

3 Übung Web- und Multimedia-Engineering

Aufgabenstellung A3 und
thematische Einführung *Node.js & AJAX*

- *Überblick (Terminplan)*
- 1. Aufgabenstellung A3: Node.js und AJAX
- 2. Anwendung von serverseitigen Technologien:
 - a) Node.js
 - b) AJAX
- 3. Hilfreiches, Tipps und Links

Woche	Datum	Übungsthema /-inhalt	Folien	Materialien
KW 41	9./10.10.	keine Übung (erste Vorlesungswoche)		
KW 42	16./17.10.	Einführung Aufgabenstellung A1: HTML + CSS + JS 		<ul style="list-style-type: none"> ■ Screenshots ■ Logo ■ Daten (CSV)
KW 43	23./24.10.	Konsultation		
KW 44	30.10.	Mo. Konsultation, Di. 31.10. keine Übung (Reformationstag)		
KW 45	6./7.11.	Abgabe A1		
KW 45	6./7.11.	Einführung Aufgabenstellung A2: PHP + XML 		<ul style="list-style-type: none"> ■ Leer-Vorlage ■ Daten (CSV)
KW 46	13./14.11.	Lösung A1 und Konsultation		
KW 47	20./21.11.	Abgabe A2		
KW 47	20./21.11.	Einführung Aufgabenstellung A3: Webservices Node.js + AJAX		Materialien für A3
KW 48	27./28.11.	Lösung A2 und Konsultation		
KW 49	4./5.12.	Konsultation		
KW 50	11./12.12.	Abgabe A3		
KW 50	11./12.12.	Einführung Aufgabenstellung A4: Anwendungsbeispiel		Materialien für A4
KW 51	18./19.12.	Lösung A3 und Konsultation		
KW 52	25./26.12.	Jahreswechsel (keine Übung)		
KW 1	1./2.1.	Jahreswechsel (keine Übung)		
KW 2	8./9.1.	Konsultation		
KW 3	15./16.01.	Konsultation		
KW 4	22./23.01.	Abgabe A4		
KW 4	22./23.01.	Abschluss, Feedback und Fragen		
KW 5	29./30.01.	Lösung A4 und offene Fragen (nur nach Anmeldung)		
Woche	Datum	Übungsthema /-inhalt	Folien	Materialien

https://imld.de/study/teaching/ws_17-18/wme_17-18/ueb-wme_17-18/

Terminplan / Ablauf

- **A1:** Grundlagen client-seitige Technologien: HTML5, CSS3 & JS
 - Grundgerüst für eine Website als Interface für world_data
- **A2: Grundlagen server-seitige Technologien: XML und PHP**
 - CSV sowie XML Transformation, Grundlagen PHP Serverkomponente
- **A3:** Erweiterung server-seitige Technologien: Node.js & AJAX
 - Erstellung eines REST-Services, Abfragen durch den Client via AJAX
- **A4:** Anwendungsfall – Visualisierung mit D3.js & Leaflet
 - Visualisierung von Daten mittels interaktiver Bar Charts und Karten



Teil 1

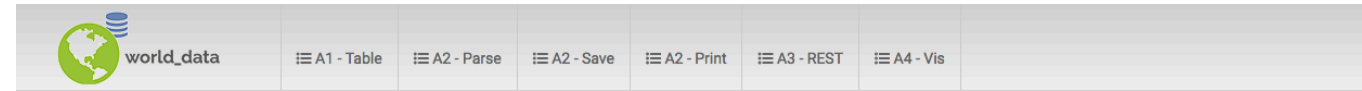
Aufgabenstellung A3: Node.js und AJAX

- Basis ist *Leer-Vorlage* (HTML & JS) *siehe* Materialien für A3
- Aufgabe umfasst insgesamt 3 Arbeitspakete

