NodeJS
Part 4

Jan.schulz@devugees.org

1. Agenda

1. Build-Tools
2. Scaffolding
3. ECMA Script 2015/2016
4. Cross Origin Response Header
5. Our Backend

1. Build-Tools

A Build-Tool is a NodeJS module that helps us to automate tasks.

1. Build-Tools

1. Chrome & Firefox do not understand SASS and LESS ...
2. Uglify + Minify
   - Minify JS = ...
   - Uglify = ...
3. Live-Reload ...
4. Show JS-Errors ...
5. Front & Backend Development is separated
6. Incompatibility to ES2016
   i.e. ‘import‘ not supported by older browsers

1. Build-Tools

1. Chrome & Firefox do not understand SASS and LESS ... ***Transpile it to CSS***
2. Uglify + Minify
   - Minify JS = ***Make the source code size small***
   - Uglify = ***Make the source code unreadable***
3. Live-Reload ... ***Run a livereload server***
4. Show JS-Errors ... ***Fix JS errors faster***
5. Front & Backend Development is separated ...
   npm run can start them both using one command
6. Incompatibility to ES2016
   i.e. ‘import‘ not supported by older browsers
   Transpile to previous verson

1. Build-Tools

7. Bundling:

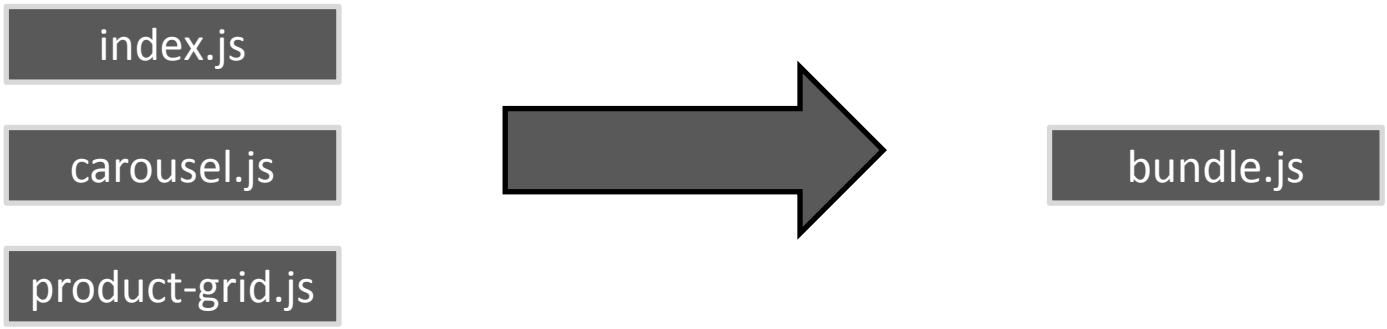
index.js

carousel.js

product-grid.js

1. Build-Tools

7. Bundling:



WHY ?

1. Build-Tools

7. Bundling:

Without bundling, the browser has to load 3 files one by one.
SLOW!

Loading 1 file is **FASTER**.

1. Build-Tools

- Webpack
- Gulp
- Browserify
- NPM scripts
- Grunt

They are quite similar. It comes down to personal preference.

1. Build-Tools

- **Webpack ← We learn Webpack**
- Gulp
- Browserify
- NPM scripts
- Grunt

2. Scaffolding

?

2. Scaffolding

2. Scaffolding

- Scaffolding: Building an app with a **shared preset** of configuration.
- Other people may build their apps with the same preset as you

2. Scaffolding

- Scaffolding: Building an app with a **shared preset** of configuration.
- Other people may build their apps with the same preset as you

Neutrino is Scaffolding for Webpack.

