Web Engineering & Design 1

AJAX

Client Template Engine

Tobias Blaser

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Content

- Handlebars Basics
- Handlebars Expression
- AJAX, JSON & Handlebars
- AJAX & JSON jQuery/XHR
- Extend Handlebars

Learning Goals

You are able to …

- … validate and create a Handlebars template to render data.
- … compile and embed a Handlebars template.
- … to render and embed JSON data by Handlebars loaded by AJAX native or jQuery.

Client Template Engine

Motivation

- **Separate** user interface and logic
- Most **modern frameworks** use template engines
- Faster and more **lightweight** web applications

Handlebars Basic Example

Demo

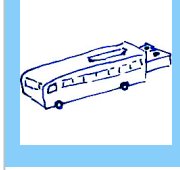
Sky Webshop

Products

T-Shirt airplane "A380"

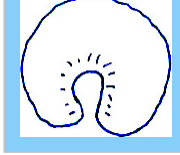
CHF 39.90

Memory-Stick bus "L5"



sale **CHF 9.90**

Bolster "dreams"



CHF 14.90

examples/webshop/

```
<h1>Sky Webshop</h1>
<h2>Products</h2>
<ul id="products"></ul>
```

Demo

Product list container

```
<script id="product-template" type="text/x-handlebars-template">
{{#each products}}
<li>
        <h3>{{name}}</h3>
        <img src="../images/{{image}}" alt="{{name}}" />
        <span class="price">
                {{#if specialOffer}}
                        <span>sale</span>
                {{/if}}
                CHF {{price}}
        </span>
</li>
{{/each}}
</script>
```

Handlebars product template

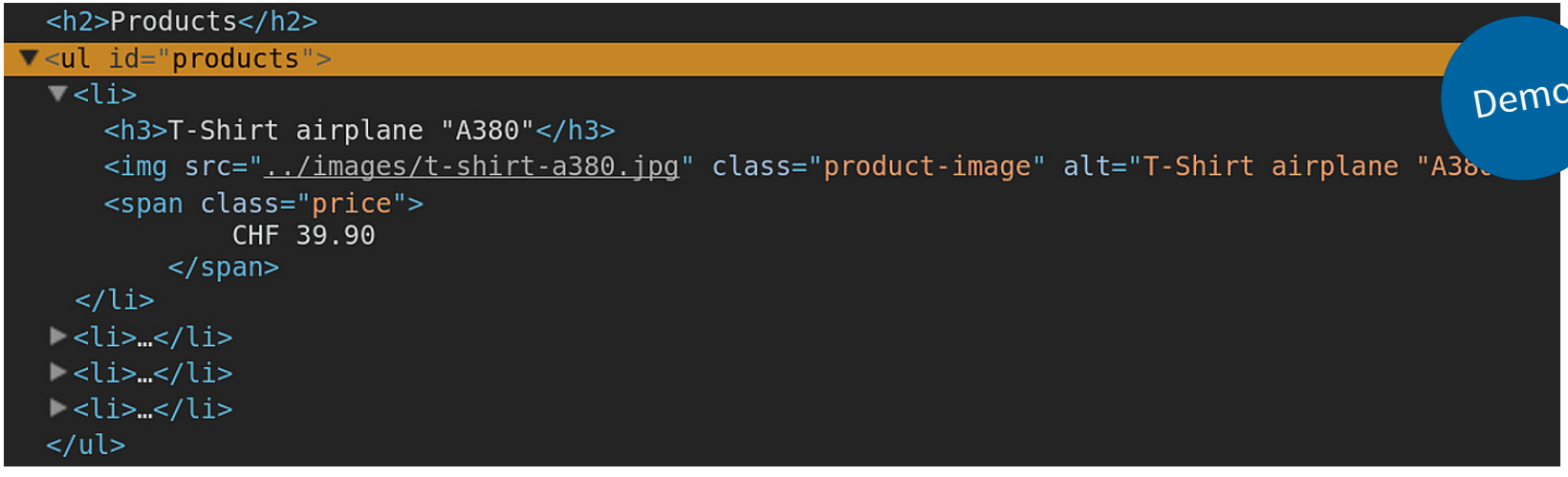
```
var products = [
        {
                name: 'T-Shirt airplane "A380"',
                image: 't-shirt-a380.jpg',
                price: '39.90'
        },{
                name: 'Memory-Stick bus "L5"',
                image: 'memory-stick-15.jpg',
                price: '9.90',
                specialOffer: true
        },
        …
];
```