1 Serverseitiges Parsen einer CSV Datei & Speichern als JSON

2 Erstellen der REST-API

3 Abfragen der REST-API via AJAX



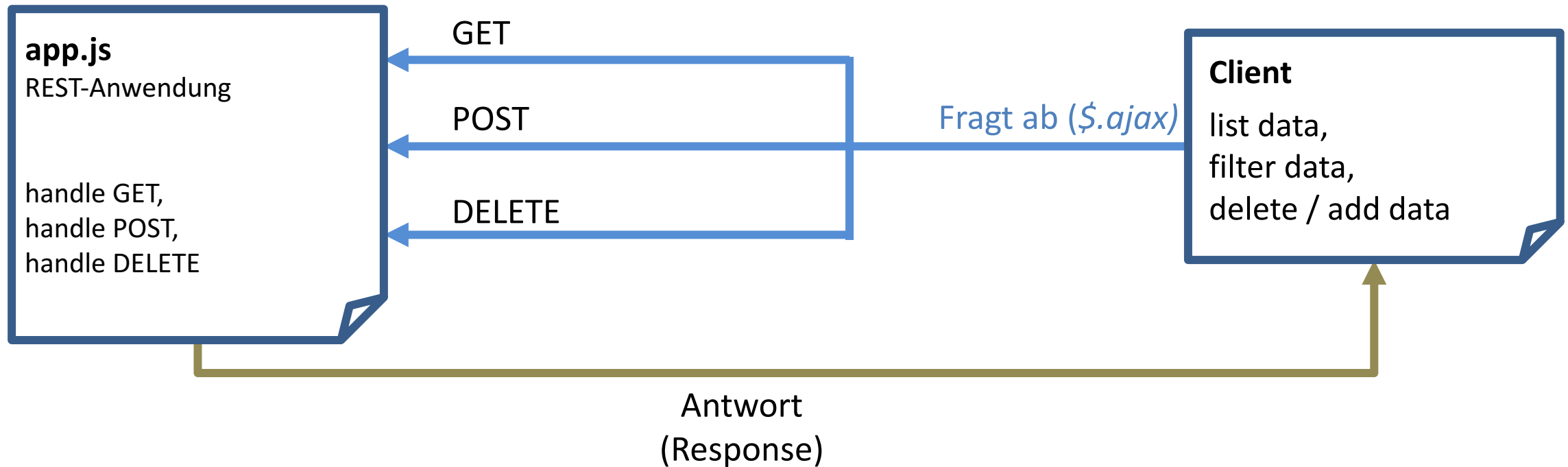
World Data Overview ...

ID	Country	birth rate / 1000	cellphones / 100	children / woman	electric usage	internet usage
----	---------	-------------------	------------------	------------------	----------------	----------------

Copyright © 2016 world_data
First course exercise HTML, CSS and JS of the lecture Web and Multimedia Engineering.

This solution has been created by:
FIXME

■ Schematische Aufbau der Anwendung



1. Serverseitiges Parsen einer CSV Datei & Speichern als JSON

- ☐ Bereitgestellte Datei `app.js` im Bereich „csvtojson“ erweitern
- ☐ Node.js Modul `csvtojson` für die Umwandlung nutzen
<https://www.npmjs.com/package/csvtojson>
- ☐ Globale Variable für das JSON-Objekt anlegen
- ☐ CSV laden, in JSON umwandeln und in der globalen Variable speichern, die Speicherung als Datei ist optional

```
[  
  {  
    "id": "001",  
    "name": "Brazil"  
    "birth_rate_per_1000": 16.405,  
    ...  
  },  
  {  
    "id": "002",  
    ...  
  }  
]
```


2. Erstellen einer REST-API

- ☐ GET Calls:
 - ☐ `/items` -> gibt alle Länder mit allen Properties zurück
 - ☐ `/items/id` -> gibt ein Land mit id und allen Properties zurück, wenn id nicht vorhanden: Status „No such id {id} in database.“
 - ☐ `/items/id1/id2` -> gibt alle Länder zwischen id1 und id2 mit allen Properties zurück, wenn Range nicht existiert: Status „Range not possible.“
 - ☐ `/properties` -> gibt alle Properties zurück (id, name, birth_rate_per ...)
 - ☐ `/properties/num` -> gibt Property mit der Nummer num zurück, wenn num nicht vorhanden: Status: „No such property available.“

