NodeJS
Part 2

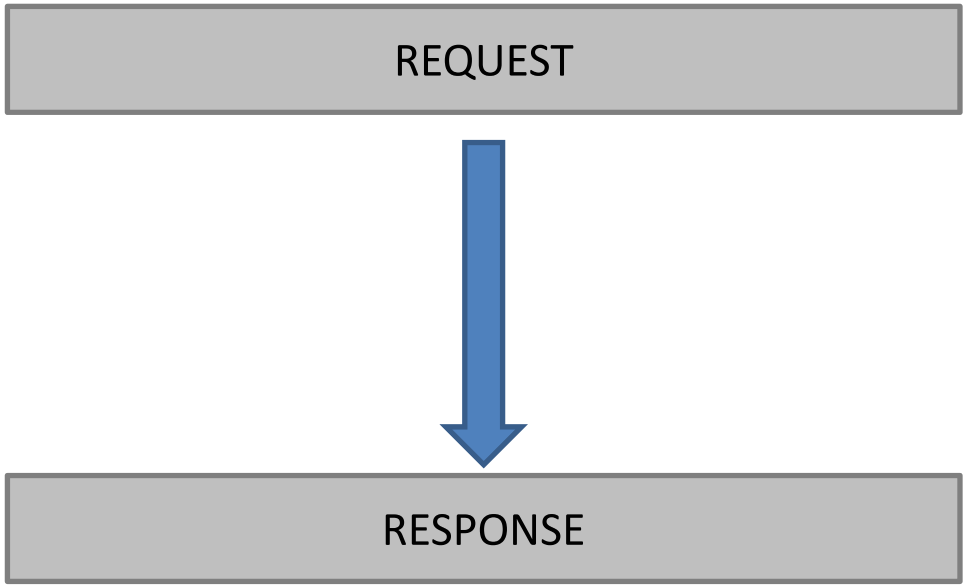
jan.schulz@devugees.org

Agenda

1. Middleware
2. Templates
3. Relational Databases: SQL

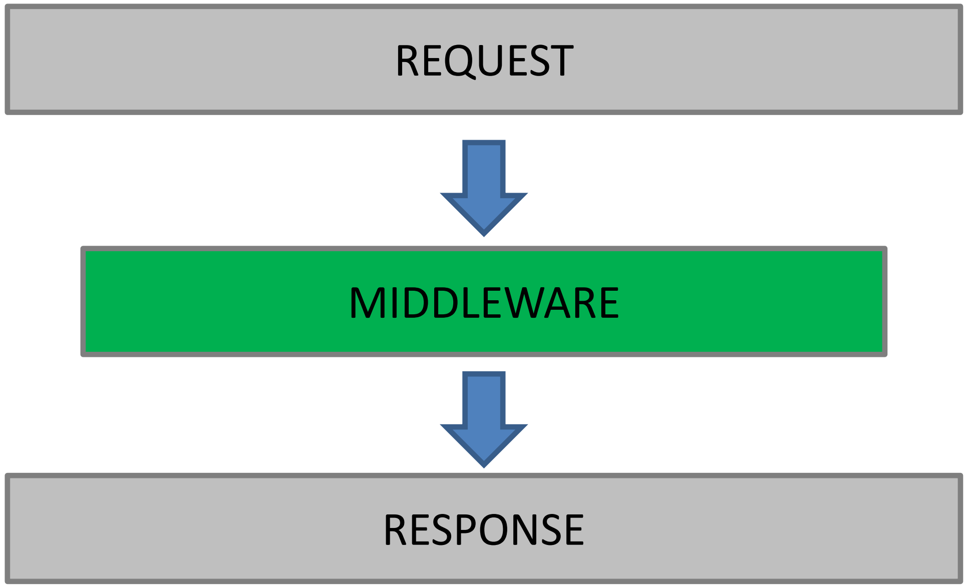
1. Middleware

- Middleware: Code that sits between two layers of software



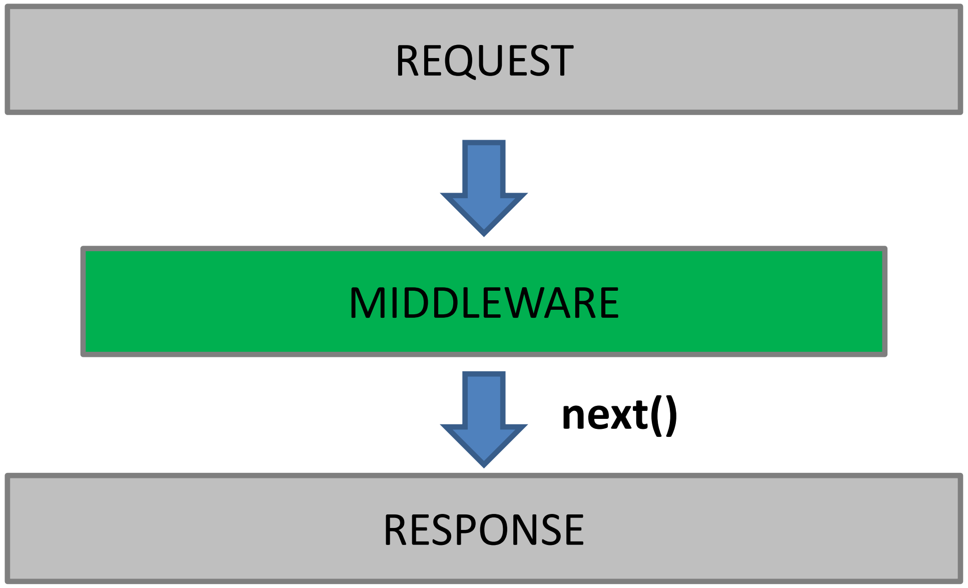
1. Middleware

- Middleware: Code that sits between two layers of software



1. Middleware

- Middleware: Code that sits between two layers of software



1. Middleware

- Middleware: **creates re-usable ways of dealing with HTTP-requests**

2. Templates

- A blueprint for HTML-files
- Placeholders will be replaced by JavaScript-variables when a template is rendered

```
<html>
  <head></head>
  <% if (halloworld) %>
      <h2><%=halloworld%></h2>
  <% } %>
  <body></body>
</html>
```

2. Templates

Task:

Convert your personal blog to your own NodeJS server.

1.) Create a static asset /public
2.) Implement a route GET /
3.) Convert your HTML file to an EJS template.
4.) Implement two routes in your app.js
    GET /contacts
    POST /contacts

- for GET /contacts
  1. Open a file contacts.json which is in the root folder of the server.
  2. If it does not exist, create it with the initial content "[]". Otherwise, read its contents and return them as JSON.
- for POST /contacts
  1. Open a file contacts.json which is in the root folder of the server.
