thinkvitamin.com

Getting Started with Handlebars.js

Why handlebars.js?

I have to disclose that I'm a little biased – I worked with Yehuda Katz on Handlebars.js. We wrote Handlebars because we loved Mustache's approach to "logic-less templating" in general but had a rough time dealing with the hoops you had to jump through to use global helpers and the lack of support for accessing variables further up the template's call stack. We also really wanted templates that could be precompiled instead of having to be compiled on the client and really wanted to write the fastest templating language possible. Although we didn't end up with the absolute fastest templating framework for JavaScript, Handlebars.js is lightning fast and accomplished our other goals.

Installation and Usage

The easiest way to install Handlebars.js is to download the latest build from the GitHub project. We're not quite to a 1.0 release yet, but Handlebars.js is being actively used by quite a few projects. Handlebars is just a JavaScript library, so you include it in your pages the same way you would any other script:

```
<script type="text/javascript"
    src="/scripts/handlebars-0.9.0.pre.4.js" />
```

For basic templating, you may want to just include your template inline in the document. You can use a script tag with a custom type to hold it:

```
<script id="some-template" type="text/x-handlebars-template">
  <table>
    <thead>
      <th>Username</th>
      <th>Real Name</th>
      <th>Email</th>
    </thead>
    <tbody>
      {{#users}}
        <tr>
          <td>{{username}}</td>
          <td>{{firstName}} {{lastName}}</td>
          <td>{{email}}</td>
        </tr>
      {{/users}}
    </tbody>
  </table>
</script>
```

Then you can compile, process, and display that template with the following code:

```
var source   = $("#some-template").html();
var template = Handlebars.compile(source);
var data = { users: [
    {username: "alan", firstName: "Alan", lastName: "Johnson", ema
    {username: "allison", firstName: "Allison", lastName: "House",
    {username: "ryan", firstName: "Ryan", lastName: "Carson", emai
  ]};
$("#content-placeholder").html(template(data));
```

I'm using jQuery for inserting the template output above, but Handlebars will work with any framework that you'd like to use it with. One thing to note is that

Handlebars always compiles templates into a JavaScript function. That makes them super easy to work with.

Basic Expressions

The simplest dynamic element in a Handlebars template is an expression. An expression is surrounded by handlebars, like `{{expression}}`. When an expression is reached in the template, Handlebars will look for an item in the current context that matches the expression given. If the matching item is a value, the value is output. If the matching item is a function, the function is called. If no matching item is found, nothing is written to the output. Expressions support using the dot (`.`) operator in expressions to output nested values. For example, `{{user.firstName}}` would output the firstName property on the user value in the current context.

By default Handlebars escapes the results of expressions, but using a “triple-stash”, like `{{{expression}}}`, will cause the expression to be output unescaped.

Blocks

Sometimes it’s helpful to focus your work on a particular expression within a template. That’s where blocks come in. Blocks are represented in Handlebars with the pound (#) symbol followed by an expression. Blocks end with a closing mustache, `{{/expression}}`.

If the expression given evaluates to an Array, Handlebars will iterate over each item in the Array, setting the current context to that item. Here’s an example:

```
var data = { people: [
    {name: "Alan"},
    {name: "Allison"},
    {name: "Ryan"}
  ], group: "Bloggers" };
```

```
<script type="text/x-handlebars-template">
  <ul>
    {{#people}}
      <li>{{name}}</li>
    {{/people}}
  </ul>
</script>
```

Because blocks change the current expression context, Handlebars supports using the `../` expression to access parent contexts. So in the previous example, we could have used the expression `../group` while iterating over each of the people to print out the name of the group:

```
<script type="text/x-handlebars-template">
  <ul>
    {{#people}}
      <li>{{name}} - {{../group}}</li>
    {{/people}}
  </ul>
</script>
```

If a block's expression evaluates to anything other than an Array, Handlebars simply sets the context to the result of evaluating the expression. This can save a lot of typing when outputting several properties of an object:

```
var data = { person: {
    firstName: "Alan",
    lastName: "Johnson",
    email: "alan@test.com",
    phone: "123-456-7890"
  } };
```

```
<script type="text/x-handlebars-template">
  {{#person}}
    <div>Name: {{firstName}} {{lastName}}</div>
    <div>Email: {{email}}</div>
    <div>Phone: {{phone}}</div>
  {{/person}}
</script>
```

What's Next?

There's a ton more to cover, so I'll be posting about advanced Handlebars.js techniques next week. We'll talk about partials, block helpers, global helpers, and how to precompile your templates so that they don't have to be compiled on the client.

Follow @thinkvitamin on Twitter Please check out Think Vitamin Membership

Other Posts You Might Find Interesting

- Sorry - No Related Posts Found