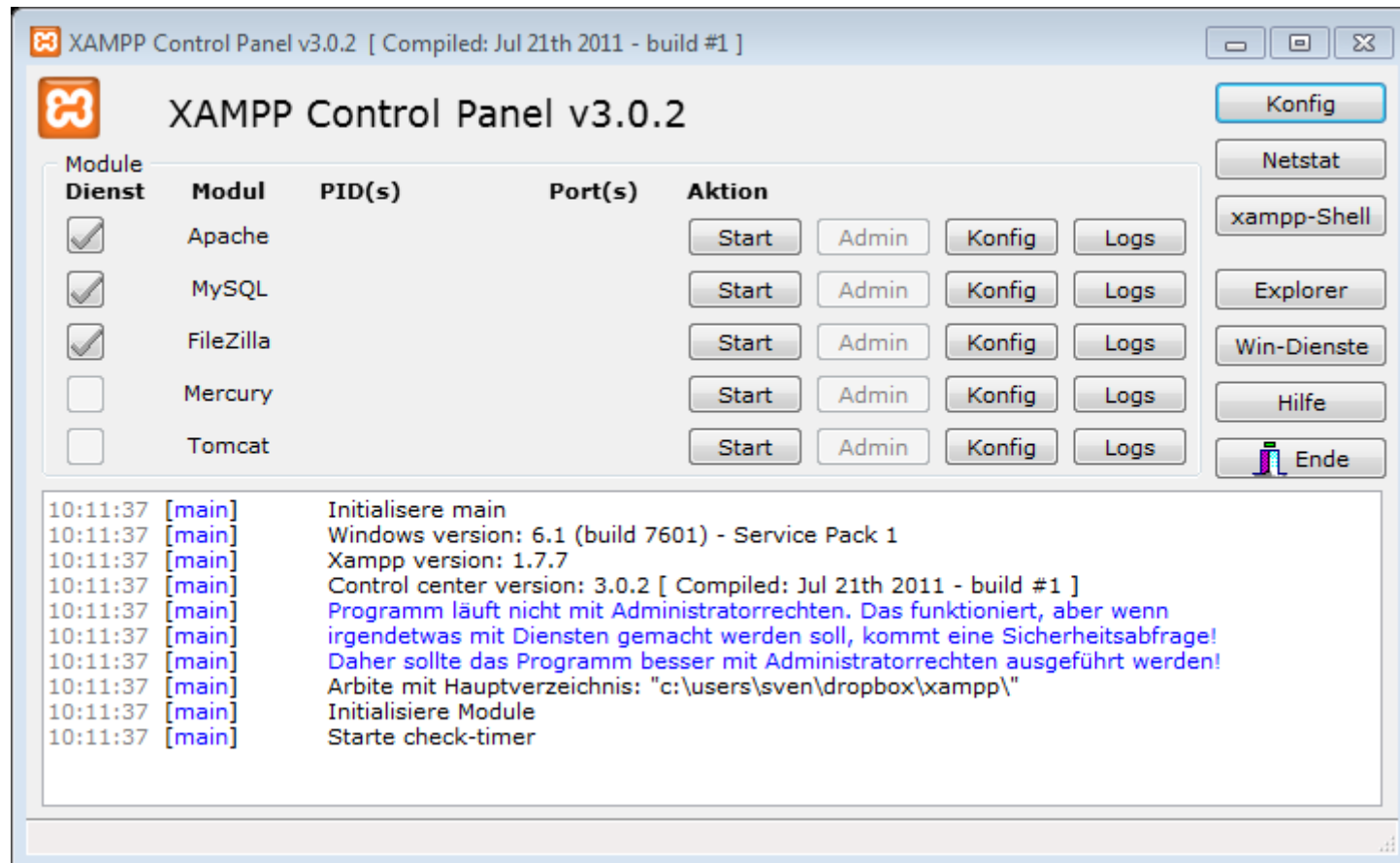
Installation XAMPP

- Installation über Installer



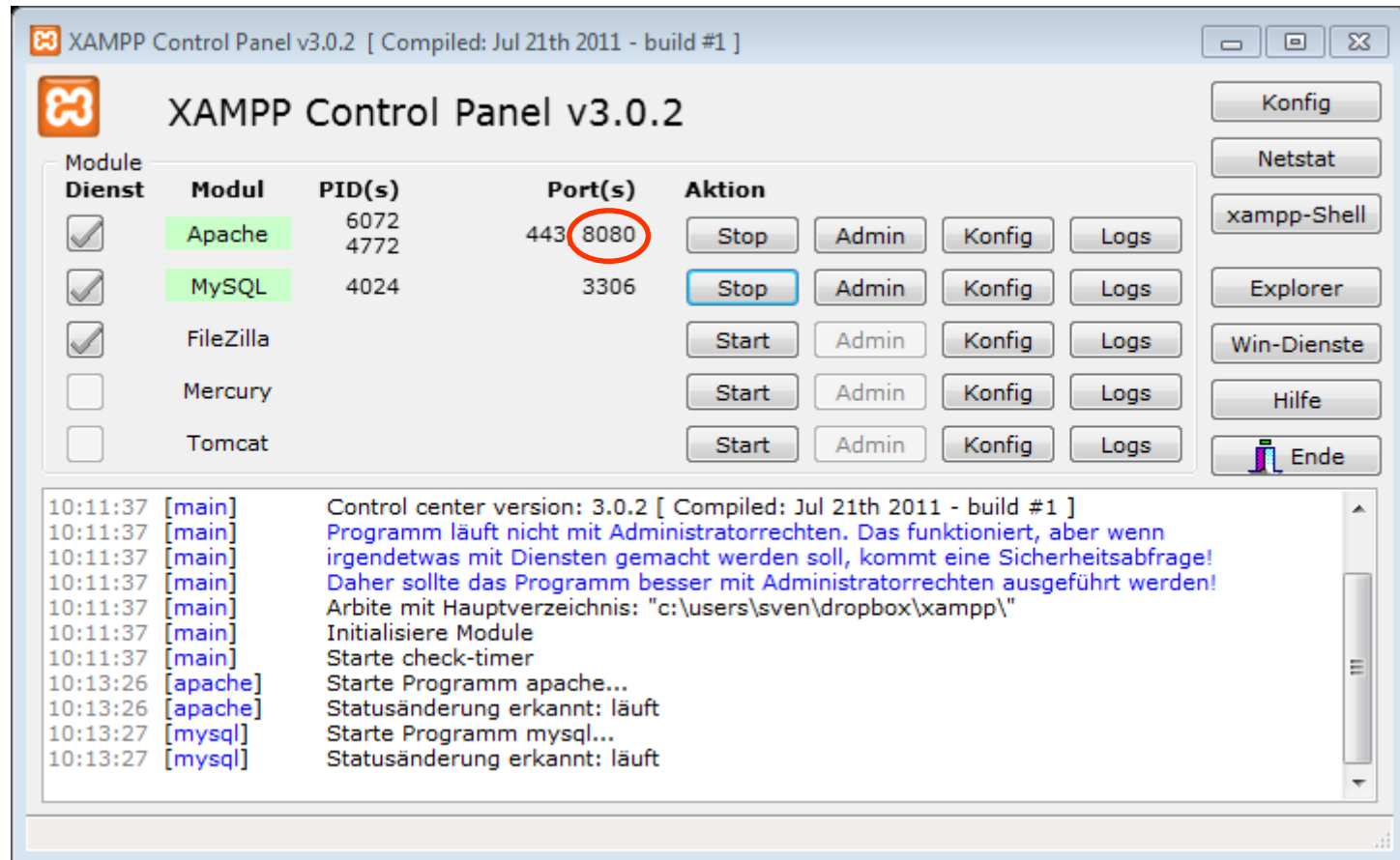
Installation XAMPP

- Komfortables Panel zum Verwalten der Services



Installation XAMPP

- Komfortables Panel zum Verwalten der Services



Installation XAMPP

- Infoseite von Xampp - <http://localhost:8080>



The screenshot shows the XAMPP for Windows installation page. At the top, there is a logo and the text "XAMPP für Windows". To the right of the logo, there are links for different languages: English / Deutsch / Français / Nederlands / Polski / Italiano / Norwegian / Español / 中文 / Português (Brasil) / 日本語. Below the header, there is a sidebar on the left with a list of links: XAMPP 1.7.7 [PHP: 5.3.8], Willkommen (highlighted), Status, Sicherheitscheck, Dokumentation, Komponenten, PHP (with sub-links: phpinfo(), CD-Verwaltung, Biorhythmus, Instant Art, Telefonbuch), Perl (with sub-links: perlinfo(), Gästebuch), J2EE (with sub-links: Status, Tomcat examples), and Tools (with sub-links: phpMyAdmin, Webalizer, Mercury Mail, FileZilla FTP). The main content area on the right has a yellow background and contains the following text: "Willkommen zu XAMPP für Windows!", "Herzlichen Glückwunsch: XAMPP ist erfolgreich auf diesem Rechner installiert!", a paragraph explaining the next steps, a note about OpenSSL support, and a signature.