  2. If it does not exist, create it with the initial content "[]". Otherwise, read its content and parse it as a JavaScript object (an array of objects).
  3. Read the POST body: you should receive 3 variables: name, email and text.
  4. Construct an object in which you store name, email and text.
  5. Push .4) into the array of 2.)
  6. Stringify 5.) as JSON and then save it as contacts.json. (overwrite contacts.json)

5.) Change your IP-address in your $.ajax – Request in your main.js to http://localhost:3000/contacts.
6.) Test it.

EXCOURSE: COMPLEXITY

- Becoming a developer is not easy

EXCOURSE: COMPLEXITY

- Becoming a developer is not easy
- It involves dealing with complexity

EXCOURSE: COMPLEXITY

- Becoming a developer is not easy
- It involves dealing with complexity
- Dealing with complexity:

Divide and Conquer

EXCOURSE: COMPLEXITY

$$\frac{(a+b)^2}{2} + (x+y)^a = ?$$

This is complexity. It gives you the feeling of being overchallenged which is **normal**.

How can you approach complexity?

EXCOURSE: COMPLEXITY

- **Divide and Conquer:**

Dividing a complex problem into simple problems will make the complex problem a simple problem.

EXCOURSE: COMPLEXITY

$$\frac{(a+b)^2}{2} + (x+y)^a = ?$$

We gather information ...

It is given that:

1. a,b,x,y > 0
2. a,b,x,y are natural numbers = { 1,2,3,4, ..., N } without the 0
3. x + a = 2
4. b < y
5. b + 4 = y / 4
6. b + 1 = 2

... those are all information you need to solve the problem.

