JavaScript.isSexy

Handlebars.js Tutorial: Learn Everything About Handlebars.js JavaScript Templating

FEB. 18 2013 123

menu

(A Comprehensive Handlebars.js Tutorial)

This is a complete tutorial, and indeed a reference, on Handlebars.js templating and, principally, JavaScript templating. Handlebars.js is a client-side (though it can be used on the server, too) templating engine for JavaScript. It is a JavaScript library that you include in your page just as you include any other JavaScript file. And with it, you can add templates to your HTML page that will be parsed and interpolated (values of properties inserted in place) with the values from the data you passed to the Handlebars.js function.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Written in JavaScript, Handlebars.js is a compiler that takes any HTML and Handlebars expression and compiles them to a JavaScript function. This derived JavaScript function then takes one parameter, an object—your data—and it returns a string with the HTML and the object properties' values inserted into the HTML. So, you end up with a string that has the values from the object properties inserted in the relevant places, and you insert that string as HTML on the page.

Do I Have To Use a JavaScript Templating Engine? If so, Why?

Yes. If you develop or plan to develop JavaScript applications, you should use a JavaScript client-side templating engine to

Receive Updates

Your Email:

Go

Table of Contents

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

keep your JavaScript and HTML sufficiently decoupled; that is, keep your HTML separate from your JavaScript, which allows you to manage your HTML and JS files reliably and easily.

Sure, you can use JSDom, or you can fire up server-side templating and send your HTML files via HTTP. But I recommend client-side templating because it typically executes faster than server-side templating and it provides the easiest way to create and maintain your templates.

In addition, just about all the JavaScript front-end frameworks use a JavaScript templating engine, so you will eventually use JavaScript templating because you will likely use a JavaScript front-end or backend framework.

When to Use a JavaScript Templating Engine and Why to Use Handlebars.js?

The answers to both of these questions follow. Indeed, there exist some specific use cases for JavaScript templating engines.

When To Use a JavaScript Templating Engine?

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

You should use a JavaScript templating engine like Handlebars.js when:

- You use a JavaScript front-end framework like Backone.js, Ember.js, and the like; most front-end JavaScript frameworks rely on templating engines

- The application's view (the HTML page or portions of the page) will be updated frequently, especially as a result of changes to the data either from the server via a REST API or from data on the client

- You have multiple tech stacks that depend on your data from the server and you want all the tech stacks to process the same data

- Your application has much interactivity and it is very responsive

- You are developing a single-page web application with multiple views

- You want to easily manage your HTML content; you don't want your JavaScript code to contain important HTML markup. Here is an example of JS code with HTML markup (it makes it difficult to manage your HTML markup):

```
shoesData.forEach (function (eachShoe)  {
//Note the intermingling of HTML and JavaScript; it is tedious to follow:
theHTMLListOfShoes += '<li class="shoes">' + '<a href="/' + eachShoe.name.toLowerCase() + '">
' + eachShoe.name + ' -- Price: ' + eachShoe.price + '</a></li>';
```

```
  });
 return theHTMLListOfShoes;
}
```

Why Use Handlebars.js (out of the eight or more templating engines)?

Unsurprisingly, there are many JavaScript client-side templating engines, but we will focus on only Handlebars.js in this tutorial, since it is the best of the lot. Some of the other worthy templating engines are Underscore.js' Template, Mustache.js, EJS, and Dust.js.

Handlebars.js is an extension of the Mustache JavaScript templating language; it supersedes Mustache.js.

The reasons you should use Handlerbars.js follow:

- Handlebars is one of the most advanced (pre-compiling and the like), feature-rich, and popular of all the JavaScript templating engines, and it has the most active community.

- Handlebars is a logic-less templating engine, which means there is little to no logic in your templates that are on the HTML page. The most important use of Handlebars, and any templating engine, is to keep your HTML pages simple and clean and decoupled from the logic-based JavaScript files, and Handlebars serves this purpose well. Indeed, Dust.js is

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

also a logic-less templating engine and a worthy alternative to Handlebars.js.

- Moreover, the cutting-edge JavaScript frameworks **Meteor.js** and **Derby.js** , which we will cover in upcoming posts, are expected to become mainstream in the coming months, and both use Handlebars.js. To be clear: Meteor.js uses Handlebars.js and Derby.js "template syntax is largely based on Handlebars" template syntax. And **Ember.js uses Handlebars** , too.

While Backbone.js is packaged with Underscore.js templating engine, it is super easy to use Handlebars.js with Backbone.js.