2. Erstellen einer REST-API

☐ POST Calls

- ☐ `/items` -> konsumiert json-Objekt mit Property *name* sowie 2 beliebigen Properties, gibt Status: „Added country {name} to list!“ zurück

☐ DELETE Calls

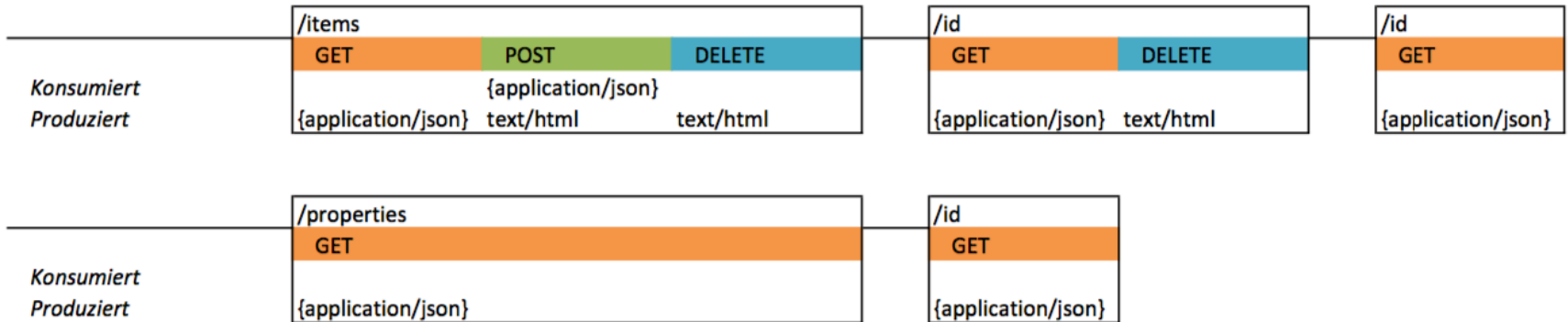
- ☐ `/items` -> löscht letztes Land aus der Liste, Status: „Deleted last country: {name}!“
- ☐ `/items/id` -> löscht Land mit der ID *id*, Status bei Erfolg: „Item {id} deleted successfully.“ oder bei Fehler: „No such id {id} in database“

☐ *Optional:*

- ☐ *Anzeigen der zurückgegebenen Statusmeldungen
(Fehler: roter Hintergrund, Erfolg: grüner Hintergrund jeweils mit Statustext)*

