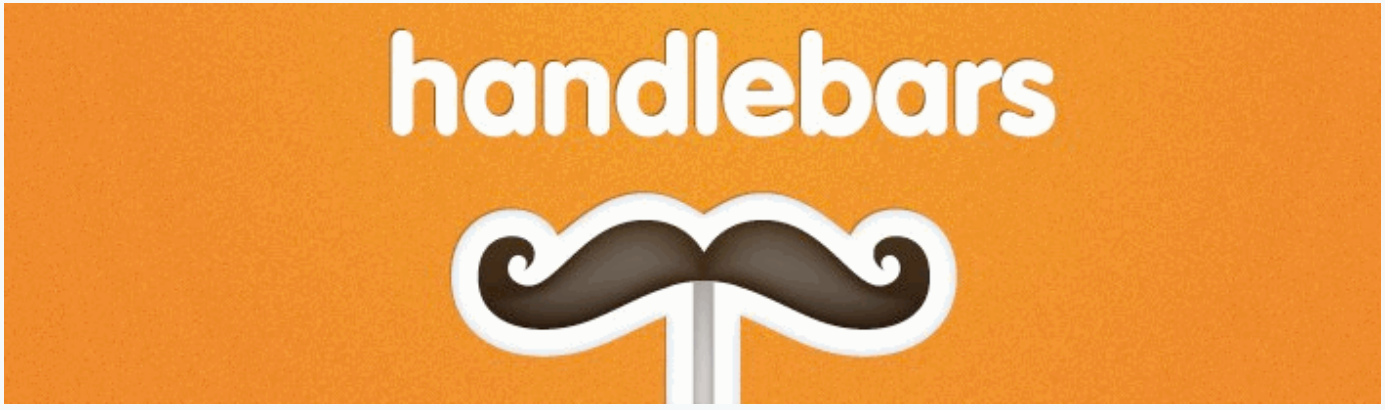
Learn

Getting Started with Handlebars.js

Treehouse

writes on March 25, 2011



Web apps are using JavaScript to create dynamic interfaces now more than ever before, and that's not a trend that will change any time soon. DOM manipulation is great for simpler JavaScript apps, but what do you do when you're changing huge chunks of the document with each change of the view? That's where

JavaScript templating comes into play.

There are quite a few amazing JavaScript templating libraries available. I started out with JavaScript templating using mustache.js, a JavaScript port of the excellent Mustache templating language and moved on to a stint using John Resig's JavaScript Micro-Templating. jQuery has its official templating plugin, and so does Underscore.js. Even 37signals has a JavaScript templating language, although it's for CoffeeScript, called eco. My personal favorite JavaScript templating language these days is 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", emai
    {username: "allison", firstName: "Allison", lastName: "House",
    {username: "ryan", firstName: "Ryan", lastName: "Carson", email
  ]};
$("#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.

code | html and xhtml | javascript

24 Responses to "Getting Started with Handlebars.js"

in chennai|health spas chennai|pet spas in chennai|hair spas in chennai|best spas of chennai|spas and salons chennai|top 10 spas in chennai|salons and spas in chennai|top 5 spas in chennai|thai spas in chennai|best spas in chennai|24 hours massage service on September 16, 2015 at 9:14 pm said:

I've been exploring for a little for any high quality articles or weblog posts in this kind of area . Exploring in Yahoo I at last stumbled upon this website. Reading this information So i'm satisfied to convey that I've a very good uncanny feeling I found out just what I needed. I such a lot for sure will make certain to do not disregard this web site and provides it a glance on a continuing basis.

Reply

test on October 4, 2013 at 12:49 am said:

hi

Reply

Alessandro on June 13, 2013 at 7:36 am said:

Great article I have just started playing around with Handlebars.js, although I haven't used it in a real project I can see the benefits of using

it. Thank you

Reply

Amol Neurgaonkar on May 2, 2011 at 6:56 am said:

Here`s a demo app using Backbone.js with Handlebars incase anyone interested – https://github.com/amoln/backbone.js-with-templates-demo

Reply

Fernando Silva on April 5, 2011 at 4:51 am said:

I think that we need a urgente demo! Please!

Reply

Barney Carroll on April 3, 2011 at 11:07 am said:

Hiya Alan, very bold and commendable of you to pitch in to the JS templating fray!

Interested that you've previously used such a broad variety of the existing templating languages (but not one in particular, as I'll go into later). I've used Resig's original tmpl and the Microsoft-developed official jQuery plugin of the same name and was satisfied in different ways by both — I want a middle ground really. I came here because, having been convinced

by Kyle Simpson's post [1] on what was wrong with JS templating as it stood, I'd decided I would use his [2] next time I started a project from scratch that demanded a framework of front-end templates. Only problem was a lack of in-the-wild examples or walkthroughs, so when my Google search found this I thought AWZM. I'm sure your framework's name having the same purpose and name bar one letter is a coincidence, but it might be worth rethinking before it develops any more traction under the former — I can well imagine the confusion I experienced becoming endemic if it isn't!

But about your lib: the relative path-fixing is cool and general syntax is very clear. The helpers have a cute name and explanation, and generally the short docs are very concise and user-friendly. I can well imagine a lot of people getting into this. Keep it up!