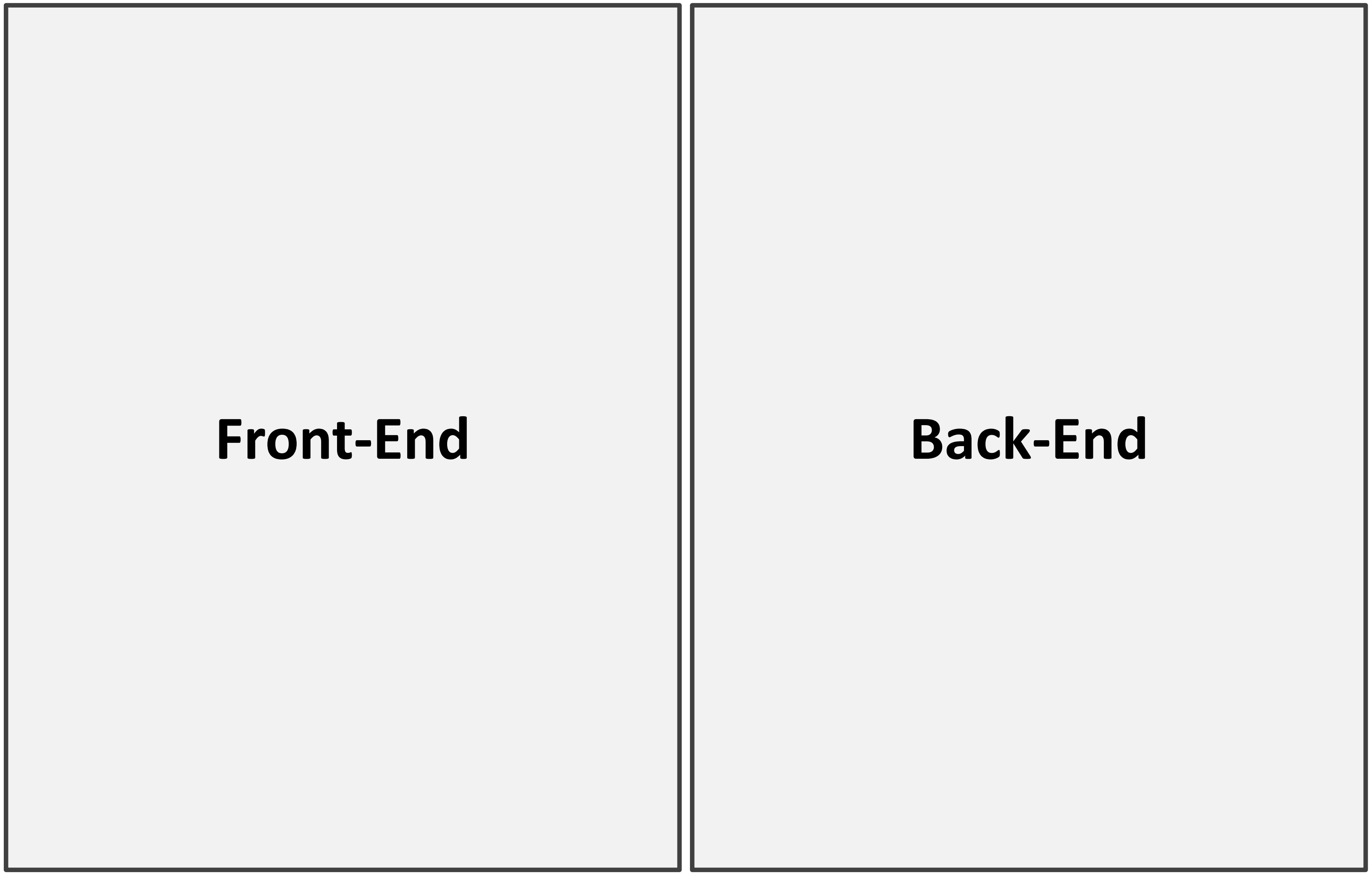
Task: Divide and Conquer this problem and solve the equation.

EXCOURSE: COMPLEXITY

Task: Create a route /sum which accepts two body-parameters x and y and which uses the method POST. The route /add should return the sum of x + y as JSON.
I.e. If x = 2 and y = 3, /sum should return
{ “sum“: 5 }

Do not implement it. Analyze this task and until it appears simple to you! Divide and Conquer!

EXCOURSE: HTTP



EXCOURSE: HTTP

Front-End

=

HTTP-Client
(Chrome, Firefox, ...)

Back-End

=

HTTP-Server
(NodeJS, PHP, ...)

