

# 3 Übung Web- und Multimedia-Engineering

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

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

# Terminplan / Ablauf

Woche	Datum	Übungsthema /-inhalt
KW 42	12./13.10.	keine Übung
KW 43	19./20.10.	Einführung Aufgabenstellung A1: HTML + CSS + JS  → Screenshots, Logo, Daten (CSV)
KW 44	26./27.10.	Konsultation
KW 45	02./03.11.	Abgabe A1
KW 45	02./03.11.	Einführung Aufgabenstellung A2: PHP + XML → Materialien für A2
KW 46	09./10.11.	Lösung A1 und Konsultation
KW 47	16./17.11.	Abgabe A2
KW 47	16./17.11.	Einführung Aufgabenstellung A3: Webservices → Materialien für A3
KW 48	23./24.11.	Lösung A2 und Konsultation
KW 49	30.11./01.12.	Konsultation
KW 50	07./08.12.	Konsultation
KW 51	14./15.12.	Abgabe A3
KW 51	14./15.12.	Einführung Aufgabenstellung A4: Anwendungsbeispiel → Materialien für A4
KW 52	21./22.12.	keine Übung
KW 53	28./29.12.	keine Übung
KW 1	04./05.01.	Lösung A3 und Konsultation
KW 2	11./12.01.	Konsultation
KW 3	18./19.01.	Abgabe A4
KW 3	18./19.01.	Abschluss, Feedback und Fragen
KW 4	25./26.01.	Lösung A4 und offene Fragen (nur nach Anmeldung)
KW 5	01./02.02.	keine Übung

