Mobile Applications
Development

Handlebars.js

Ivano Malavolta

ivano.malavolta@univaq.it

http://www.di.univaq.it/malavolta

dipartimentoinformatica
Università degli Studi dell'Aquila

Roadmap

- Why Handlebars
- Handlebars Basics
- Usage with Require.JS

Why Handlebars

We are building apps, not web sites

We want to separate presentation from logic

We don't want to put any HTML element into Javascript code

What we want to avoid

```
18
19  function searchCallback(data) {
20      $('#busy').hide();
21      movies = data.movies;
22      $.each(movies, function(index, movie) {
23          $('#moviesList').append('<li><a href="moviedetails.html?id=' + movie.id + '">' +
24          '<img src="' + movie.posters.thumbnail + '" class="list-icon">' +
25          '<h4>' + movie.title + '</h4>' +
26          '<p>' + movie.critics_consensus + '</p>' +
27          '</li></li>');
28      });
29      $('#moviesList').listview('refresh');
30  }
31
```

Imagine yourself trying to change how a movie should be rendered in your app...

Roadmap

- Why Handlebars
- Handlebars Basics
- Usage with Require.JS

Example of Template

```
<div class=“userEntry">
  <h1>
     {{username}}
  </h1>
  <img>
     {{profilePic}}
  </img>
  <div class=“status“>
     {{status}}
  </div>
</div>
```

A handlebars expression is a {{, some contents, followed by a }}

Values Escaping

Handlebars HTML-escapes all the values returned by a {{expression}}

If you don't want Handlebars to escape a value, use the "triple-stash“

```
<div class=“userEntry">
  <h1>{{username}}</h1>
  <div class=“status“>
     {{{status}}}
  </div>
</div>
```

Template Context

A context is a Javascript object used to populate a template

```
var context = {
  username: “Ivano“,
  profilePic: “./images/pic.png“,
  status: “feeling good”
};
```

Compiling a Template

Templates are defined within a <script> tag
or in external files

```
<script id="user-template"
  type="text/x-handlebars-template">

  // template content

</script>
```

Compiling a Template

Handlebars.compile is used to compile a template

```
var source = $("#user-template").html();
var template = Handlebars.compile(source);
```

Compiling = obtaining a JS object representing the template

Obtaining the final HTML code

You have to execute a template with a **context** in order to get its corresponding HTML code

```
var context = {username: “Ivano“,
               status: “feeling good” };
```

```
var html = template(context);
```

```
<div class=“userEntry">
  <h1>Ivano<h1>
  <div class=“status“>
    feeling good
  </div>
</div>
```

Expressions

The simplest expression is a simple identifier

```
<h1>{{username}}</h1>
```

This expression means

"look up the title property in the current context"

Expressions

Handlebars expressions can also be dot-separated paths

```
<h1>{{user.username}}</h1>
```

This expression means

"look up the *user* property in the current context, then look up the *username* property in the result"

Helpers

Helpers are Javascript functions that return HTML code

```
Handlebars.registerHelper(`test`, function(user) {
    return new Handlebars.SafeString(
       "<a href='" + user.name + "'>" +
            user.surname + "</a>"
   );
});
```

You should return a Handlebars SafeString if you don't want it to be escaped by default

Calling Helpers

A Handlebars helper call is a simple identifier, followed by zero or more parameters

Each parameter is a Handlebars expression

es.

```
{{{ test user }}}
```

In this case, *link* is the name of a Handlebars helper, and *user* is a parameter to the helper

Built-in Helpers

With

It shifts the context for a section of a template

```
{ title: "My first post!",
  author: { firstName: "Charles", lastName: "Jolley" }
}
```

```
<div class="entry“>
<h1>{{title}}</h1>
{{#with author}}
<h2>By {{firstName}} {{lastName}}</h2>
{{/with}}
</div>
```

```
<div class="entry“>
<h1>My first post!</h1>
<h2>By Charles Jolley</h2>
</div>
```

Built-in Helpers

Each

To iterate over a list

Inside the block, you can use *this* to reference the element being iterated

```
{ people: [ "Yehuda Katz", "Alan Johnson", "Charles Jolley" ] }
```

```
<ul class="people_list">
 {{#each people}}
 <li>{{this}}</li>
{{/each}}
 </ul>
```

```
<ul class="people_list">
  <li>Yehuda Katz</li>
  <li>Alan Johnson</li>
  <li>Charles Jolley</li>
</ul>
```

Built-in Helpers

If - Else

To conditionally render a block

It will render the block if its argument is not equal to
false, undefined, null, []

```
<div class="entry">
{{#if author}}
<h1>{{firstName}} {{lastName}}</h1>
{{else}}
<h1>Unknown Author</h1>
{{/if}}
</div>
```

The unless helper is the inverse of if

Handlebars Recall

Each Template can contains Expressions and Helpers operating on them

You can define your own Helpers that operate on expressions, they return HTML code

A template can be (pre)-compiled and must be executed with a Context in order to return the final HTML fragment

Roadmap

- Why Handlebars
- Handlebars Basics
- Usage with Require.JS

Usage with Require.JS

There is a RequireJS plugin that allows you to

- Automatically precompile all your templates
- Manage templates dependencies with RequireJS
- Refer to templates directly within your JS code

Reference:
https://github.com/SlexAxton/require-handlebars-plugin

Library Usage

Include the *hbs.js* plugin and the Handlebars.js file in the same directory of your *require.js*

```
www/js/lib/require.js
www/js/lib/hbs.js
www/js/lib/Handlebars.js
www/templates/Hi.hbs
```

Template Definition

```
// in file yourApp/templates/Hi.hbs
<div class=“test">
  Hi, my name is {{ name }}, nice to meet you!
</div>
```

Template Execution

In your JS files now you can require and execute your templates

es.

```
require(['hbs!../templates/Hi'], function ( tmplHi ) {
   $('#block').html(tmplHi({name: "Ivano"}));
 });
```

```
<div class="test">
   Hi, my name is Ivano, nice to meet you!
</div>
```

References

handlebarsjs.com