EXCOURSE: HTTP

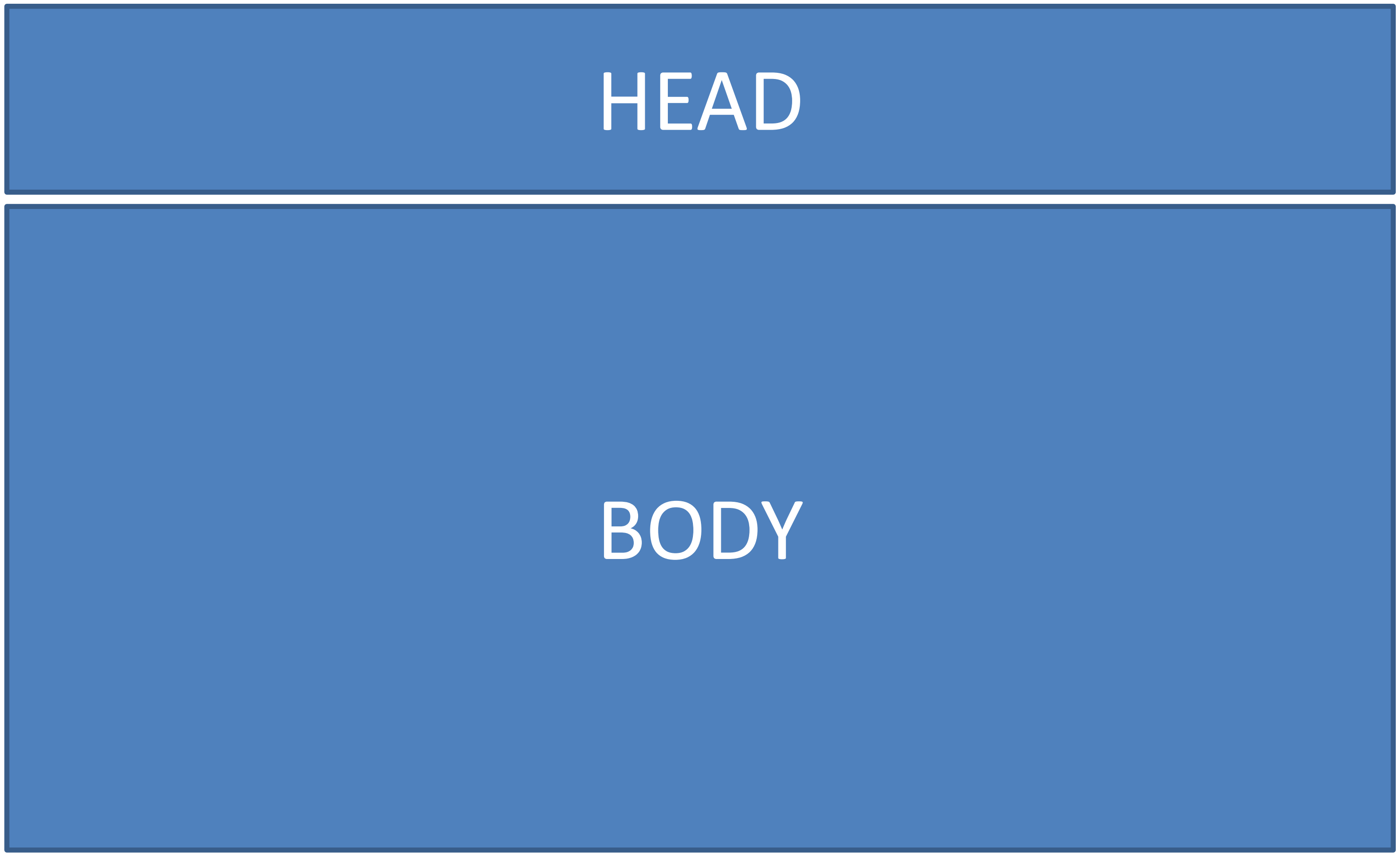
- What is HTTP?
  - HTTP: Protocol
  - Protocol = Set of commands
  - Most used HTTP-commands
    - **GET:** Reading a resource from a server
    - **POST:** Creating a new resource on a server
- Command is either a ...
  - REQUEST
  - RESPONSE

EXCOURSE: HTTP

- What is HTTP?
  - HT
  - Pro
  - Mo
    -
    -
- Comr
  - REQUEST
  - RESPONSE

How does an HTTP-command look like?

EXCOURSE: HTTP



EXCOURSE: HTTP

The request
/GET has an empty body!

EXCOURSE: HTTP

- There are **17 HTTP-REQUESTS**
  - GET -> show me a resource
  - POST -> create a new resource based on the information in the request‘s body
  - PUT -> change a resource based on the requests‘ body
  - DELETE -> delete a resource
  - ...

EXCOURSE: HTTP

- There are **50+ HTTP-RESPONSES**
  - 200 -> OK, your request was processed completely
  - 304 -> the last 200 was not modified
  - 404 -> resource not found
  - 400 -> bad request, i.e. wrong request head or body
  - 408 -> timeout, processing the request took too long