# 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

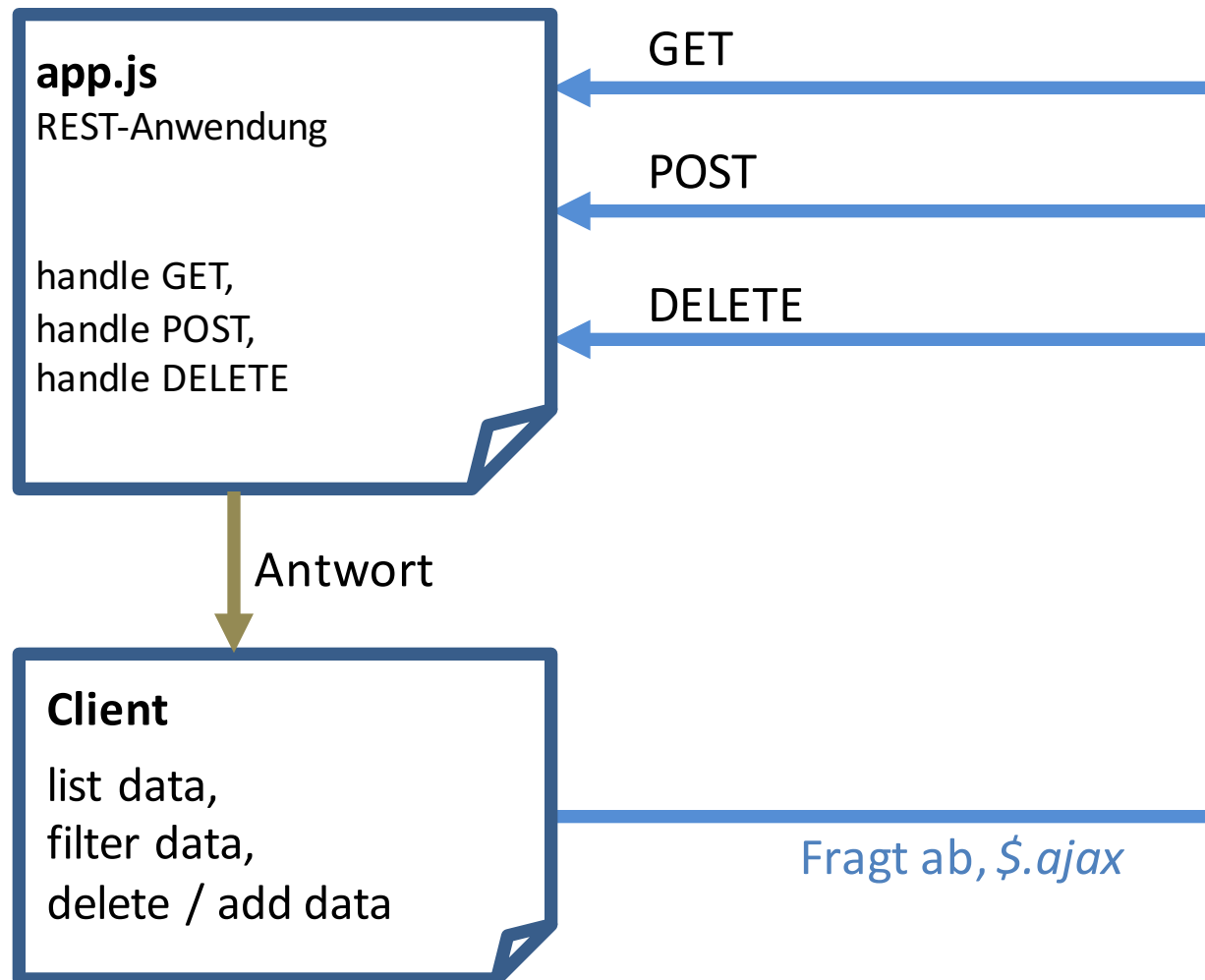


Teil 1

## Aufgabenstellung A3: Node.js und AJAX

- Basis ist
  - Vorlage aus den Materialien für A3 (HTML & JS Skeleton)
- Aufgabe umfasst insgesamt 3 Arbeitspakete
  1. Serverseitiges Parsen einer CSV Datei & Speichern als JSON
  2. Erstellen einer REST-API
  3. Abfragen der REST-API via AJAX

## ■ Schematischer Aufbau der REST-Anwendung



## 1. Serverseitiges Parsen einer CSV Datei & Speichern als JSON

- ☐ Bereitgestellte Datei `app.js` im Bereich: „csv2json“ erweitern
- ☐ Node.js Modul: `csv2json` für die Umwandlung nutzen
- ☐ Globale Variable für das JSON-Objekt anlegen
- ☐ CSV laden und in JSON umwandeln und in der globalen Variable speichern, die Speicherung als Datei ist optional

<https://www.npmjs.com/package/csv2json>

```
[
  {
    "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

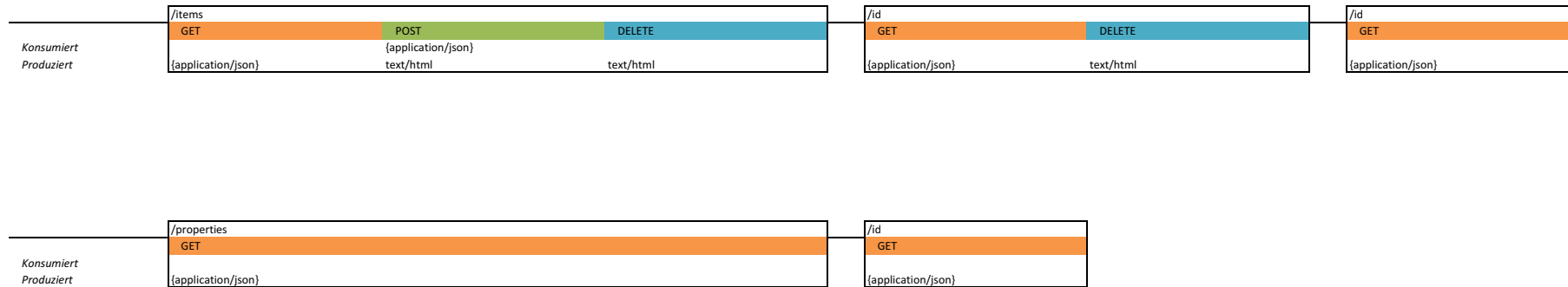
- ☐ `/items` -> löscht letztes Land aus der Liste, Status: „Deleted last country: {name}!“
- ☐ `/items/id` -> löscht Land mit der ID id, Status bei Erfolg: „Item {id} deleted successfully.“ oder bei Fehler: „No such id {id} in database“

### ☐ Bonus:

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

## 2. Erstellen einer REST-API

Schnittstellen-  
beschreibung



Element-  
beschreibung

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

properties
birth_rate_per_1000
cell_phones_per_100
children_per_woman
electricity_consumption_per_capita
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 => 2.1.4) darf verwendet werden
- ☐ Implementieren der AJAX-Abfragen und Verarbeitung für
  - ☐ „Filter Countries“ (Anfragen an GET Calls)
    - ☐ Wenn Range angegeben wurde -> Abfrage der Range (Range > Single id)
  - ☐ „Properties“ (Anfragen an GET Calls)
  - ☐ „Add Country“ (Anfragen an POST Calls)
  - ☐ „Delete Country“ (Anfragen an DELETE Calls)
    - ☐ Wenn keine id angegeben wird -> letztes Land aus der Liste löschen
    - ☐ Wenn id angegeben wird -> Land mit id aus der Liste löschen

<b>Filter Countries</b> country id <input type="text"/> country id range (eg: 2-5) <input type="text"/> <b>Filter Countries</b>	<b>Properties</b> <input type="text" value="gps_long"/> <b>Show</b> <b>Hide</b>	<b>Add Country</b> country name <input type="text"/> birth rate / 1000 <input type="text"/> cellphones / 100 <input type="text"/> <b>Add Country</b>	<b>Delete Country</b> Country ID to delete <input type="text"/> <b>Remove Country</b>
--	---	---	--

## ■ 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
- ☐ Skeleton (Material A3) benutzen, Dateistruktur:

```
Team_XX
  app.js
  node_modules
  package.json
  public
  assets
    css
      font-awesome
      html5reset.css
      style.css
    js
      jquery-2.1.4.js
      ajax.js
    img
      world_data_logo.png
  index.html
  world_data.csv
```

- Erlaubte Hilfsmittel: jQuery, Nodejs-Module: express, csvtojson, body-parser 
- Testen: Node.js (Version **>= 4.2.2 LTS** oder **>= 0.12.7**),  
jQuery (Version **>= 2.1.4**),  
Firefox (aktuelle Version)
- Abgabe: Montag/Dienstag, **14.12.2015 & 15.12.2015** 

Teil 2

# **Einführung Node.js & AJAX**

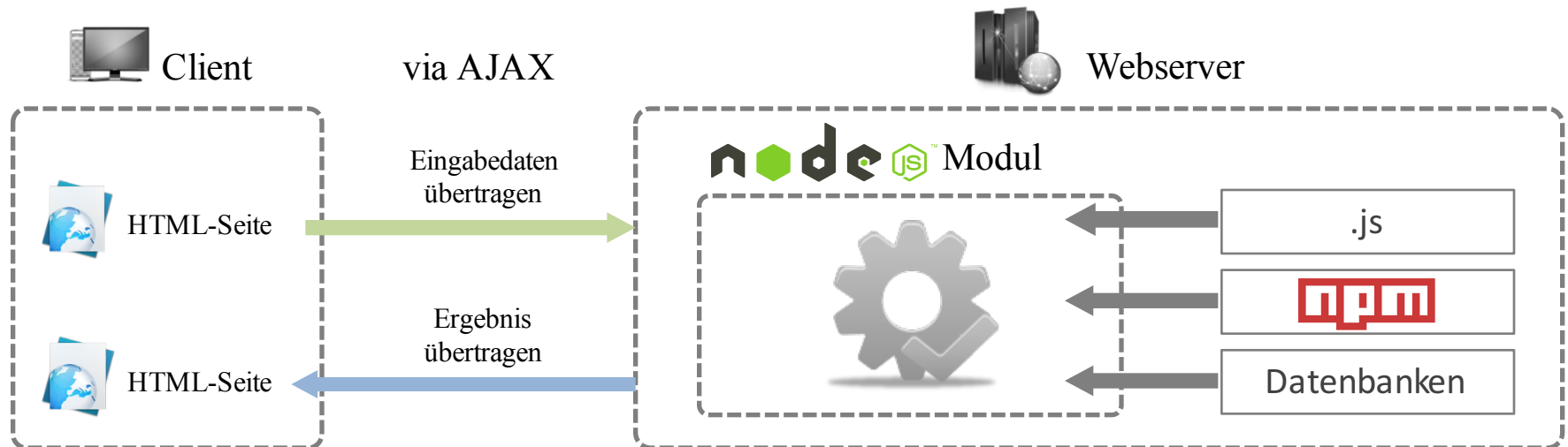
Teil 2a

**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 ca. 204.965 Module



# Node.js – helloworld.js