2. Erstellen der REST-API

Schnittstellen- beschreibung



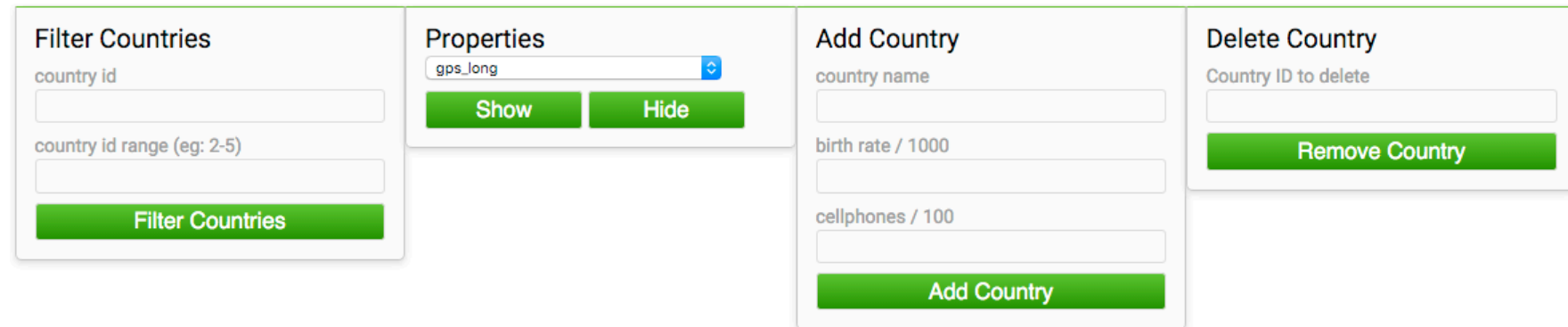
Element- beschreibung

item
int id
string name
string birth_rate_per_1000
string cell_phones_per_100
...

properties
string birth_rate_per_1000
string cell_phones_per_100
string children_per_woman
string electricity_consumption_per_capita
string gdp_per_capita
string gdp_per_capita_growth
string gps_lat
string gps_long
string id
string inflation_annual
string internet_user_per_100
string life_expectancy
string military_expenditure_percent_of_gdp
string name

3. Abfragen der REST-API durch den Client (AJAX)

- ☐ Endlich 😊 jQuery (Version => 3.1.1) darf verwendet werden
 - ☐ Kann auch in der Vorlage aktualisiert werden, vorhandene Version ergänzen/ersetzen
- ☐ Implementieren der AJAX-Abfragen und Verarbeitung für
 - ☐ „Filter Countries“ (Anfragen an GET Calls)
 - ☐ Wenn Range angegeben wurde -> Abfrage der Range (Range > Single id)
 - ☐ „Properties“ (Anfrage GET Calls)
 - ☐ „Add Country“ (Anfr. POST Calls)
 - ☐ „Delete Country“ (Anfrage DELETE Calls)
 - ☐ Wenn keine id angegeben wird -> letztes Land aus der Liste löschen
 - ☐ Wenn id angegeben wird -> Land mit id aus der Liste löschen



The image displays four distinct UI form panels arranged horizontally, each with a title and specific input fields and buttons.

- Filter Countries:** Contains two text input fields labeled 'country id' and 'country id range (eg: 2-5)', followed by a green 'Filter Countries' button.
- Properties:** Features a dropdown menu currently showing 'gps_long', and two green buttons labeled 'Show' and 'Hide'.
- Add Country:** Includes three text input fields labeled 'country name', 'birth rate / 1000', and 'cellphones / 100', with a green 'Add Country' button at the bottom.
- Delete Country:** Has a text input field labeled 'Country ID to delete' and a green 'Remove Country' button.

- Allgemeine Kriterien
 - ☐ Dateikodierung (Encoding):
UTF-8 und *Unix-LF (Zeilenende)*
 - ☐ Dokumentation Node.js und JavaScript Teil im Code
 - ☐ Ausfüllen von „name“ und „author“ in der package.json
 - ☐ Module: „express“, „csvtojson“ & „body-parser“
in package.json unter: „dependencies“ eintragen
(npm install NAME --save)
 - ☐ Beim Laden der Website (ohne Filter)
soll die gesamte Tabelle angezeigt werden
 - ☐ Leer-Vorlage (Material A3) benutzen, Dateistruktur ->

```
Team_XX
  app.js
  node_modules
  package.json
  public
    assets
      css
        font-awesome
        html5reset.css
        style.css
      js
        jquery-3.1.1.js
        ajax.js
      img
        world_data_logo.png
  index.html
  world_data.csv
```

- Erlaubte Hilfsmittel
 - jQuery
 - Node.js Module: express, csvtojson, body-parser 
- Testen: Node.js (LTS Version **>= 8.9.1 LTS**)
Firefox (aktuelle Version)
- Abgabe: Montag/Dienstag, **11.12.2017 & 12.12.2017** 

