JavaScript Beginner's Course Part 4

jan.schulz@devugees.org

Agenda

- 1. Functions as Arguments & Callbacks
- 2. Object Iterator
- 3. JSON
- 4. Session Storage
- 5. Local Storage
- 6. HTTP Protocol
- 7. AJAX

1. Functions as Arguments

```
function whichOne(f) {
  f();
function a() { console.log('i am a'); }
function b() { console.log('i am b'); }
function c() { console.log('i am c'); }
whichOne(b);
```

1. Callbacks

```
function waitForSomething(callback) {
  ... // important calculations here ...
  callback();
waitForSomething( function() {
  alert('Hallo World');
});
```

2. Object Iterator

Object iteration:

```
for(key in Obj) {
   ...
}
```

- JSON = JavaScript Object Notation
- Data format for interchanging objects accross multiple system

```
person = {
    'firstname': 'Jan',
    'age': 31 }
}
or
[ 1, 3, 4, 9, 10 ]
```

- JSON = JavaScript Object Notation
- Data format for interchanging objects accross multiple system

```
person = {
    "firstname" : "Jan",
    "age" : 31 }
}
or
[ 1, 3, 4, 9, 10 ]
```

JSON keys are always strings!

- Convert JavaScript Object to JSON String var x = JSON.stringify(OBJ);
- Parse JSON String to JavaScript Object var y = JSON.parse(str);

 Great tool to view JSON Data: http://jsonviewer.stack.hu

2. Session Storage

window.sessionStorage

- An object which has key value relationships
 - string only, no objects
- Until the browser is closed, data can be saved in the sessionStorage object
- Client-side storage, 5m size
- Methods:

```
.setItem( key, value)
.getItem( key )
.removeItem( key )
.clear( )
```

3. Local Storage

window.localStorage

= Same as sessionStorage

EXCEPT: Data is saved until user explicitly removes it, no pre-defined expiration date and 10m size

Back-End Front-End

Front-End

_

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

Back-End

=

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

- 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

```
    What is HTTP?

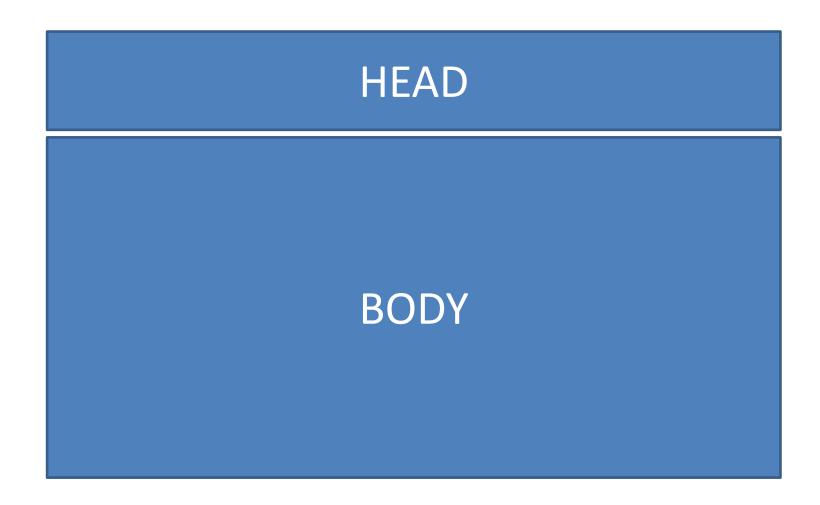
  -HT
  — Pro

    Mo How does an HTTP-command look

                        like?

    Comr

  - REQUEST
  - RESPONSE
```





- 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

— ...

- 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

- 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

Client/Server communication example

You open a website www.google.com

HTTP-Client

HTTP-Server

REQUEST: GET /

HEAD: GET /

BODY: (EMPTY)

Client/Server communication example

You receive an answer from Google.com

HTTP-Client

HTTP-Server

RESPONSE: 200

HEAD: CODE: 200

BODY:

<html>

<head>

</head>

<body>

</body>

</html>

Client/Server communication example

You post a new contact request to your localhost/contacts

HTTP-Client

HTTP-Server

REQUEST: POST /

HEAD: POST /contacts

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

Client/Server communication example

You receive an answer from localhost

HTTP-Client

HTTP-Server

RESPONSE: 200

HEAD: CODE: 200

errorCode: "0"

- 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

- 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.** <u>Compile</u> JavaScript and make the website interactive.

Task: 15 mins

- 1. Describe the difference between frontend and backend development.
- 2. Describe the difference between JavaScript run in your browser and JavaScript run on a server.
- 3. Do you have access to the filesystem (i.e. "/home/user/halloworld.txt") from our frontend JavaScript code?