```
// 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/>

- 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');`

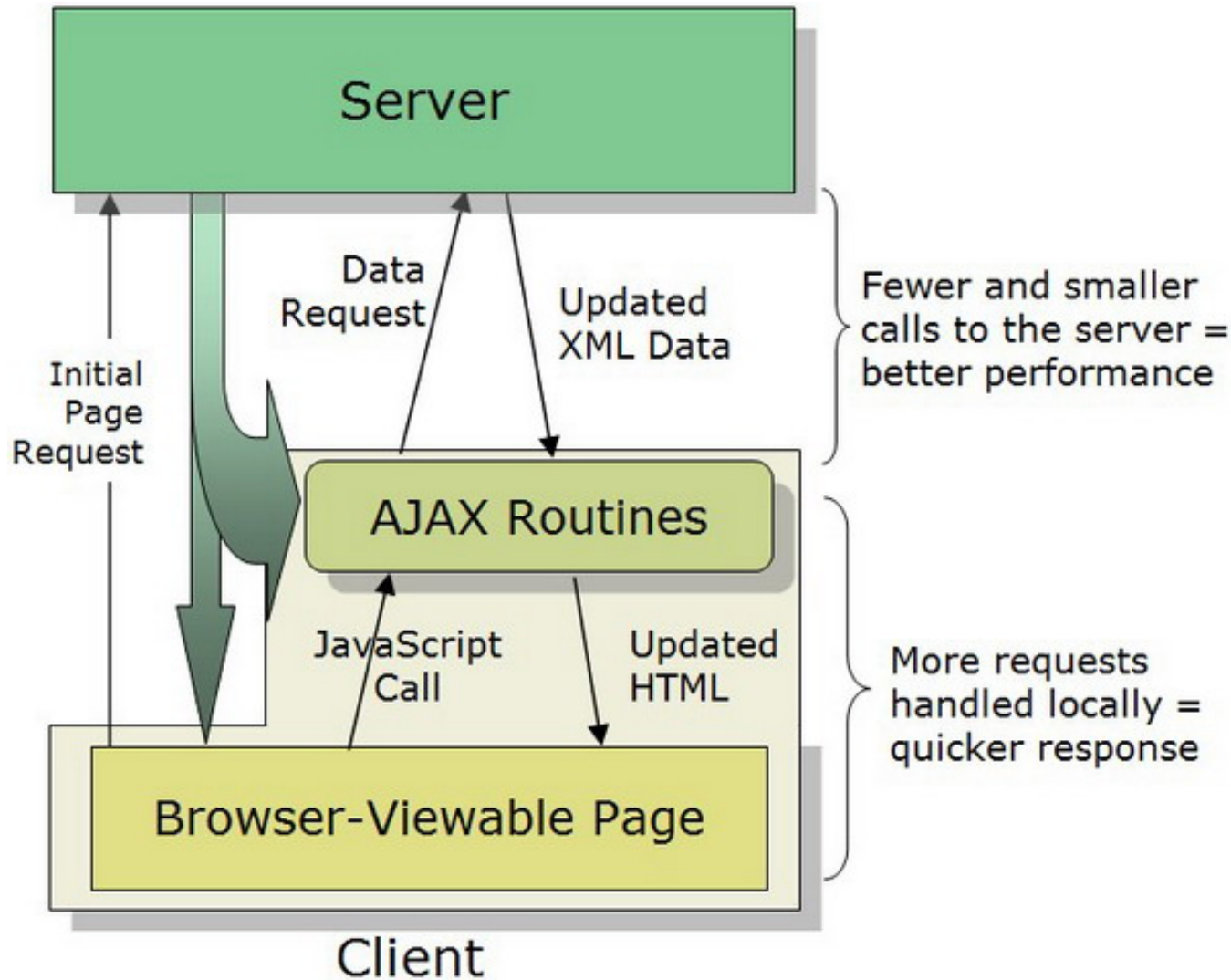
- Ausführliche Einführung:

- Node.js Einführung -> Folie 26

Teil 2b

## **AJAX – Asynchronous JavaScript and XML**

# AJAX – Asynchronous JavaScript and XML



# AJAX - Beispiel

```
$.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  
    }  
});
```

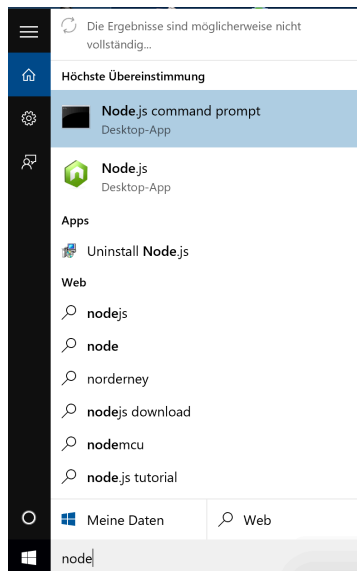
- type – anzuwendende HTTP-Methode
- url – anzusprechende URL, die auf die Methode reagiert

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/>“

# Fragen?



**Interactive Media Lab Dresden**  
Professur für Multimedia-Technologie

*Kontakt:*

Ricardo Langner ([ricardo.langner@tu-dresden.de](mailto:ricardo.langner@tu-dresden.de))

Lucas Recknagel ([lucas.recknagel@tu-dresden.de](mailto:lucas.recknagel@tu-dresden.de))

# Changelog

Datum / Zeit	Beschreibung
2015-11-13 21:00	▪ Initiale Download-Version
2015-11-16 10:13	▪ Aufgaben stellenweise konkretisiert