Demo

Product data

```
var template = document.getElementById('product-template').innerHTML;
var productList = document.getElementById('products');

var viewRenderer = Handlebars.compile(template);
var view = viewRenderer({ products: products });

productList.innerHTML = view;
```

Render & embed template



DOM after list item insertion

Handlebars

```
<!-- Inline template (script tag) -->
<script id="prod…" type="text/x-handlebars-template"> … </script>
```

```
<!-- NOT YET WORKING! HTML5-template tag in head or body) -->
<template id="product-template"> … </template>
```

- Compile template

```
var viewRenderer = Handlebars.compile('<li>template</li>');
```

- Render template & data

```
var view = viewRenderer({ products: products });
```

- Append rendered template

```
element.innerHTML = view;
```

➡ Data **changed** → render template **again**

Handlebars Template Expressions

```
<!-- print variable content -->
{{variable}}
{{variable.property.subproperty}}

{{customHelper variable}}

    {{~variable}}                    <!-- remove whitespace -->
```

```
<!-- loop block helper -->
{{#each products}}
    <strong>{{this}}</strong>     <!-- this: current item -->
    {{name}}                     <!-- this.name -->
    {{../variable}}              <!-- variable of outer scope -->
{{/each}}

{{#each products as |product productId|}}
    <p>
        <strong>{{productId}}</strong>:
        {{product.name}}
    </p>
{{/each}}
```

```
<!-- if block helper -->
{{#if variable.isItTrue}}
        <p>it is true</p>
<!-- if & else helper chained -->
{{else if variable.alreadyNotSet}}
        <p>it is unknown</p>
{{else}}
        <p>it is false</p>
{{/if}}
```

```
<!-- with block helper -->
{{#with variable}}
        <!-- variable.name, variable.address -->
        <p>{{name}}, {{address}}</p>
{{/if}}
```

```
<!-- raw helper -->
{{{{raw-helper}}}}
    {{variable}}          <!-- will print '    {{variable}}' -->
{{{{/raw-helper}}}}
```

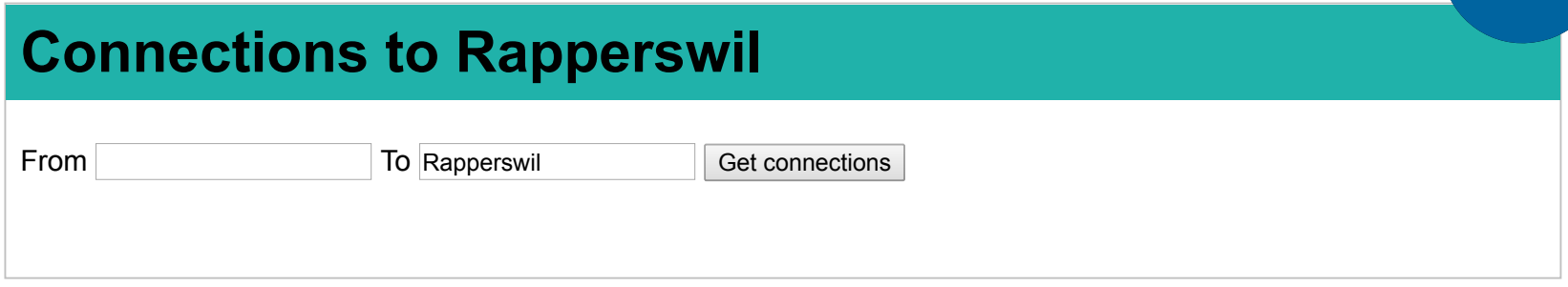
Handlebars expressions: http://handlebarsjs.com/expressions.html
Handlebars block helpers: http://handlebarsjs.com/block_helpers.html

Handlebars limitations

- Data **changed** → developer needs to **watch and apply** data changes
- **No event-registration** feature. Manual solutions:
  - Re-register every event listener after template update
  - **Listen on template container** → bubble and handle cases
  - Add inline-event listeners in template (e.g. onclick="…")
- **Not many built-in helpers**. E.g. no comparison helper