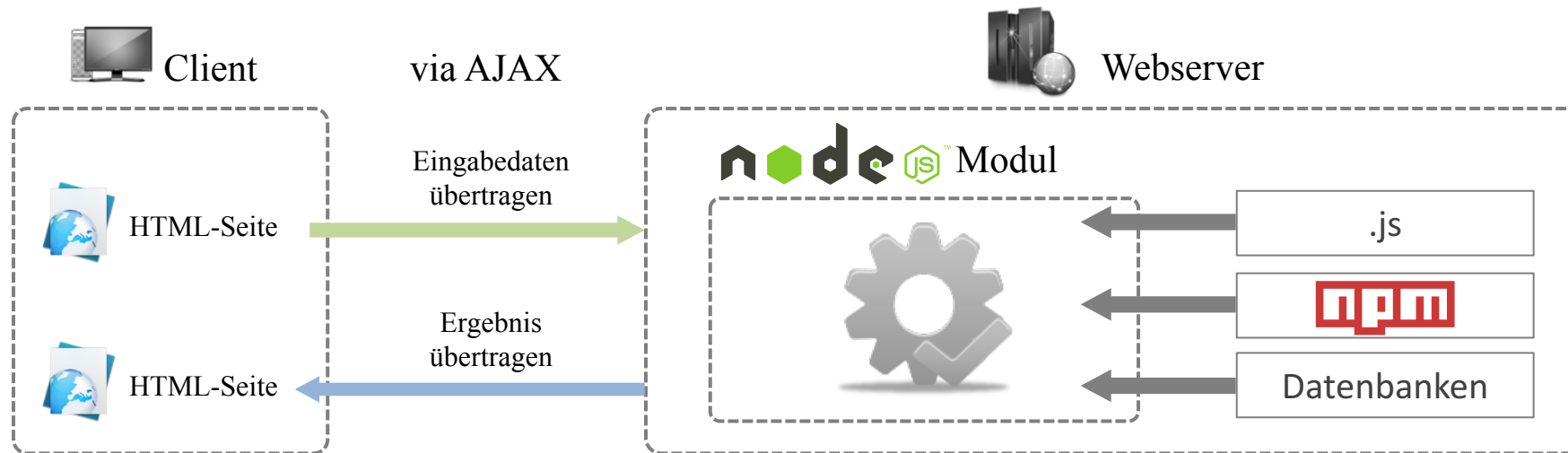
Teil 2

Einführung Node.js & AJAX

Teil 2a

Node.js

- Was ist Node.js ?!
 - Plattform zum serverseitigen Betrieb von Netzwerkanwendungen
 - Basiert auf JavaScript Laufzeitumgebung V8
 - Programmiersprache ist JavaScript
 - Funktionalität wird durch Module (node_modules) ermöglicht
 - Manager dieser Module: npm
 - Aktuell > 470000 Module



```
// Load the http module to create an http server.
var http = require('http');

// Configure our HTTP server to respond with Hello World to all requests.
var server = http.createServer(function (request, response) {
    response.writeHead(200, {"Content-Type": "text/plain"});
    response.end("Hello World\n");

});

// Listen on port 8000, IP defaults to 127.0.0.1
server.listen(8000);

// Put a friendly message on the terminal
console.log("Server running at http://127.0.0.1:8000/");
```

■ Starten der App:

- In der Konsole(Win, Linux, OSX): node helloworld.js
- App „theoretisch“ erreichbar via: <http://localhost:8000/>

Node.js – Einbinden von Modulen

- Installation von Modulen

`npm install module_name --save` -> Beispiel: `npm install express --save`

- Verwenden von Modulen

// In der Datei: app.js

`var modulename = require('modulname');`

Beispiel: `var express = require('express');`

- Ausführliche Einführung:

- Node.js Einführung -> Folie 26

- Serverseitiges Web Application Framework

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

- Middleware einbinden

```
var bodyParser = require('body-parser');  
  
app.use( bodyParser.json() );
```