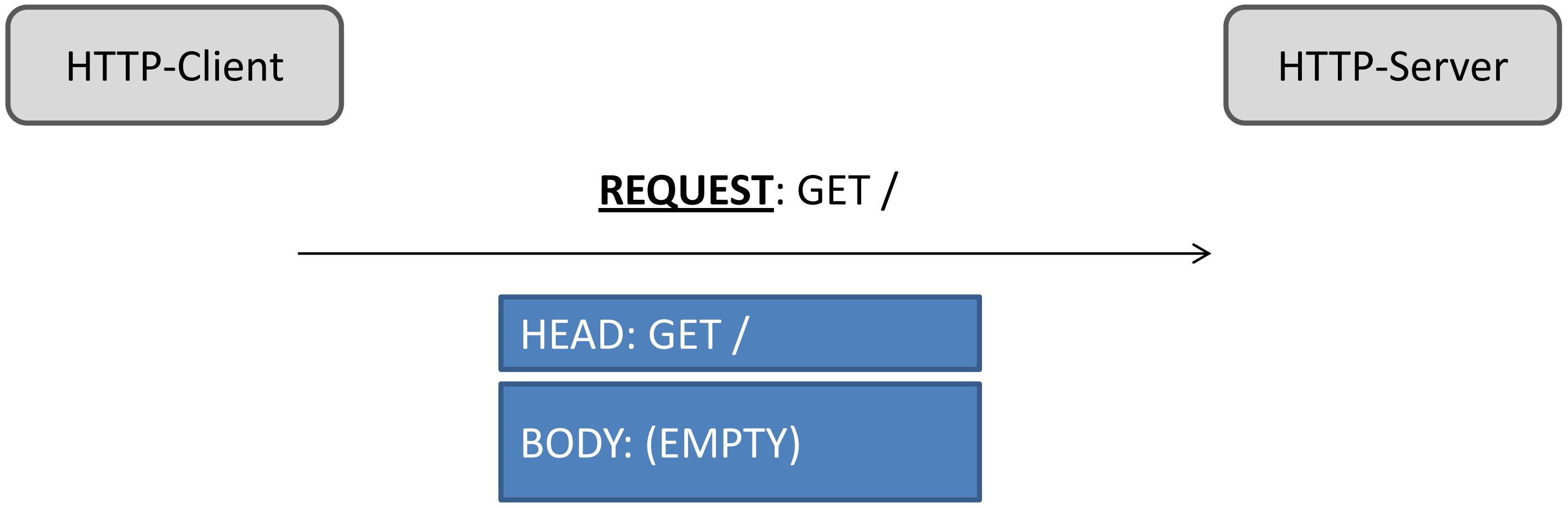
EXCOURSE: HTTP

- We are dealing with GET and POST requests
- **GET requests** have a head and an empty body
- **POST requests** have a head and an non-empty body

EXCOURSE: HTTP

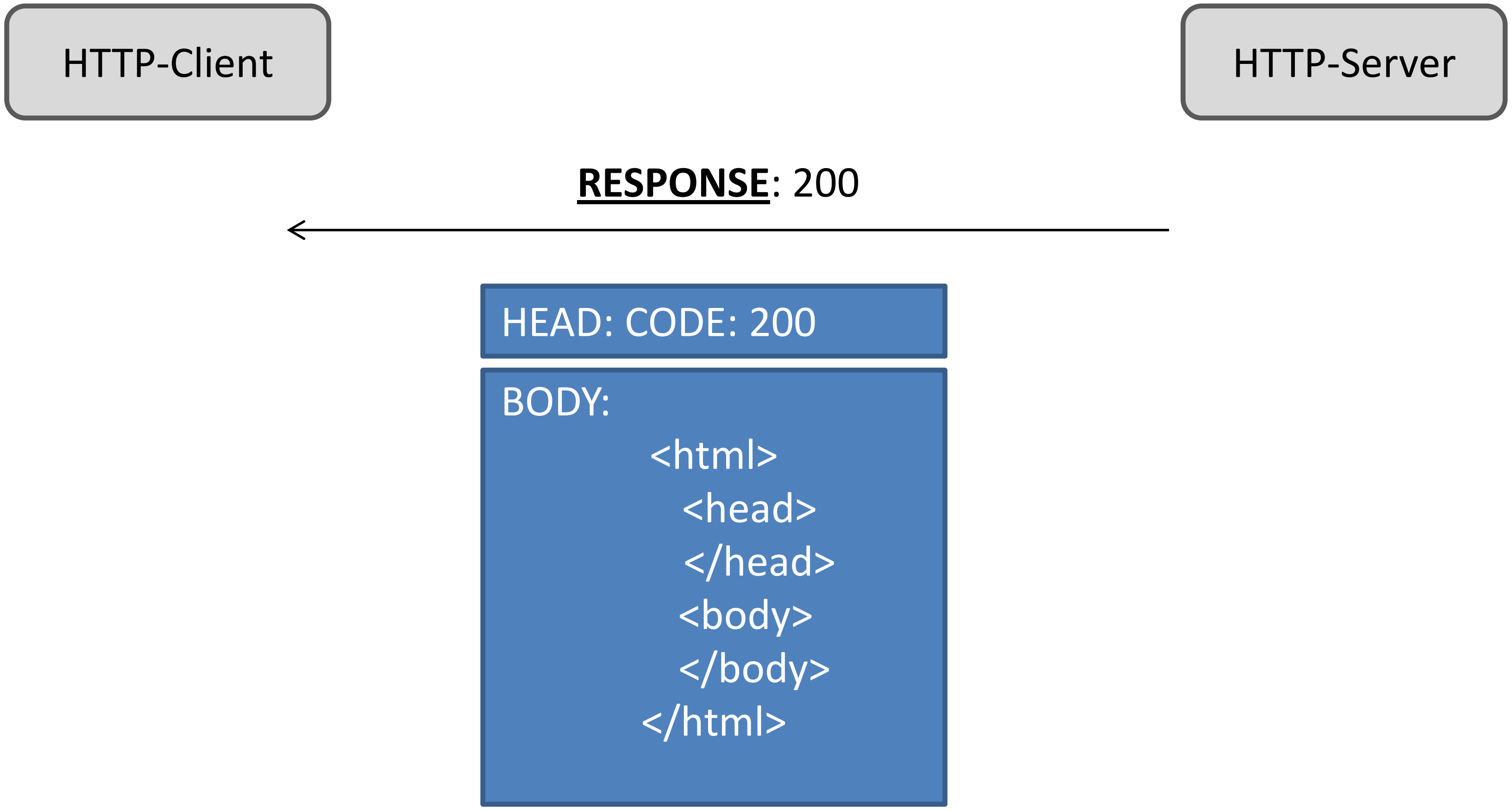
<u>Client/Server communication example</u>