XAMPP 1.7.7
[PHP: 5.3.8]

Willkommen
Status
Sicherheitscheck
Dokumentation
Komponenten

PHP
phpinfo()
CD-Verwaltung
Biorhythmus
Instant Art
Telefonbuch

Perl
perlinfo()
Gästebuch

J2EE
Status
Tomcat examples

Tools
phpMyAdmin
Webalizer
Mercury Mail
FileZilla FTP

Willkommen zu XAMPP für Windows!

Herzlichen Glückwunsch:
XAMPP ist erfolgreich auf diesem Rechner installiert!

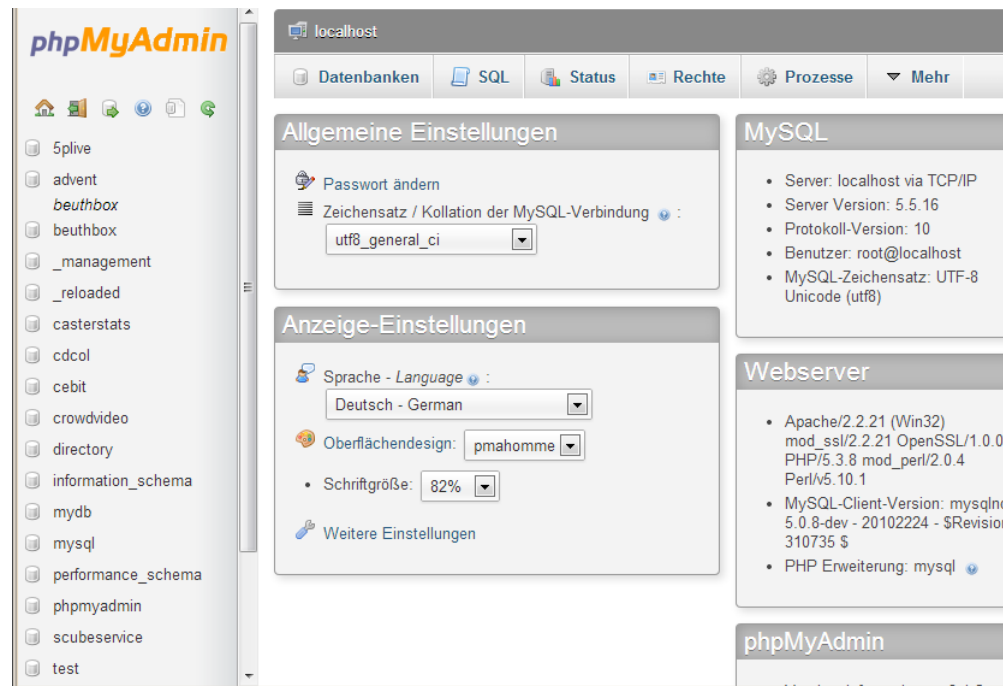
Nun kann es losgehen. :) Als erstes bitte einmal auf der linken Seite auf »Status« klicken. Damit bekommt man einen Überblick was alles schon funktioniert. Ein paar Funktionen werden ausgeschaltet sein. Das ist Absicht so. Es sind Funktionen, die nicht überall funktionieren oder evtl. Probleme bereiten könnten.

Für die OpenSSL Unterstützung benutzt bitte das Testzertifikat mit der URL <https://127.0.0.1> bzw. <https://localhost>

Viel Spaß, Kay Vogelgesang + Kai 'Oswald' Seidler

Installation XAMPP

- MySQL mit PHPMyAdmin- <http://localhost:8080/phpmyadmin/>
 - Zugang: Nur root als Benutzername ohne Passwort