[1] http://blog.getify.com/2010/02/grab-ui-by-the-handlebar
[2] https://github.com/getify/HandlebarJS

Reply

> **Anonymous on April 4, 2011 at 1:56 am said:**
> Thanks for the feedback, Barney! Unfortunately I don't have control of the name. Yehuda started the project and I've been simply helping out. From what I've seen, though, the name collision hasn't been too much of an issue thus far.

Reply

---

android gui on **March 29, 2011 at 7:53 pm** said:

nice article i'll definitely have to try it. Looking forwards to the see helpers

Reply

Anonymous on **March 28, 2011 at 4:58 pm** said:

seems great, i'll check it soon...

Reply

Mia on **March 25, 2011 at 9:55 pm** said:

loved reading your article , thank you for a thorough and elaborate explanation , it was quite insightful and as @Lemac mentioned ; I would like to read a follow up on this post and possibly see a live demo at some point in time ? ..... possibly ?..... maybe ? ^_^

Thank you for your effort 🙂

Reply

Sven Lito on **March 25, 2011 at 5:24 pm** said:

looking forward to the more advanced version of this blog post as the above is pretty much what everyone covers about a given templating library.

Reply

> **Anonymous** on **March 25, 2011 at 6:36 pm** said:
>
> Sorry – I couldn't figure out any way around teaching the basics first. Are you using any JavaScript templating?
>
> Reply
>
> > **Sven Lito** on **March 25, 2011 at 8:45 pm** said:
> >
> > No worries – Yes, I currently use Mustache and icanhazjs in various projects but handlebars certainly got me curious at it seems to be more flexible.
> >
> > Reply
> >
> > > **Anonymous** on **April 4, 2011 at 1:58 am** said:
> > >
> > > I wanted to let you know that I posted a part 2 to this original post, and it includes a ton more code and a link to a GitHub repo with example pages.

http://thinkvitamin.com/code/handlebars-js-part-2-partials-and-helpers/

Reply

Sven Lito on **March 25, 2011 at 5:24 pm** said:

looking forward to the more advanced version of this blog post as the above is pretty much what everyone covers about a given templating library.

Reply

Max Luzuriaga on **March 25, 2011 at 5:21 pm** said:

While I think this is a cool idea, I would be loath to use this kind of thing in an actual project. What happens if a user doesn't have Javascript? They don't get any content! And what of search engine crawlers? It's the same kind of problems that Less.js and others face.

On the other hand, I think a server-side implementation of this (in Ruby or Python) would be incredibly cool.

Reply

> **Anonymous** on **March 25, 2011 at 5:39 pm** said:
>
> This approach works best when creating a web app instead of

a document to be crawled. Picture an admin interface. As far as users without JavaScript, you need to know your audience.

Reply

Anonymous on March 25, 2011 at 6:35 pm said:

Great comment! You certainly need to use any javascript you're using strategically. I think audience is part of the equation. Sometimes you also just have to bite the bullet and write both a JavaScript and non-JavaScript version of the application. There are a lot of apps like GMail that have both javascript and non-javascript interfaces.

Reply

Florian Gilcher on March 25, 2011 at 8:54 pm said:

Not a problem: handlebars.js runs fine in therubyracer (a V8 binding for Ruby) (https://github.com/cowboyd/handlebars.rb). Otherwise, you can still use Mustache on the serverside and restricting yourself to using it in the frontend as well.

Reply

Barney Carroll on April 3, 2011 at 3:36 pm said:

You can run JS on the middle-end (back-end, but away from all the hardcore data storage and biz logic). Using this cleverly, you could write your templates once in this format, and have them run on the back-end or the front-end, without

writing separate manipulation code for in-browser changes and server-side permutations.

The distinct advantage in using front-end templates is that you can save untold amounts of back-and-forth with the server as far as repetitive markup is concerned: most of what servers send the user in traditional models is just tag soup whose basic structure the user already has. With a proper front-end templating framework, you can load all the HTML structures you'll ever need in one simple go, then only make further server requests for extra-lite JSON-formatted data, which is much much smaller in byte-size then fully parsed HTML+relevant content.

Producing the same content regardless of Javascript support with AJaX-reliant web apps is an increasingly pertinent problem that ultimately depends on the back-end being able to do the same things as the scripted front-end. What better way of doing that then writing once in Javascript and running anywhere depending on the client's JS support?

Reply

Lemac on March 25, 2011 at 1:39 pm said:

Could you link to a demo of the above code? I don't see it anywhere.

Reply

> **Anonymous on March 25, 2011 at 4:10 pm said:**
> Great question – unfortunately I don't have a demo of it up at this point. Let me know if you have any questions about handlebars, though – I'd love to help!
>
> Reply
>
> > **DominixZ on March 26, 2011 at 3:25 pm said:**
> > Is it possible to get template code to external javascript ?
> >
> > Reply
> >
> > > **Anonymous on April 4, 2011 at 1:59 am said:**
> > > It's something we're working on. I'm hoping to have it finished up and be able to post some examples of compilation to external javascript files in the next couple of weeks.
> > >
> > > Reply

Leave a Reply

Name *

Email Address (will not be published) *

Website

Submit Comment

Want to learn more about Javascript?

Learn how to use JavaScript to add interactivity to websites.

Learn more

Stay current

Sign up for our newsletter, and we'll send you news and tutorials on web design, coding, business, and more!

Email Address

Subscribe