3. ECMA Script 2015/2016

- ECMA-Script 2016 is JavaScript with new features shown by new syntaxes

3. ECMA Script 2015/2016

- **Arrow Functions**, mostly used for anonymous callback functions
- Reduce verbosity
- Not used for function constructors

```
sayWhat( function() { return ‘what‘; } );
=
sayWhat( ( ) => { return ‘what‘; } );
```

3. ECMA Script 2015/2016

Arrow Functions can also have names

```
var x = () => { return ‘hallo world‘; }
```

=

```
var x = function() { return ‘hallo world‘; }
```

3. ECMA Script 2015/2016

Importing other frontend modules

import halloWorld from ‘./halloworld.js‘;

⇒ Since we Webpack-Bundling, we avoid to include by <script> Tag.

⇒ Webpack will generate the HTML file with one <script src=“bundle.js“> for us.

3. ECMA Script 2015/2016

Exporting frontend modules

```
export default function halloWorld() {
   $(‘body‘). html(‘hallo world‘);
}
```

(**default** means one function per module)

3. ECMA Script 2015/2016

const: Illegal to reassign identifier, legal to mutate

```
const x = 1;
x = 2;

const obj = {};
obj.x = 1;
obj.y = 2;
obj = {hallo: ‘world‘};
```

3. ECMA Script 2015/2016

var and let: Legal to reassign variable and to mutate

```
var x = 1;
x = 2;

var obj = {};
obj.x = 1;
obj.y = 2;
obj = {hallo: ‘world‘};
```

3. ECMA Script 2015/2016

var and let

```
for(var i=0;i<10;i++) {
    // i exists here
}
// i exists here
```

3. ECMA Script 2015/2016

var and let

```
for(let i=0;i<10;i++) {
    // i exists here
}
// i does NOT exists here
```

3. ECMA Script 2015/2016

Destructuring

```
var obj = {x:1, y:2}
var { x } = obj;
```

=

```
var obj = {x:1,y:2}
var x = obj.x;
```

4. Cross Origin Resource Sharing

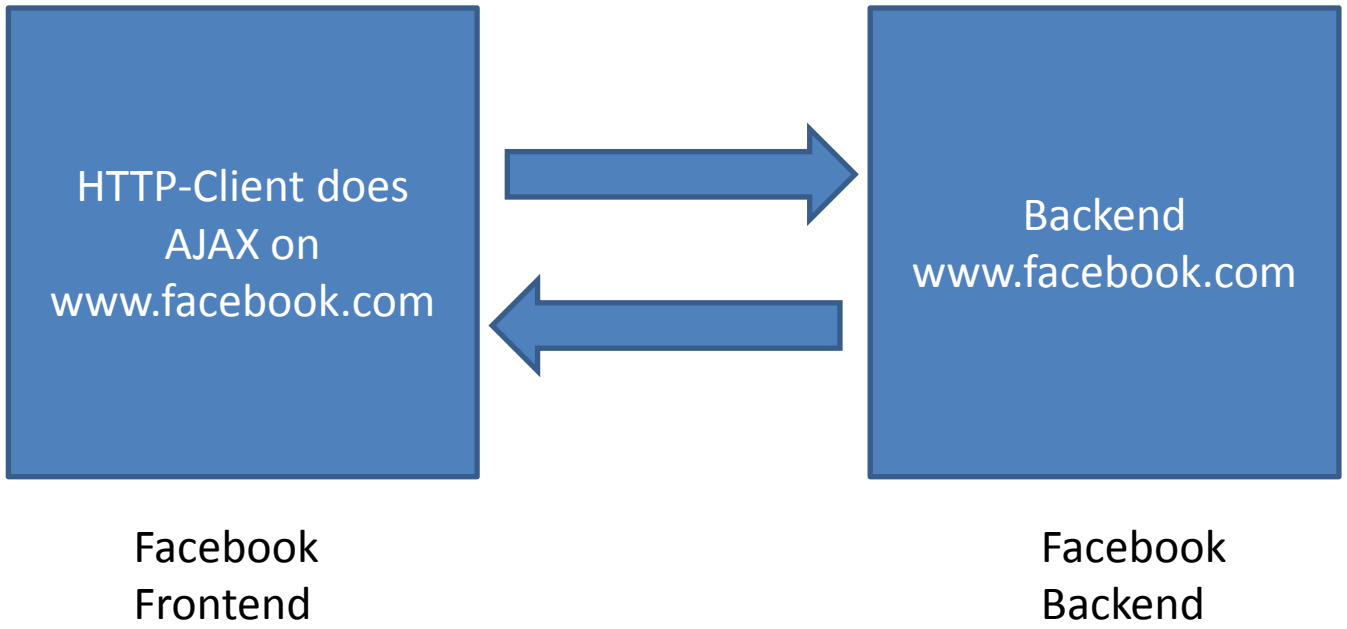
- Cross Origin Response Headers (CORS) are part of the HTTP – response of a server.
- It allows responding to HTTP-Clients which are outside of the current domain.
- Examples for domains?

4. Cross Origin Resource Sharing

- Cross Origin Response Headers (CORS) are part of the HTTP – response of a server.
- It allows responding to HTTP-Server which are outside of the current domain.
- Examples for domains?
  - youtube.com, google.com, facebook.com, localhost
- Example for subdomains?
  - **WWW**.youtube.com, **API**.youtube.com, **API**.facebook.com

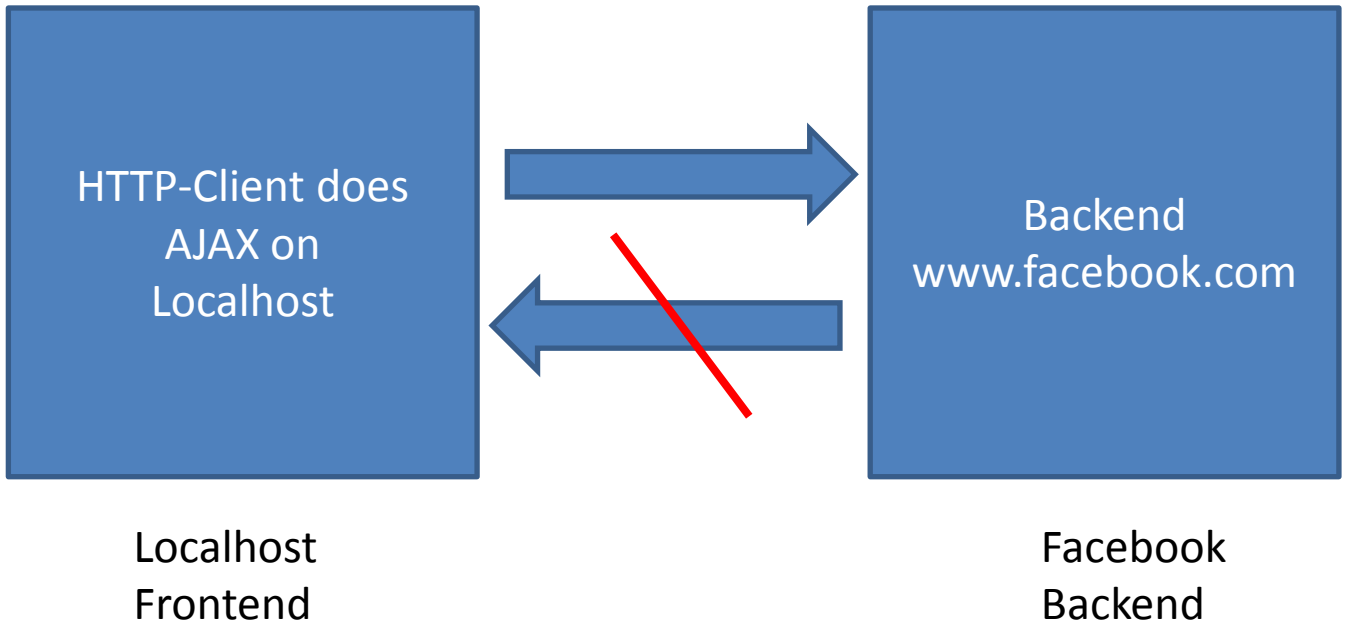
4. Cross Origin Resource Sharing

- CORS prevents servers from sending data to unauthorized HTTP-Clients, i.e.