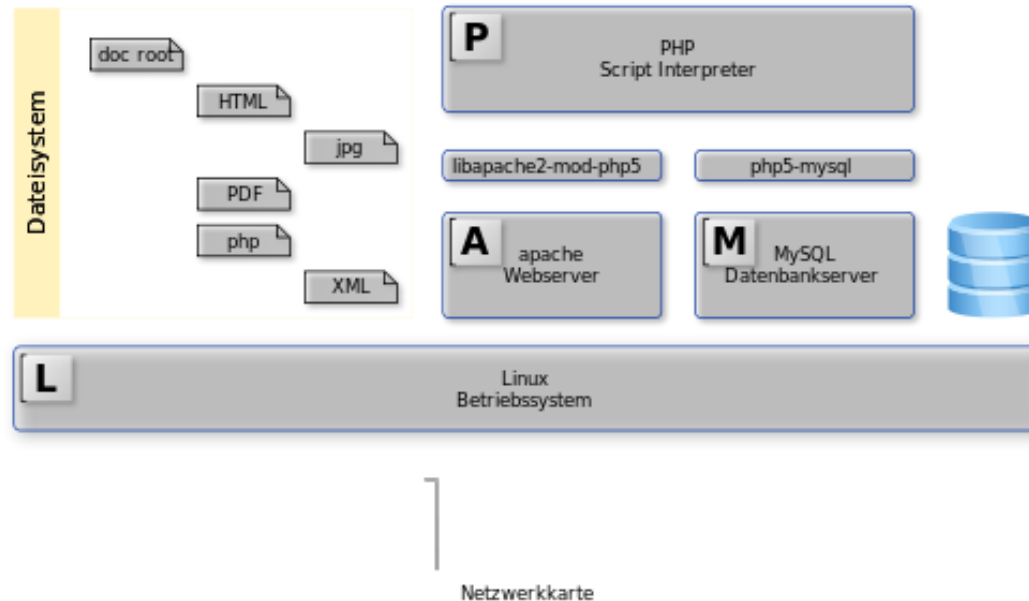


Installation XAMPP

- ACHTUNG! Xampp sollte nicht als Webserver für einen Produktivbetrieb genutzt werden!
 - Mangelnde Sicherheitseinstellungen
- Der MySQL-Administrator (root) hat kein Passwort.
- Der MySQL-Server ist übers Netzwerk erreichbar.
- phpMyAdmin ist übers Netzwerk erreichbar.
- Das XAMPP Verzeichnis ist nicht geschützt.
- Bekannte Beispiel-Benutzer bei FileZilla FTP und dem Mercury Mail Server.

Linux

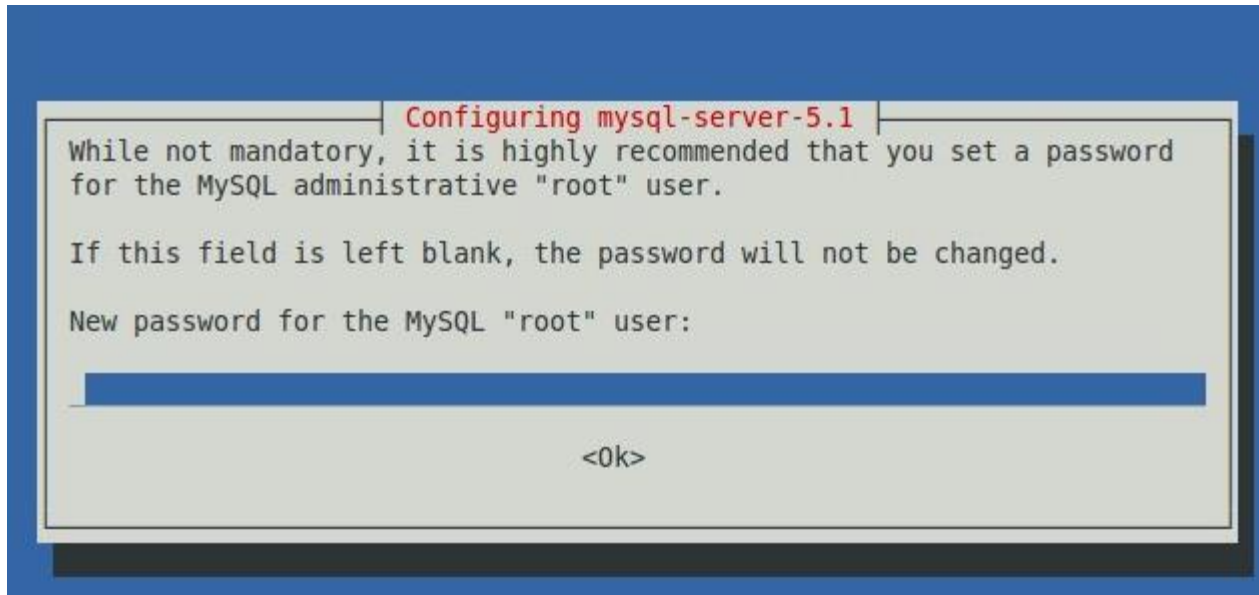
- LAMP - **L**inux, **A**pache, **M**ySQL and **P**HP