Therefore, the experience and knowledge you will have gained from learning Handlebars.js now will be well worth it, if you use, or plan to use, any of the noted JS frameworks.

In short, learning Handlebars.js now is an investment and a wise choice: you will program more effectively now and you will adapt easily to the JS frameworks tomorrow and in the coming weeks and months.

Handlebars.js Overview

Now that we have seen how to use Handlebars in a simple application, let's study Handlebars in detail.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

How Handlebars.js Works?

As noted in the introduction: Handlebars.js is a compiler built with JavaScript that takes any HTML and Handlebars expression and compiles them to a JavaScript function. This derived JavaScript function then takes one parameter, an object—your data—and it returns an HTML string with the object properties' values inserted (interpolated) into the HTML. So, you end up with a string (HTML) that has the values from the object properties inserted in the relevant places, and you insert the string on a page.

This sounds way more complex that it is, so let's take a closer look.

The 3 Main Parts of Handlebars Templating

To use Handlebars, first you link to the Handlebars.js file in the head block of your HTML page, just like you do for jQuery or any .js files.. Then there are 3 main pieces of code you use for Handlebars templating:

1. **Handlebars.js Expressions**
   A simple Handlebars expression is written like this (where "content" can be a variable or a helper function with—or without—parameters:

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
{{ content }}
```

Or like this, in the case of Handlebars block expressions (which we will discuss in detail later):

```
{{#each}}
HTML content and other Handlebars expresion go here.
{{/each}}
```

Below is a Handlebars expression with HTML. The customerName variable is the property that will be interpolated (its values will be inserted in place) by the Handlebars.compile function:

```
<div> Name: {{ customerName }} </div>
```

The output will be the following (if the customerName variable has the value “Richard”):

Richard

Since you have to pass the Handlebars expression (with any containing HTML) to the Handlebars.compile function, a script tag is used to enclose each Handlebars template when they are on the HTML page. Indeed, the script tag is not necessary when a template is in its own HTML file, but it is necessary when the Handlebars template is on the page

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

along with other Handlebars template and other HTML content.

— Script Tag
Handlebars templates are embedded in script tags (where the script tag's type property is set to "text/x-handlebars-template"). The script tag is similar to the script tag you normally use to embed JavaScript in the HTML page, except the type attribute is different.   You retrieve the HTML content from the script tag and pass it to the Handlebars compiler.

Here is an example of the Handlebars script tag:

```
<script id="header" type="text/x-handlebars-template">
 <div> Name: {{ headerTitle }} </div>
</script>
```

2. **Data (or Context)**
The second piece of code in Handlebars templating is the data you want to display on the page. You pass your data as an object (a regular JavaScript object) to the Handlebars function. The data object is called the context. And this object can be comprised of arrays, strings, numbers, other objects, or a combination of all of these.

If the data object has an array of objects, you can use Handlebars each helper (more on helpers later) function to iterate the array, and the current context is set to each item in

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

the array.

Here are examples of setting up the data object and how to iterate it in a Handlebars template.

— Data object with array of objects

```
//The customers object has an array of objects that we will pass to Handlebars:
var theData = {customers:[{firstName:"Michael", lastName:"Alexander", age:20}, {firstName:"John",
 lastName:"Allen", age:29}]};
```

You can use the each helper to iterate the customer's object like this:

```
 <script id="header" type="text/x-handlebars-template">
 {{#each customers}} // Note the reference to the customers object
 <li> {{ firstName }} {{ lastName }} </li>
   {{/each}}
</script>
```

Or, since we are passing the customer's object as an array of objects, we can use a block helper (more on block helpers later) statement like this and reference the customers directly:

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
<script id="header" type="text/x-handlebars-template">
 {{#customers}} // In this example, because the customer's
  <li> {{ firstName }} {{ lastName }} </li>
    {{/customers}}
</script>
```

— Data object with Strings

```
// In this example, the data object contains properties with strings:
 var theData = {headerTitle:"Shop Page", weekDay:"Wednesday"};

<script id="header" type="text/x-handlebars-template">
  <div> {{ headerTitle }} </div>
Today is {{weekDay}}
</script>
```

3. **The Handlebars Compile Function**

   The last piece of code we need for Handlebars templating is actually a two-step execution:
   1. Compile the template with the Handlebars.compile function.
   2. Then use that compiled function to invoke the data object passed to it (it takes a data object as its sole parameter). And this will return an HTML string with the interpolated

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

object values inserted into the HTML.