AJAX, JSON & Handlebars

Demo



examples/autocompletion/client/

Suggestion API:

```
http://localhost:8070/suggestions?search=Rapp
```

```
{"suggestions":[
        "Rapperswil",
        "Rapperswil, ZSG",
        "Rapperswil, Bahnhof",
        "Rapperswil, Sonnenhof"
]}
```

localhost:8070/
localhost:8070/suggestions?search=Rapp

AJAX / Handlebars Demo

Demo

```
{{#each suggestions}}
<option value="{{this}}">{{this}}</option>
{{/each}}
```

suggestions.html template

```
$.get('suggestions.html', function(suggestionsTemplate){
    …
});
```

Template loading

```
<datalist id="suggestions"></datalist>
<form>
    <label>
        From
        <input type="text" id="from" ... list="suggestions" />
    </label>
        …
</form>
```

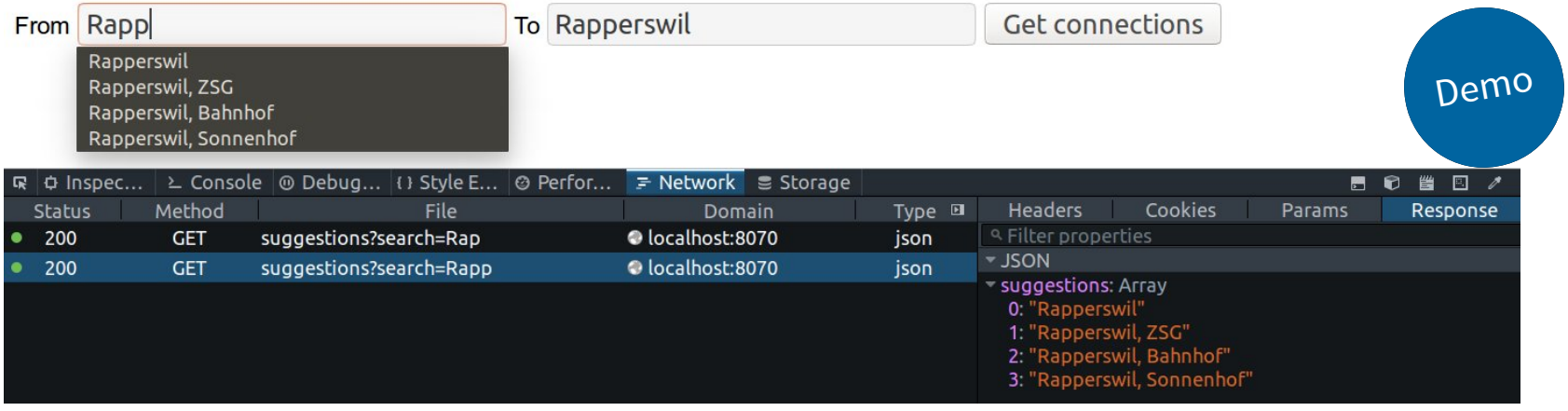
Application HTML snippet

```
var suggestionAPI = 'http://localhost:8070/suggestions?search='
var fromField = $('#from');
var suggestionListElement = $('#suggestions');

$.get('suggestions.html', function(suggestionsTemplate) {
    // compile suggestion template
    var viewRenderer = Handlebars.compile(suggestionsTemplate);

    fromField.on('input', function() {
        if(fromField.val().length >= 3) {
            var url = suggestionAPI+fromField.val();
```

Demo

```
            $.getJSON(url, function(suggestionData) {
                // get suggestions list from JSON object
                var suggestions = suggestionData.suggestions;
                // render view from template
                var view = viewRenderer({ suggestions: suggestions });
                suggestionListElement.get(0).innerHTML = view;
            });
```

```
        } else {
             suggestionListElement.empty();
        }
    });
});
```



Suggestion requests

```
<h1>Connections to Rapperswil</h1>
▼<datalist id="suggestions">
    <option value="Rapperswil">Rapperswil</option>
    <option value="Rapperswil, ZSG">Rapperswil, ZSG</option>
    <option value="Rapperswil, Bahnhof">Rapperswil, Bahnhof</option>
    <option value="Rapperswil, Sonnenhof">Rapperswil, Sonnenhof</option>
  </datalist>
▼<form>
  ▶<label>…</label>
  ▶<label>…</label>
    <button type="submit">Get connections</button>
  </form>
```