You open a website www.google.com



EXCOURSE : HTTP

<u>Client/Server communication example</u>
You receive an answer from Google.com

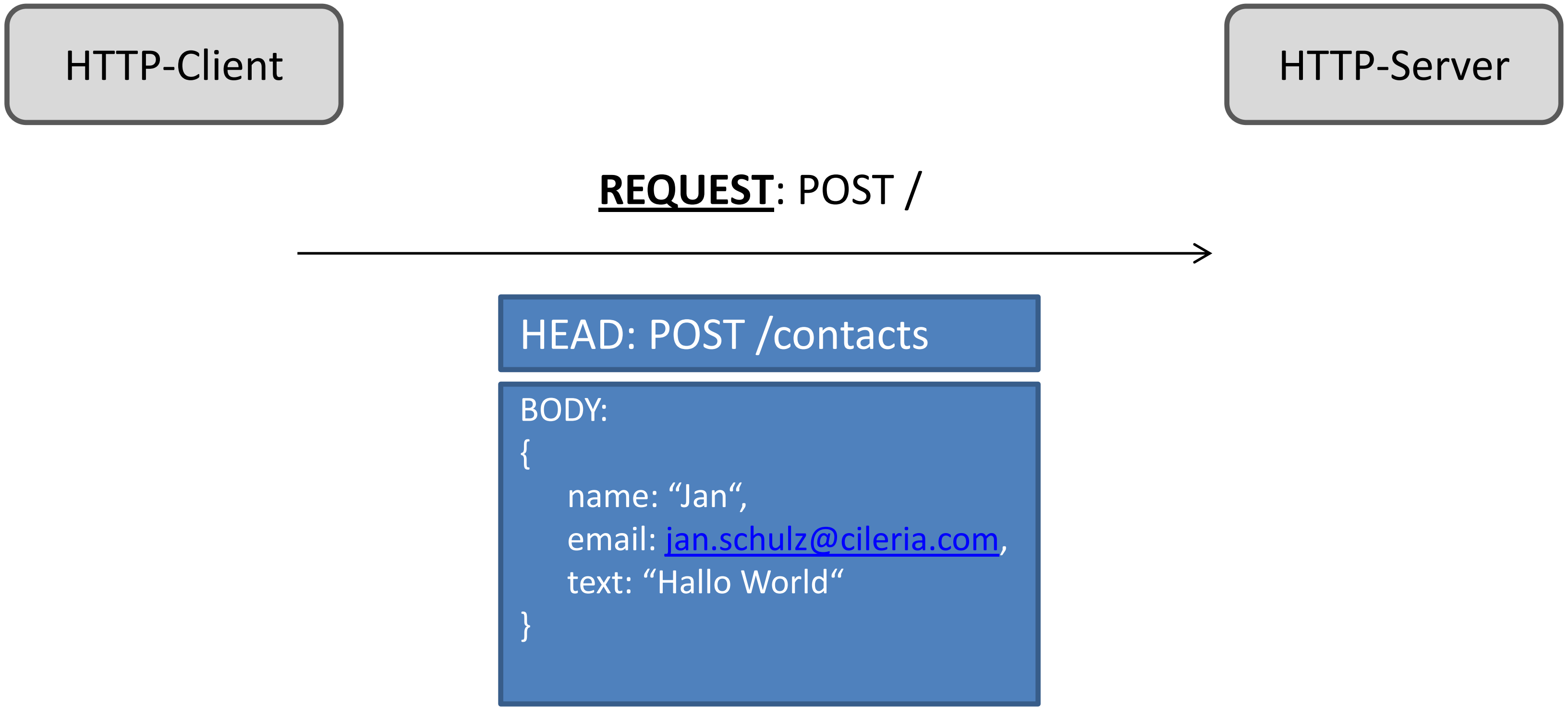


```
<html>
  <head>
  </head>
  <body>
  </body>
</html>
```

EXCOURSE : HTTP

Client/Server communication example

You post a new contact request to your localhost/contacts

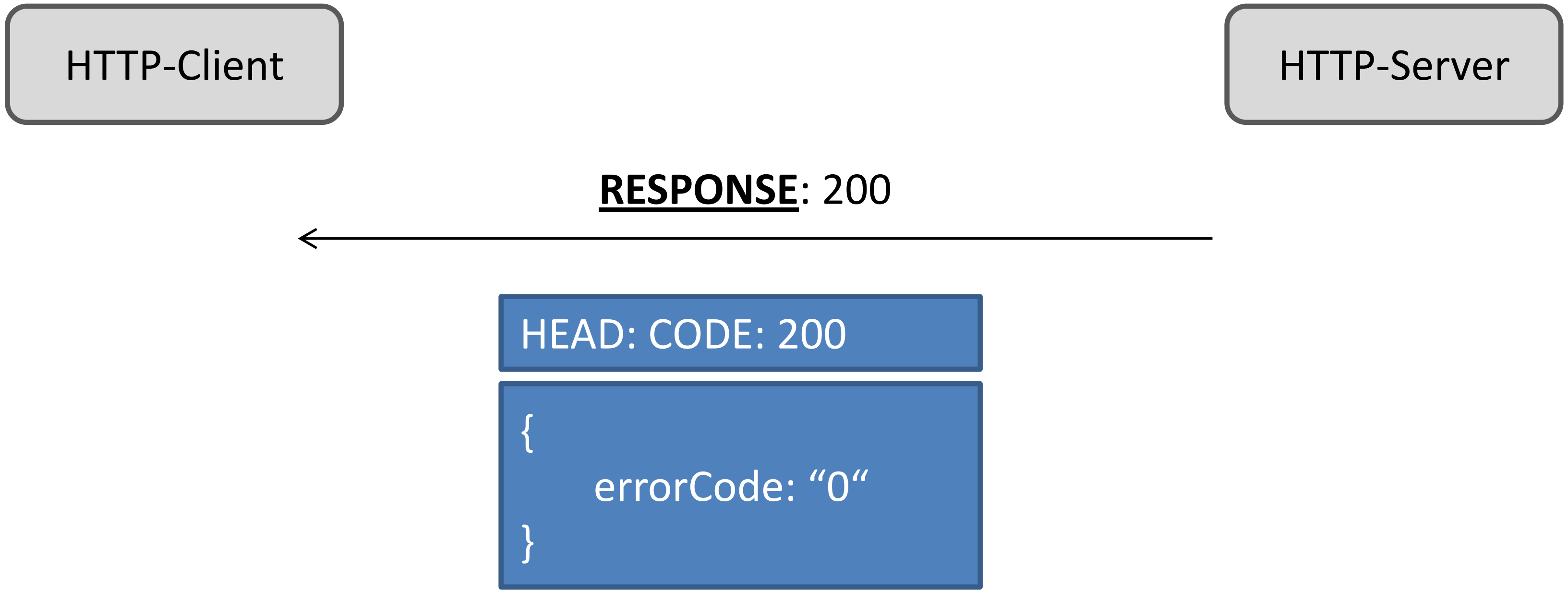


```
BODY:
{
    name: “Jan“,
    email: jan.schulz@cileria.com,
    text: “Hallo World“
}
```

EXCOURSE : HTTP

<u>Client/Server communication example</u>

You receive an answer from localhost



```
HEAD: CODE: 200
```

```
{
      errorCode: “0“
}
```

EXCOURSE: HTTP

- What‘s the purpose of browsers like Chrome/Firefox/etc. ?

EXCOURSE: HTTP

- What‘s the purpose of browsers like Chrome/Firefox/etc. ?
- They
  1. GET the HTML, JavaScript and CSS

EXCOURSE: HTTP

- What‘s the purpose of browsers like Chrome/Firefox/etc. ?
- They
    1. GET the HTML, JavaScript and CSS
    2. **<u>Render</u>** a website using HTML and CSS

EXCOURSE: HTTP

- What‘s the purpose of browsers like Chrome/Firefox/etc. ?
- They
  1. GET the HTML, JavaScript and CSS
  2. **Render** a website using HTML and CSS
  3. **Compile** JavaScript and make the website interactive.

EXCOURSE: HTTP

Task: 25 mins

1. Describe the difference between frontend and backend development.
2. Describe the difference between JavaScript run in your browser and JavaScript run on your NodeJS-server.
3. Describe the difference between $.get() (Jquery) and app.get() (NodeJS).
4. Do you have access on your DOM-elements in your NodeJS – application?
5. Do you have access to the filesystem (i.e. “/home/user/halloworld.txt“) from our frontend code?

EXCOURSE: Teamwork

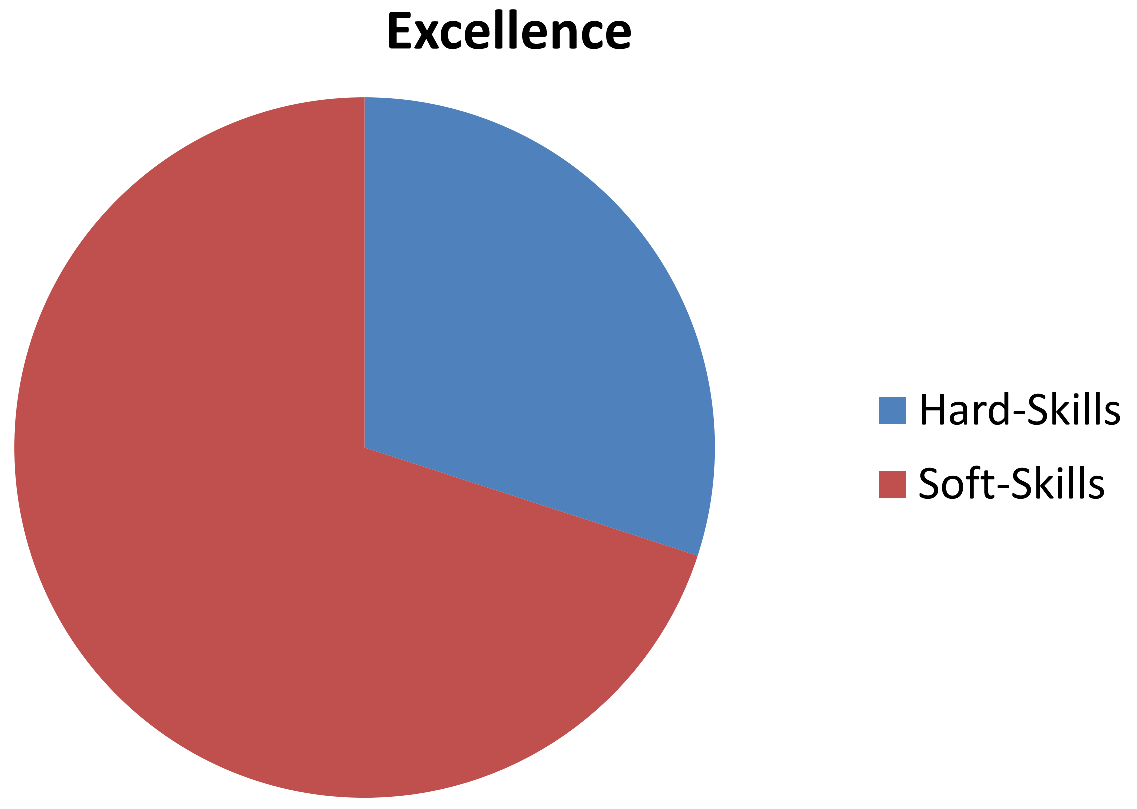
EXCOURSE: Teamwork

EXCOURSE: Teamwork

EXCOURSE: Teamwork

It is only fun working with <u>**excellent**</u> people.

EXCOURSE: Teamwork



EXCOURSE: Teamwork

<u>**However, Hard-Skills are a precondition.**</u>

Without any Hard-Skills, Soft-Skills become meaningless. The team does not need you.
=> The 5th Weel. No success.

Without any Soft-Skills, the team does not want you and takes the first chance of getting rid of you.
=> The person non-grata. Low success.

EXCOURSE: Teamwork

Success

<u>Perfect Developer:</u>

Excellent Communicator

+

Very Good Developer

EXCOURSE: Teamwork

What does this picture tell you?

3. Relational Databases - SQL

- Relational Database:

ID	Firstname	Lastname
17	John	Doe

3. Relational Databases - SQL

- Relational Database:

ID	Firstname	Lastname
17	John	Doe

Person	Street	Number
17	Main Street	2

3. Relational Databases - SQL

- Relational Database:

ID	Firstname	Lastname
17	John	Doe

Person	Street	Number
17	Main Street	2

Task: How would this look as JavaScript Object?

3. Relational Databases - SQL

```
{
   firstname: ‘John‘,
   lastname: ‘Doe‘,
   address: {
        street: ‘Main Street‘,
        number: 2
   }
}
```

3. Relational Databases - SQL

<u>Task:</u>

Implement two routes for your personal blog NodeJS server.

GET /contact

  Lists all of your contact requests

POST /contact

  Creates a new contact request.

Note: Create a MySQL database and a table for this.