In short:
The Handlebars.compile function takes the template as a parameter and it returns a JavaScript function. We then use this compiled function to execute the data object and return a string with HTML and the interpolated object values. Then we can insert the string into the HTML page.

Here are the 3 pieces together:
1. On the HTML age: Setup the templates by using Handlebars expressions, and add the templates to a script tag (if using script tags: templates in individual HTML files don't need script tags):

```
 <script id="header" type="text/x-handlebars-template">
 <div> {{ headerTitle }} </div>
Today is {{weekDay}}
</script>
```

2. In the JavaScript file: Initialize the data object

```
var theData = {headerTitle:"Shop Page", weekDay:"Wednesday"};

```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
// Retrieve the HTML from the script tag we setup in step 1
 // We use the id (header) of the script tag to target it on the page
 var theTemplateScript = $("#header").html();
```

3. Also in the JavaScript file: Then we use the Handlebars compile function to compile the templates.
Compile the template retrieved from the script tag:

```
// The Handlebars.compile function returns a function to theTemplate variable
 var theTemplate = Handlebars.compile (theTemplateScript);
```

Use the theTemplate () function returned by the compile function to generate the final string with interpolated object values. We pass the object data as a parameter. Then attach the resulting string with HTML to the page:

```
 $(document.body).append (theTemplate (theData));
```

This will return our HTML with the values from the object inserted in place, and the result will look like this:

Shop Page

Today is Wednesday

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Compare a Non-Handlebars Project With a Handlebars.js Project

To get a high-level overview of what development with Handlebars entails when compared with non-JavaScript templating, let's build a quick, tiny JavaScript project.

Before we use Handlebars, let's make use of jQuery and JavaScript without Handlebars, to get a sense of what we are building and how Handlebars will make a difference.
Without Handlebars, this is what a typical JavaScript/jQuery project will look like when you have content to add to the page with some data. We are simply displaying a list of shoes and prices .

A Little Non-Handlebars Project

1. Download Handlebars.js and jQuery:
   Download the latest version of Handlebars from Github (we are not using it yet, though, but still include it on the page). Get the full version, not the "runtime only" version (more on the runtime version later):

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

https://github.com/wycats/handlebars.js/downloads

Also, download the latest version of jQuery here (we will use it throughout this tutorial): http://code.jquery.com/jquery-1.9.1.min.js

2. Create a new directory on your computer named "Handlebars_tuts" and place the jQuery and Handlebars.js files in it.
   Or, you can open Terminal (on MAC) and change directory to the "Handlebars_tuts" directory. Then type the following commands to download both JS files directly to the directory with the curl command:

```
curl http://code.jquery.com/jquery-1.9.1.min.js > jquery-1.9.1.min.js
curl https://github.com/downloads/wycats/handlebars.js/handlebars-1.0.rc.1.min.js > Handlebars.js
```

3. Make an index.html file and add the following

```
<html>
 <head>
 <script type="text/javascript" src="jquery-1.9.1.min.js"></script>
 </head>
   <body>
    The List of Shoes:
```

```
    <ul class="shoesNav"></ul>
   </body>
</html>
```

4. Create a main.js file and add the following:

   Note this JS file has both HTML and JavaScript mixed together in an unhealthy soup

```
$(function () {

var shoesData = [{name:"Nike", price:199.00 }, {name:"Loafers", price:59.00 }, {name:"Wing Tip", price:259.00 }];

function updateAllShoes(shoes) {
var theHTMLListOfShoes = "";

shoesData.forEach (function (eachShoe) {
// Note the coupling and mixing of HTML and JavaScript; it is tedious to follow
 theHTMLListOfShoes += '<li class="shoes">' + '<a href="/' + eachShoe.name.toLowerCase() + '">' + eachShoe.name + ' -- Price: ' + eachShoe.price + '</a></li>';
   });
   return theHTMLListOfShoes;
}
$(".shoesNav").append (updateAllShoes(shoesData));
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```

});
```

If you open the index.html file in your browser, you should see a simple list with 3 items. This is how we normally develop on the front end without a JavaScript template engine.

A Little Handlebars Project

Now, lets refactor the code above and use Handlebars.js templating instead of relying on HTML and JavaScript coupled together “unhealthily.”

1. Changes to index.html:
   Add this code just below the closing ul tag

```
<script id="shoe-template" type="x-handlebars-template">
  {{#each this}}
    <li class="shoes"><a href="/{{name}}">{{name}} -- Price: {{price}} </a></li>
   {{/each}}
</script>
```

2. Changes to main.js:

And here is the refactored JS code that makes use of Handlebars.

Remove all the JS code and replace with the code below

— Note that we have gotten rid of the entire updateAllShoes functions

— And also note there is no HTML in the JS file anymore, all the HTML now live in the HTML file.