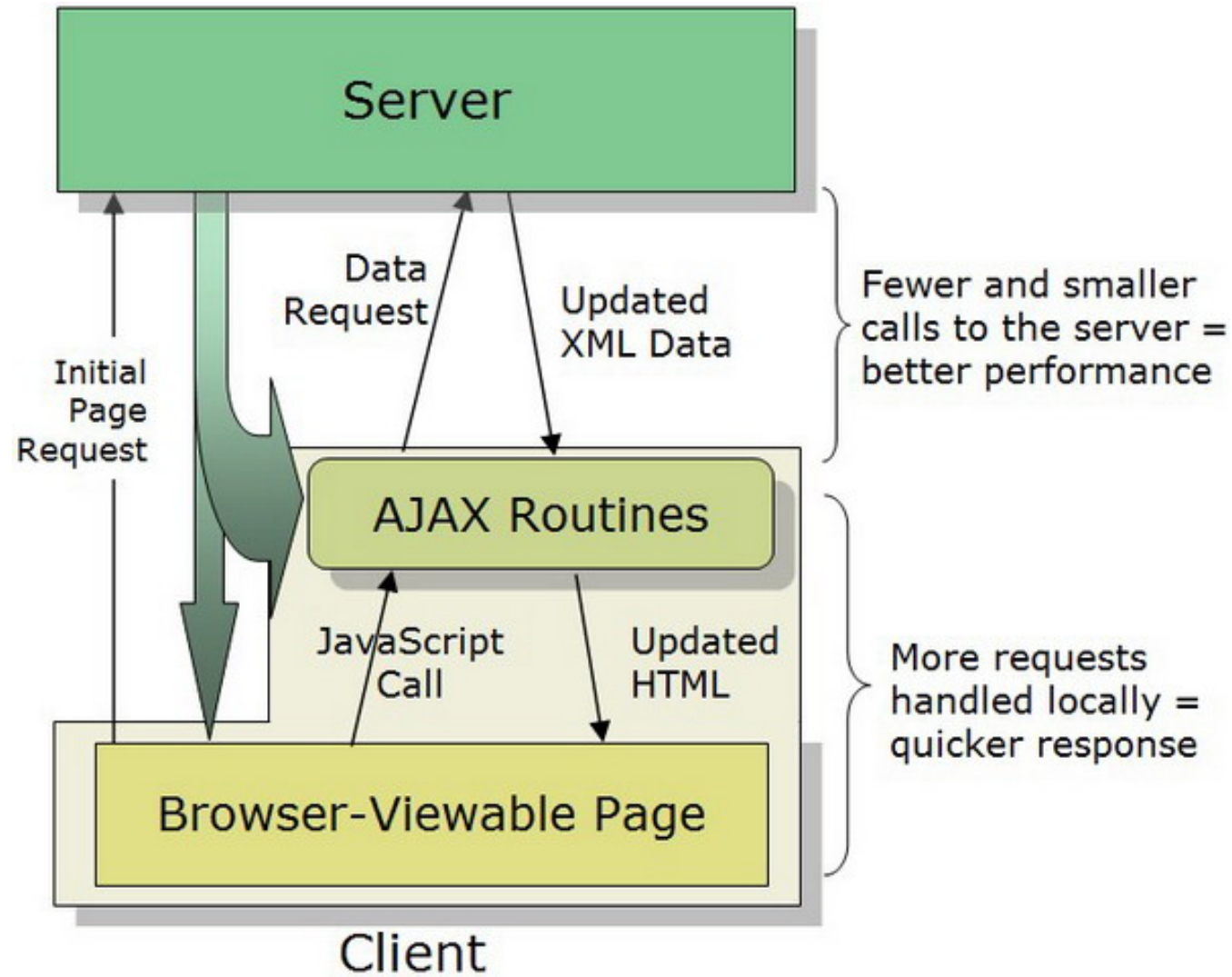
- (http) Request Handling

```
app.get('/data', function (req, res) {  
  var answer = doSomething(req.params);  
  res.send( answer );  
});
```

Teil 2b

AJAX – Asynchronous JavaScript and XML

AJAX – Asynchronous JavaScript and XML



```
$.ajax({  
    type: "GET, POST, DELETE, ...",  
    url: "http://localhost:3000/my_api",  
    async: true,  
    success: function(data) {  
        // Handle returned data  
    }, error: function(jqXHR, text, err) {  
        // Handle error if occurred  
    }  
});
```

- type – anzuwendende HTTP-Methode
- url – anzusprechende URL, die auf die Methode reagiert

■ JavaScript Object Notation

- Unabhängiges Format zum Beschreiben und Austauschen von Informationen
- Benutzt JavaScript Object-Syntax
- Kürzer/Kompakter als XML
- Erlaubt Arrays

JSON

```
{ "employees": [  
  {  
    "firstName": "John",  
    "lastName": "Doe"  
  },  
  {  
    "firstName": "Anna",  
    "lastName": "Smith"  
  },  
  {  
    "firstName": "Peter",  
    "lastName": "Jones"  
  }  
]
```

XML

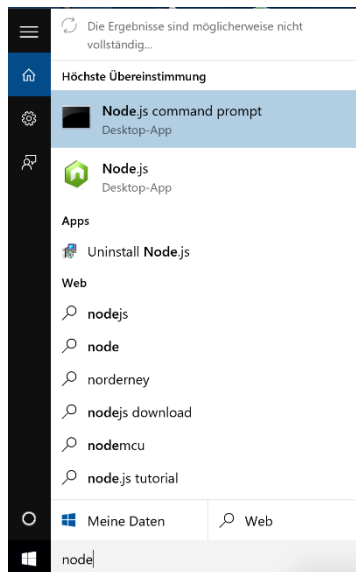
```
<employees>  
  <employee>  
    <firstName>John</firstName>  
    <lastName>Doe</lastName>  
  </employee>  
  <employee>  
    <firstName>Anna</firstName>  
    <lastName>Smith</lastName>  
  </employee>  
  <employee>  
    <firstName>Peter</firstName>  
    <lastName>Jones</lastName>  
  </employee>  
</employees>
```


Teil 3

Hilfreiches, Tipps und Links

Hilfreiches, Tipps und Links

- Node.js herunterladen und installieren
 - Website: [\[https://nodejs.org/\]](https://nodejs.org/)
 - Aktuelle Version für alle Systeme (Windows, MacOS X, Linux)
 - npm-Paketmanager wird automatisch mit Node.js installiert
 - Benutzung unter Windows:



```
Node.js command prompt - node app.js

Your environment has been set up for using Node.js 4.2.1 (x64) and npm.

C:\Users\LucasRecknagel>cd Downloads\server

C:\Users\LucasRecknagel\Downloads\server>node app.js
Example app listening at http://:::3000
json wrote successfully!
```

- Node.js herunterladen und installieren

- Benutzung unter Linux:

```
sudo apt-get update
sudo apt-get install nodejs
sudo apt-get install npm
node app.js
```

- Benutzung unter OSX:

- via .pkg File von der Node.js Website
 - via Homebrew:

```
brew install node
```

■ Links:

- Kostenlose Tutorials / Workshops: [\[http://nodeschool.io/\]](http://nodeschool.io/)
- RESTful Cookbook: [\[http://restcookbook.com/\]](http://restcookbook.com/)
- Node.js Einführung: [\[http://netzleben.com/2013/12/node-js-eine-einfuehrung-fuer-anfaenger-teil1/\]](http://netzleben.com/2013/12/node-js-eine-einfuehrung-fuer-anfaenger-teil1/)

Fragen?



Interactive Media Lab Dresden
Professur für Multimedia-Technologie

Kontakt:

Ricardo Langner (ricardo.langner@tu-dresden.de)

Philipp Heisig (philipp.heisig@tu-dresden.de)

Changelog

Datum / Zeit	Beschreibung
2017-11-20 09:00	▪ Initiale Download-Version
2017-11-20 09:56	▪ Folien für Aufgabe wieder hinzugefügt, Datum (Jahr) Abgabe korrigiert