4. Cross Origin Resource Sharing

- CORS prevents servers from sending data to unauthorized HTTP-Clients, i.e.
- **Checked by our browser**

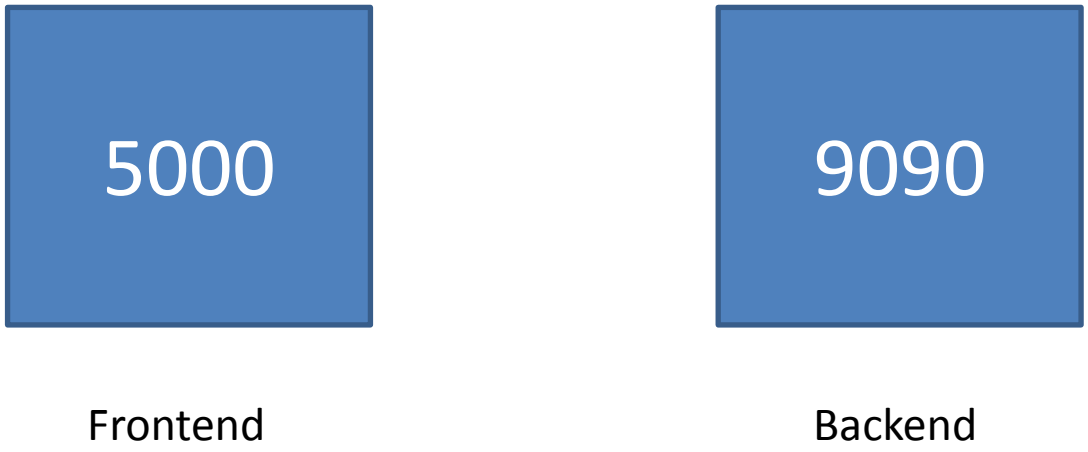


4. Cross Origin Resource Sharing

- Domains can also be separeted by **ports**
- In our case we will develop on a
  - frontend with port 5000
  - Backend with port 9090
- Therefore, we will enable all CORS
- When we deliver our Devugees-Shop, we will only have one port 80

5. Online-Shop

- Let‘s integrate Express ...



5. Online-Shop

Task:

1. Implement database access for the routes
   /products and /categories to our online shop.

2. Use the database to create a new category “Smart Phone“ and add 6 new products:
   - Samsumg Galaxy S6, S7, S8
   - Iphone 10, 11 and 12

3. Find a new picture and integrate it.

5. Online-Shop

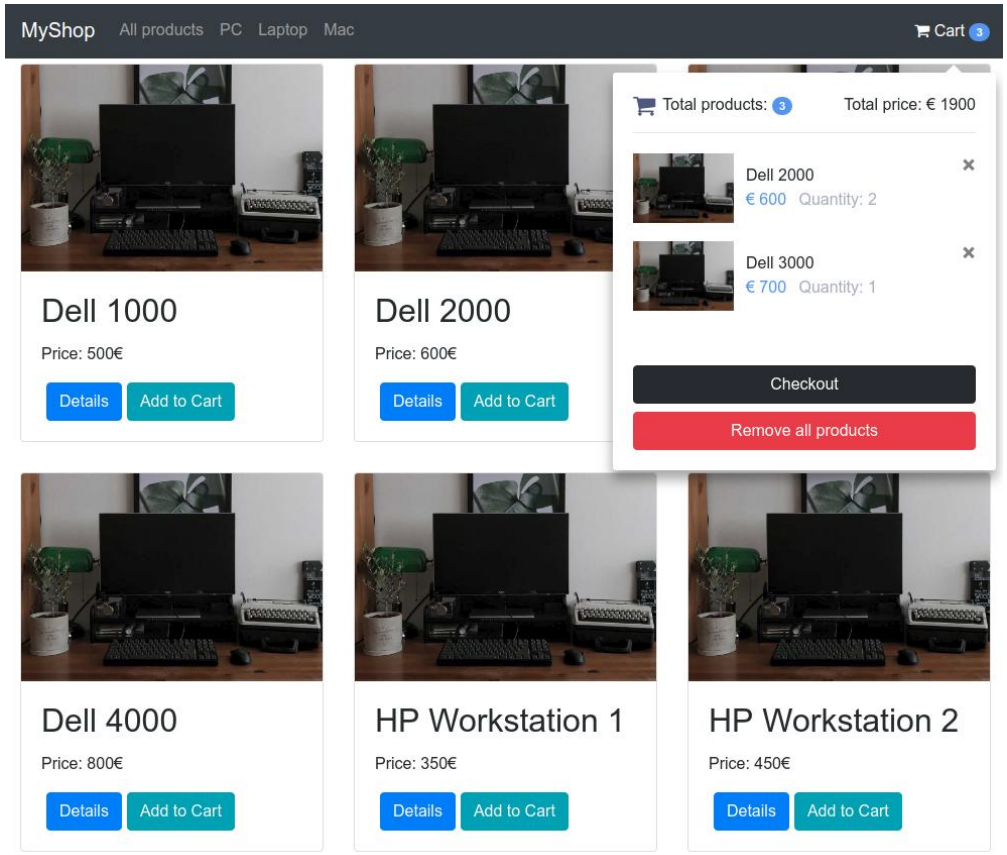
Task:

4. Until we implement signup and login, we will load a testuser from the database into our localStorage.

   Therefore, randomly select one user from the database when the document loads and put the data into the localStorage.

5. Online-Shop

Order-Process

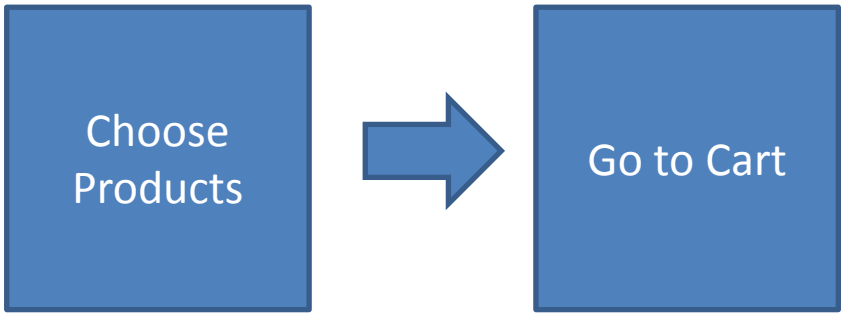


5. Online-Shop

Order-Process

5. Online-Shop

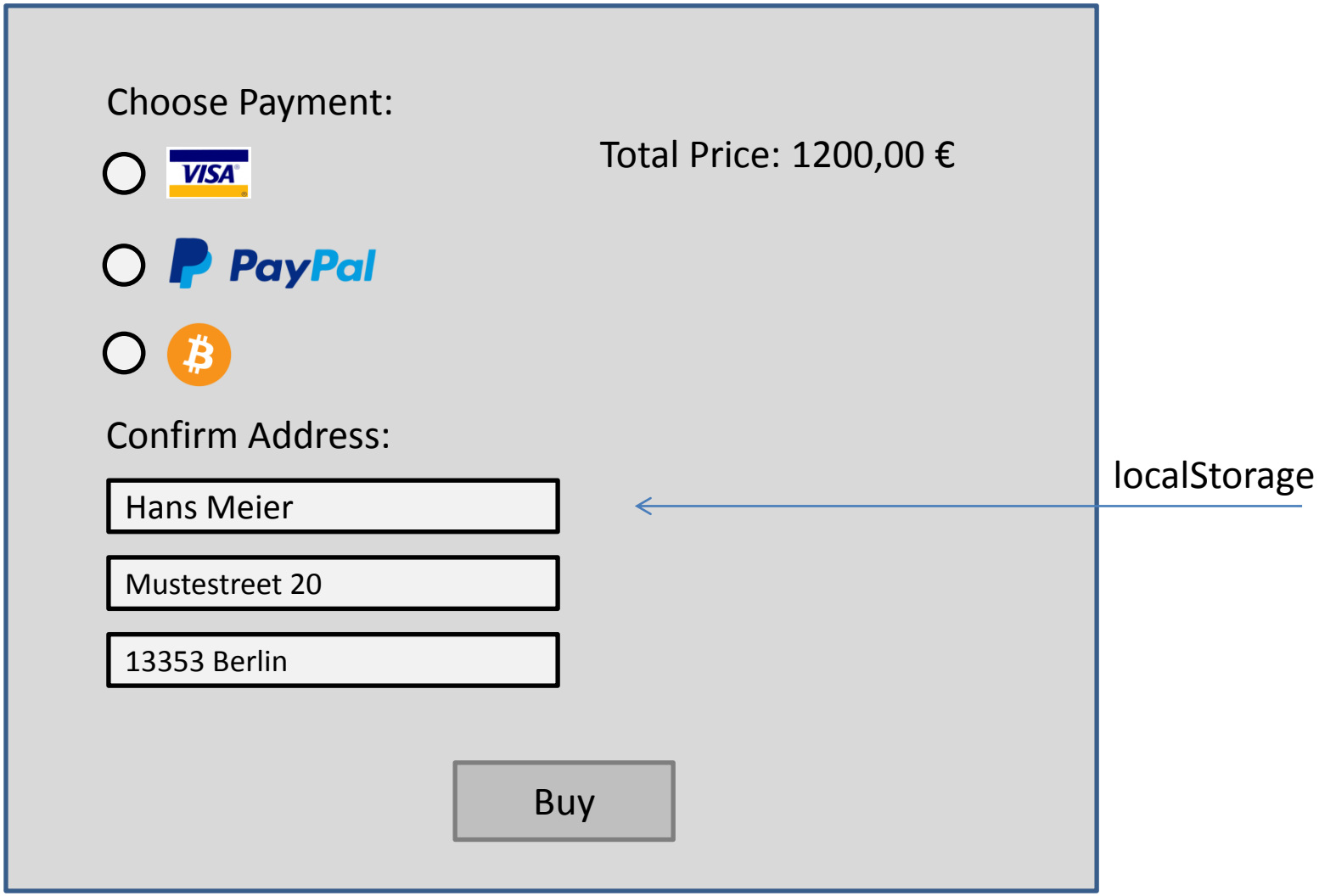
Order-Process



5. Online-Shop

Order-Process

5. Online-Shop