```
$(function  () {
  var shoesData = [{name:"Nike", price:199.00 }, {name:"Loafers", price:59.00 }, {name:"Wing Tip", price:259.00 }];
   //Get the HTML from the template   in the script tag
    var theTemplateScript = $("#shoe-template").html();

   //Compile the template
    var theTemplate = Handlebars.compile (theTemplateScript);
    $(".shoesNav").append (theTemplate(shoesData));

//We pass the shoesData object to the compiled handleBars function
// The function will insert all the values from the objects in their respective places in the HTML and returned HTML as a string. Then we use jQuery to append the resulting HTML string into the page
});
```

When you refresh the index.html page in your browser should see the same output we

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

had in the previous non-Handlebars example above.

The preceding illustrate the very basic use of Handlebars.js. As you have seen, using Handlebars allowed us to separate the HTML from the JavaScript. This is even more important as our application gets more complex; the easier it will be to develop separate template files and manage them effectively. Whereas, the non-Handlebars example would be a mess to manage as our application gets larger.

That in essence is how using Handlebars.js allows us to keep our HTML files decoupled from our JavaScript files.

The Main Difference Between the Two Projects

This is the main difference between the non-Handlebars.js project and the Handlebars.js project: The non-Handlebars.js project has important HTML markup inside the JavaScript code, which makes it difficult to manage (create and update) the HTML:

```
// You can see the HTML and JS intermingled

function updateAllShoes(shoes)  {
var theHTMLListOfShoes = "";
shoesData.forEach (function (eachShoe)  {
```

```
theHTMLListOfShoes += '<li class="shoes">' + '<a href="/' + eachShoe.name.toLowerCase() + '
">' + eachShoe.name + ' -- Price: ' + eachShoe.price + '</a></li>';
  });
  return theHTMLListOfShoes;
}
$(".shoesNav").append (updateAllShoes(shoesData));
```

While the Handlebars.js project’s JavaScript code does not contain the HTML markup (the HTML markup is on the HTML page; only JS is in the JS code):

```
var theTemplateScript = $("#shoe-template").html();
var theTemplate = Handlebars.compile (theTemplateScript);
$(".shoesNav").append (theTemplate(shoesData));
```

Learn Handlebars.js Syntax

- **Handlebars.js Expressions**

  We saw the Handlebars Expressions above. Handlebars expressions are written like this (a double stash before, followed but the content to be evaluated, followed a double closing double stash):

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
<div>{{ content goes here }}
```

The customerName variable is the property (the expression to be evaluated by the Handlebars compiler) that will be interpolated (its values will be inserted in place) by the Handlebars compiled function, when it executes:

```
<div> Name: {{ customerName }} </div>
```

- **Comments**

This is how you add comments in a Handlebars template:

```
{{! Whatever is inside this comment expression will not be outputted  }}
```

And you can also use regular HTML comments, but they will be outputted on the HTML page source, as HTML comments of course:

```
<!-- Regular HTML comments will be in the output -->
```

- **Blocks**

Blocks in Handlebars are expression that has a block, an opening {{# }} followed by a closing {{/ }}.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

We cover blocks in detail later; this is the syntax for a block:

```
{{#each}} Content goes here. {{/each}}
```

Here is an if block

```
{{#if someValueIsTrue}} Content goes here {{/if}}
```

The words block and helper are sometimes used interchangeably because most built-in helpers are blocks, although there are function helpers and block helpers.

Paths (with dot notation)

A path in Handlebars is a property lookup. If we have a name property that contains an object, such as:

```
var objData = {name: {firstName: "Michael", lastName:"Jackson"}}
```

We can use nested paths (dot notation) to lookup the property you want, like this:

```
{{name.firstName}}
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Parent Path ../

Handlebars also has a parent path ../ to lookup properties on parents of the current context. Thus:

With a data object such as this:

```
var shoesData = {groupName:"Celebrities", users:[{name:{firstName:"Mike", lastName:"Alexander" }}, {name:{firstName:"John", lastName:"Waters" }} ]};

