codecademy

Templating With Handlebars

`Handlebars.compile()`

`Handlebar.compile()` can be used to produce a templating function. A template string with expressions must be passed into `Handlebar.compile()`. That function then takes an object as an argument, interpolates the object's values into the template expressions, and returns a completed HTML string.

```
const template = '<span>{{greetingMsg}}
</span>';
const templateFunction
= Handlebars.compile(template);
const html = templateFunction({
greetingMsg: 'Greetings from the club!'
});
console.log(html); // <span>Greetings
from the club!</span>
```

Handlebars `{{each}}` block helper

Handlebars `{{each}}` block helper is a built-in helper expression that can accept an array to iterate through. Inside an `{{each}}` block, `this` serves as a placeholder for the current iteration value.

```
const template = `<ul>
{{#each serialList}}
  <li>{{this}}</li>
{{/each}}
</ul>`;

const templateFunction
= Handlebars.compile(template);
const htmlStr = templateFunction({
serialList: [202, 204, 338, 342, 400]
});
console.log(completedHtml);
/* Output:
<ul>
  <li>202</li>
  <li>204</li>
  <li>338</li>
  <li>342</li>
  <li>400</li>
</ul>
*/
```

Handlebars block helpers

Handlebars comes with built-in block helpers which allow us to embed HTML or other expressions in between helper expression tags.

The starting expression of the block helper will have a `#` that appears before a keyword, and the ending expression will have a `/` followed by the same keyword to denote the end.

```
{{#blockName}}
  Block text content
  <p>An HTML paragraph element</p>
{{/blockName}}
```

The Handlebars.js JavaScript Library

Handlebars.js is a Javascript library used to create reusable webpage templates. The templates are combination of HTML, text, and expressions. The expressions are included in the html document and surrounded by double curly braces.
These expressions are placeholders for content to be inserted by our Handlebars.js code in our js document. When the compiled templated function is called, values are substituted into the expressions.

```
<script id="spaceCraft" type="text/x-
handlebars-template">
    <p>{{spacecraftName}} landed the
first human on the Moon.</p>
</script>
```

Handlebars.js and the `<script>` Element

An HTML `<script>` element with `type` value of `text/x-handelbars-template` can be used to contain Handlebars.js template text and expressions. This allows writing Handlebars.js templates in an HTML document.

```
<script id="handlebars-template"
type="text/x-handlebars-template">
  <p>Hello {{loggedInUser}}</p>
</script>
```

Handlebars `{{if}}` block helper

The Handlebars `{{if}}` block helper is one of the built-in helpers that will conditionally render a block of code. The given block of code contains an example of the usage of it.

```
const template = `<h1>
{{#if quotaFull}}
  Please come back tomorrow.
{{/if}}
</h1>`;

const templateFunction
= Handlebars.compile(template);
const completedHtml
= templateFunction({ quotaFull: true
});
console.log(completedHtml); //
<h1>Please come back tomorrow.</h1>
```

The Handlebars `{{else}}` expression

The Handlebars `{{else}}` expression can be inserted into an `{{if}}` block helper. Template contents inside the else section comes into play when the previous condition(s) are falsy. In the given example, `isAdult` is set to `false`. Hence, `You can enter the ride.` will not be in the returned template string.

```
const template = `<span>
{{#if isAdult}}
  You can enter the ride.
{{else}}
  A guardian must accompany you.
{{/if}}
</span>`;

const templateFunction
= Handlebars.compile(template);
const htmlStr = templateFunction({
isAdult: false });
console.log(htmlStr); // <span>A
guardian must accompany you.</span>
```