DOM after data insertion

AJAX & JSON jQuery/XHR

- $.ajax() & JSON

```
$.ajax({ dataType: "json", url: url, data: data }).done(
        function(data, textStatus, jqXHR) { ... }
);
```

- $.getJSON() → based on ajax()

```
$.getJSON(url, function(data) { ... });
```

➡ No postJSON()-shorthand → use ajax()

```
$.ajax({
        type: 'POST',
        url: url,
        data: JSON.stringify(data),
        contentType: "application/json",
        dataType: 'json'
}).done(function(data) { ... });
```

XHR GET-JSON

```
request = new XMLHttpRequest();
request.onreadystatechange = function() {
    if (request.readyState == 4 && request.status == 200) {

        // Parse JSON object from answer
        var data = JSON.parse(request.responseText);
        var suggestions = data['suggestions'];
        // do something with suggestions
    }
};

request.open("GET", url);
request.send();
```

XHR POST-JSON

```
request = new XMLHttpRequest();
request.onreadystatechange = function() {
    if (request.readyState == 4 && request.status == 200) {

        // Parse JSON object from answer
        var data = JSON.parse(request.responseText);
        var suggestions = data['suggestions'];
        ...
    }
};

request.open("POST", url);
// JSON content type header
request.setRequestHeader("Content-Type", "application/json");

request.send(JSON.stringify(data));
```

Extend Handlebars

```
Handlebars.registerHelper('xif', function (v1, operator, v2, options) {
    switch (operator) {
        case '==':
        return (v1 == v2) ? options.fn(this) : options.inverse(this);
        case '===':
        return (v1 === v2) ? options.fn(this) : options.inverse(this);
        case '<':
        return (v1 < v2) ? options.fn(this) : options.inverse(this);
        case '<=':
        return (v1 <= v2) ? options.fn(this) : options.inverse(this);
        ...
        default:
        return options.inverse(this);
    }
});
```

```
{{#xif var1 '==' var2}}

{{/xif}}
```

Source: stackoverflow

Handlebars Partials

```
<ul>
{{#each products}}
        {{> productDetail}}
{{/each}}
</ul>
```

```
<li>
        <strong>{{id}}</strong>: {{name}}
</li>
```

```
var products = [{id: 1, name: 'Memory stick'}, ...]; // data

var productsTmpl = document.getElementById('products-tmpl').innerHTML;
var detailTmpl = document.getElementById('detail-tmpl').innerHTML;
var productsContainer = document.getElementById('products');

// register partial
Handlebars.registerPartial('productDetail', detailTmpl);

var viewRenderer = Handlebars.compile(productsTmpl);
productsContainer.innerHTML = viewRenderer({ products: products });
```

Web Engineering & Design 1

AJAX

Client Template Engine

Tobias Blaser

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Self Check

1. Which steps are necessary to compile and embed a Handlebars template?
2. How do you receive and send JSON data using native AJAX request and jQuery?

3. Create a template which produces the following output from the given data (username is linked to mail):

```
{ user: {
        username: "hansmeyer",
        email: "hansmeyer@gmail.com",
        books: [
                "JavaScript for Dummies", "JavaScript the Good Parts"
        ],
        papers: [
                {name: "ES6 Modules", year: 2015},
                {name: "WebAssembly", year: 2016}
        ]
}}
```

hansmeyer

Books

- JavaScript for Dummies
- JavaScript the Good Parts

Papers

2015. ES6 Modules
2016. WebAssembly

Solutions

1.
```
var viewRenderer = Handlebars.compile(htmlTemplate);
var view = viewRenderer(data);
viewContainer.innerHTML = view;
```

2.
```
// native
request.onreadystatechange = function() {        // receive
        if (request.readyState == 4 && request.status == 200) {
                var data = JSON.parse(request.responseText);
        }
};
request.send(JSON.stringify(data));              // send

// jQuery
$.getJSON(...);                                  // receive
$.post(.., JSON.stringify(data), ...);           // send
```

3. {{#with user}}

```
<h3><a href="mailto:{{ email }}">{{ username }}</a></h3>
<h4>Books</h4>
<ul>
        {{#each books}}
        <li>{{this}}</li>
        {{/each}}
</ul>
<h4>Papers</h4>
<ol>
        {{#each papers}}
        <li value="{{year}}">{{name}}</li>
        {{/each}}
</ol>
{{/with}}
```

Web Engineering & Design 1

AJAX

template engine

Tobias Blaser

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz