UNIT 1

JavaScript

Part 4 - Annex Handlebars.js

Client-side Web Development
2nd course – DAW
IES San Vicente 2020/2021
Author: Arturo Bernal Mayordomo

Index

Handlebars.js

Handlebars.js is a JavaScript library (based on Mustache) used to generate HTML automatically from predefined templates and JSON data (received, for example, from a web service).

A template is nothing more than HTML code with some expressions inside double curly braces {{}} which will be evaluated and many of them replaced with values from a JSON object (others will be control statements).

We can install Handlebars with NPM in our project as a production dependency:

```
npm i handlebars -S
```

If we're using Webpack, we'll also need to install its loader:

```
npm i -D handlebars-loader
```

Let's see how to create basic templates and use them with **Webpack** and the **handlebars loader**.

Value replacement

A Handlebars template is a normal HTML structure which will be processed by the Handlebars compiler. The most basic functionality is value replacement. This is an example of a template with 2 values that will be generated from a JSON object:

```
<p>{{name}}</p>
<p>{{age}}</p>
```

And this is the JSON object with the corresponding values:

```
{
    name: "Pepe",
    age : 43
}
```

When we pass the JSON object to the template it will generate this HTML:

```
<p>Pepe</p>
<p>43</p>
```

We can also use nested properties:

```
<p>{{name}}</p>
<p>{{age}}</p>
<p>{{address.street}}, {{adress.city}}</p>
```

```
JSON Object

{
    "name": "Pepe",
    "age" : 43,
    "address": {
        "street": "Calle Ancora 15",
        "city"  : "Madrid"
    }
}
```

```
Result:

<p>Pepe</p>
<p>43</p>
<p>Calle Ancora 15, Madrid</p>
```

Conditional expressions (if, unless)

In a template, we can use conditional expressions like **if..else**. It can be used in a limited way because it can't evaluate expressions, so we just verify if a value exists or not (or equivalent to false → null, 0, ‘’).

```
<p>{{name}}</p>
<p>{{age}}</p>
{{#if address}}
  <p>{{address.street}}, {{adress.city}}</p>
{{else}}
  <p>No address!</p>
{{/if}}
```

Instead of the **if** clause we can use **unless** which does the opposite (prints contents when the value does not exist or is equivalent to false).

The with block

Using {{**#with property**}}, we can access nested properties (inside this block) without the need of using this property as a prefix.

```
<p>{{name}}</p>
<p>{{age}}</p>
{{#with address}}
  <p>{{street}}, {{city}}</p>
{{/with}}
```

Iterators (each)

If we need to iterate through a property that is an array, we'll use the each block, which will generate as many HTML blocks as positions this array has. Inside the block we'll access the properties of each object inside the array.

```
{
    "name": "Pepe",
    "age"  : 43,
    "addresses": [
        {
            "street": "Calle Ancora 15",
            "city": "Madrid"
        }, {
            "street": "Calle Tudela 12",
            "city": "Valladolid"
        }
    ]
}
```

```
<p>{{name}}</p>
<p>{{age}}</p>
{{#each addresses}}
  <p>{{street}}, {{city}}</p>
{{/each}}
```

```
Result:
```

```
<p>Pepe</p>
<p>43</p>
  <p>Calle Ancora 15, Madrid</p>
  <p>Calle Tudela 12, Valladolid</p>
```

If the array contains values (strings, integers, other arrays) and not objects, we can reference the actual value using the **this** property, or even use properties like **@index** which will contain the current position:

```
{
    "name"   : "Pepe",
    "age"    : 43,
    "phones": [92485325,4353246,69496334]
}
```

```
<p>{{name}}</p>
<p>{{age}}</p>
{{#each phones}}
  <p>Phone{{@index}}: {{this}}</p>
{{/each}}
```

```
Result:
```

```
<p>Pepe</p>
<p>43</p>
  <p>Phone 0: 92485325</p>
  <p>Phone 1: 4353246</p>
  <p>Phone 2: 69496334</p>
```

HTML Generation (Webpack)

If we're using Webpack with the Handlebars loader is very simple to use a template inside our JavaScript file. First of all, we'll need to import the templates we're going to need:

```
import productsTemplate from '../templates/products.handlebars';
```

And generate the corresponding HTML like this:

```
let htmlProds = productsTemplate(prodsJSON);
```

As the generated HTML will be in a string format and not actual DOM objects, we have to use the innerHTML property and put it inside a container.

```
var container = document.getElementById("container");
container.innerHTML = htmlProds;
```