- Tools to test HTTP Requests
 - Postman
 - CURL
 - WGET
- Tools to parse JSON
 - JSONViewer

4. AJAX

- Asynchronouse JavaScript XML Requests
 - HTTP-Requests that not
 - Include Source Files like CSS, JS etc.
 - Include Initial page load like Hitting F5 + Reloading the page
 - Do not block other JavaScript code

4. AJAX

- Asynchronouse JavaScript XML Requests
 - HTTP-Requests that not
 - Include Source Files like CSS, JS etc.
 - Initially load the page like Hitting F5 + Reloading the page
 - In total, 17 types of HTTP-Requests
 - GET read a resource
 - POST create a new resource
 - PUT change an existing resource
 - DELETE delete an existing resource

• ...

4. AJAX

- window.XMLHTTPRequest
- Methods:
 - open(): defines the HTTP method and URL
 - onload(): defines an event that handles the response data
 - send(): actually sends the request to the server

<u>Users</u> | Comments | Add User | Add Comment

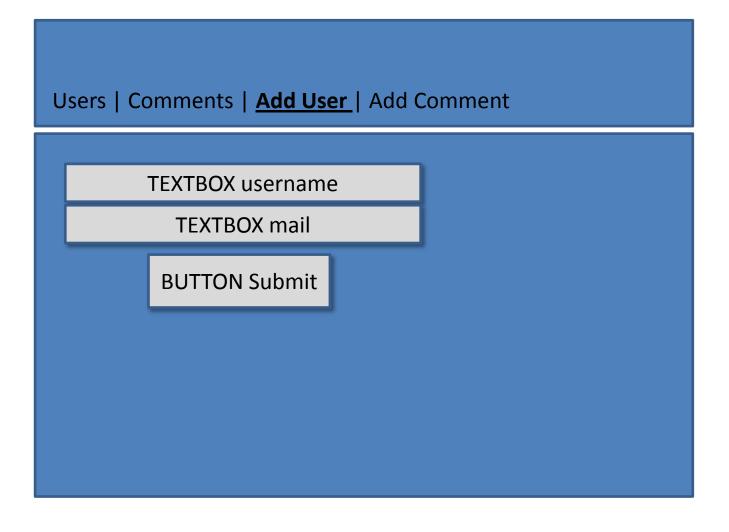
Name	Email	ID
Test1	test1@gmail.com	1
Test2	test2@gmail.com	2
Test3	test3@gmail.com	3

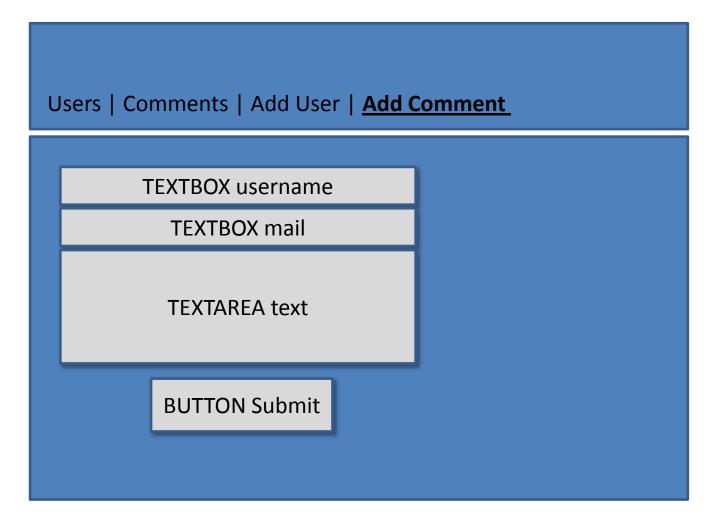
Users | Comments | Add User | Add Comment

From Test1

Email is Test1@gmail.com

Hallo, I think that Test1 is a super great user name.





- 1. Send a GET request to http://35.156.88.18/users and check the response. How can you convert this to a JavaScript object?
- 2. Send the GET request with AJAX to the server and parse the response.
- 3. Use the response to create a table of users.
- 4. Do 1., 2. and 3. for the comments that you can get via http://35.156.88.18/comments

- 5. Send a POST request to http://35.156.88.18/users with a POST body of {name: 'Bob', username: 'Smith', email: 'bob@gmail.com'}. Use Postman for this.
- 6. Send the POST request with AJAX to the server based on your form data.
- 7. Check if a new user has been added.
- 8 Do 5., 6. and 7. for the comments that you can POST to http://35.156.88.18/comments. This time the POST body looks like {name: 'Bob', email: 'bob@gmail.com', body: 'Hallo World'}