Quelle: Wikipedia

Installation LAMP

- LAMP - **L**inux, **A**pache, **M**ySQL and **P**HP
 - `sudo apt-get install lamp-server^`
 - ➔ Installiert alle nötigen Komponenten für eine Reibungslose Kommunikation zwischen Apache, MySQL und PHP

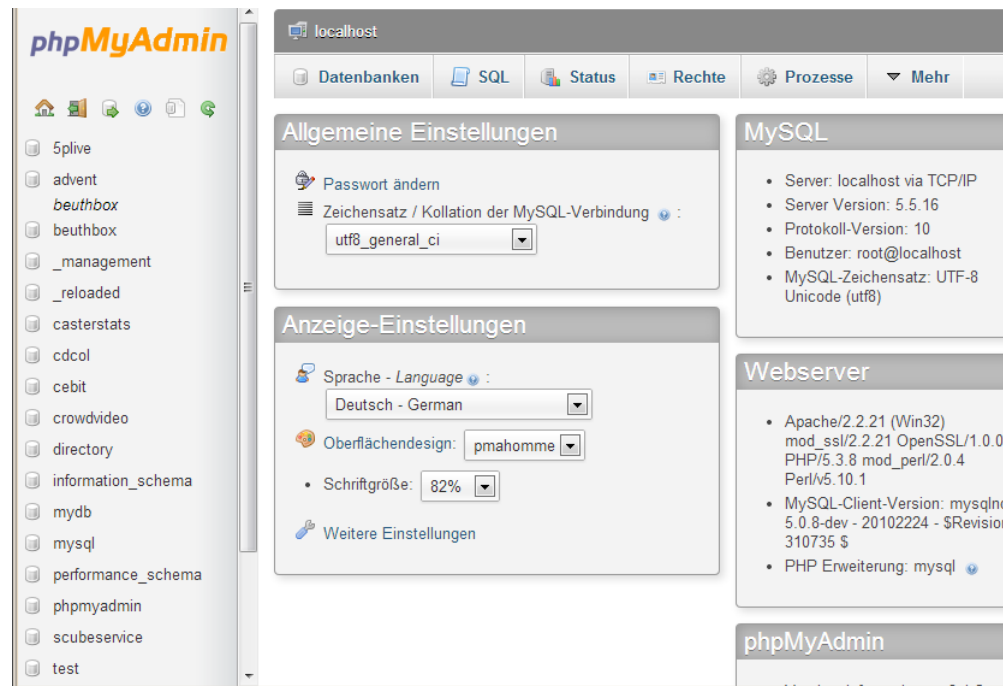


Installation LAMP

- LAMP - **L**inux, **A**pache, **M**ySQL and **P**HP
 - Web: /var/www
- PHPMyAdmin
 - `sudo apt-get install phpmyadmin`
 - Eingabe des Root-Passwords von MySQL während des Set-Ups
- Adressen:
 - <http://server>
 - <http://server/phpmyadmin>

Installation LAMP

- MySQL mit PHPMyAdmin- <http://localhost/phpmyadmin/>
 - Zugang: root und gewähltes Passwort



Apache Server steuern

```
# Allgemein sudo /etc/init.d/apache2  
{start|stop|restart|reload|force-reload}  
# Beispiel sudo /etc/init.d/apache2 restart
```

- `start` - startet den Webserver
- `stop` - stoppt den Server
- `restart` - startet den Server neu, bestehende Verbindungen auf den Server werden gekappt
- `reload` - lädt die Konfigurationsdateien neu, ohne dass Verbindungen getrennt werden
- `force-reload` - Lädt die Konfigurationsdateien neu, auch wenn dabei Verbindungen getrennt werden müssen
- `sudo update-rc.d -f apache2 remove` - Apache2 aus der Autostartliste entfernen
- `sudo update-rc.d apache2 defaults` - Apache2 wieder der Autostartliste hinzufügen