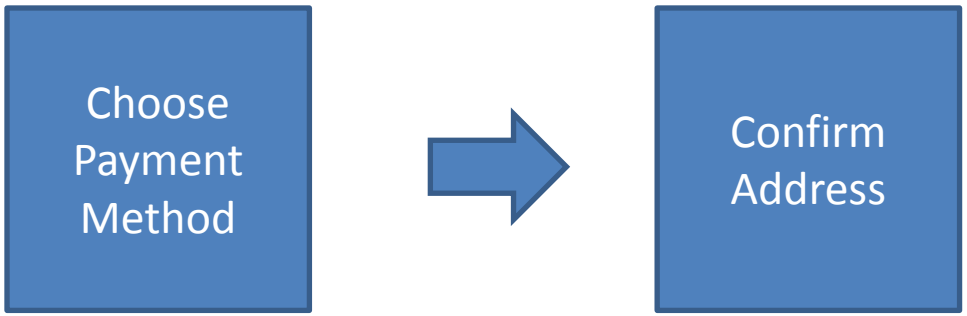


5. Online-Shop

Checkout-Process

5. Online-Shop

Checkout-Process



5. Online-Shop

Checkout-Process

5. Online-Shop

Task:

1. Create a new table “payment_method“. What fields do we need? What is the relation to the order table?

   Insert four payment methods VISA, SEPA, Bitcoin and PayPal.

5. Online-Shop

Task:

2. Create a new POST route /order that creates a new order. What tables are affected as well? Think of the SQL-statements carefully.