We can use the parent path ../ to get the groupName property:
<script id="shoe-template" type="x-handlebars-template">
  {{#users}}
    <li>{{name.firstName}} {{name.lastName}} is in the {{../groupName}} group.</li>
   {{/users}}
</script>
```

The rendered HTML will be:

Mike Alexander is in the Celebrities group.

John Waters is in the Celebrities group.

Context

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Handlebars refers to the object you passed to its function as the context. Throughout this article, we use "object data" and sometimes "data" or "object" to refer to the context object. All these words are used interchangeably from time to time, but you will no doubt understand that we are referring to the object being passed into the Handlebars function.

- ## Triple Stash {{{ }}} For Non-escape HTML

Ordinarily, you use Handlebars's double stash {{ }} for expressions, and by default, Handlebars content in this standard double stash is escaped to "protect you against accidental XSS problems caused by malicious data passed from the server as JSON." [1]This ensures that nefarious code in the HTML cannot be injected into the page.   But sometimes you want the raw (un-escaped) HTML instead. For this, you can use Handlebars's triple stash instead {{{ }}}. The triple stash tag signifies to handlebars to do not escape the HTML content contained in the triple stash.

- ## Partials (sub-templates)

Sometimes you have to render a section of a template within a larger template. You use Partials to do this in Handlebars, and this is the partials expression:

```
{{> partialName}}
```

Let's add a partial template to the basic Handlebars project we built earlier. We will add

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

color and size under each shoe.

Changes to main.js

1. Replace the existing data object with the following (we are adding properties for color and size):

```
var shoesData = {allShoes:[{name:"Nike", price:199.00, color:"black", size:10 }, {name:"Loafers", price:59.00, color:"blue", size:9 }, {name:"Wing Tip", price:259.00, color:"brown", size:11 }]};

```

2. Add the following just before the line with the jQuery append method:

```
Handlebars.registerPartial("description", $("#shoe-description").html());
```

Changes to index.html:

1. Replace the shoe-template template with this:

```
<script id="shoe-template" type="x-handlebars-template">
  {{#each allShoes}}
   <li class="shoes">
   <span class="shoe-name"> {{name}} - </span> price: {{price}}
```

```
        {{> description}}
    </li>
    {{/each}}
</script>
```

2. Add this new template for the description; add it just below the shoe-template:

```
<script id="shoe-description" type="x-handlebars-template">
<ul>
        <li>{{color}}</li>
        <li>{{size}}</li>
</ul>
</script>
```

The Final HTML should show color and size under each shoe list item.

That is how you add a partial- (or sub-template) to the main template.

Handlebars.js Built-in Helpers (Conditionals and Loops)

Expand Content

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Handlebars.js Custom Helpers

(very important and versatile)
In addition to the built-in helpers we just discussed, Handlebars allows us to add our own custom helpers, and this is even more important than the built-in helpers, since we can add any kind of complex logic to the custom helpers. We can even re-create the built-in helpers with our own custom helpers, though this would be a waste of time.

With custom helpers, we can add any kind or JavaScript logic. We register the custom helper before all the rest of the Handlebars JS code. Custom helpers are created in the JavaScript code, not inside the Handlebars template.

There are two types of custom helpers you can create: a custom **function helper** is simply a helper that does not use a block, and a custom **block helper** is a helper with a block.

Custom Function Helpers

Let's make a simply custom function helper that executes some conditional logic, since we could not use conditional logic with the built-in helpers.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

The data object :

```
var contextObj = {score:85, userName:"Mike"};
```

First, we have to register the custom helper with Handlebars.registerHelper method. This method takes a string (the name of the helper) as the first parameter and a function with any number of parameters as the second parameter.

```
Handlebars.registerHelper ("theNameOfTheHelper", function (theScore) {
   console.log("Grade: " + theScore );
   var userGrade = "C";

  if (theScore >= 90) {
      return "A" ;
  }
  else if (theScore >= 80 && theScore < 90) {
      return "B" ;
  }
  else if (theScore >= 70 && theScore < 80) {
      return "C" ;
  }
  else {
      return "D" ;
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
   }

});
```

Here is the Handlebars template that uses the custom function helper we just created:

```
 <script id="shoe-template" type="x-handlebars-template">
 {{theNameOfTheHelper score}}
</script>
```

The output on the HTML page is:
 B

Custom Block Helpers

In addition to custom function helpers, we can also add custom block helpers. When we register a custom block helper, Handlebars automatically adds an options object as the last parameter in the callback function. And the options object has an **fn** method, a **hash** object, and an **inverse** method.

The **options.fn** method:
The fn method takes an object (your data) as a parameter that it uses as the context inside the custom helper block template. You can pass any data object, or if you want to use the

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

same data context referenced in the template, you can use this.

An example will better illustrate. This is the data object we are using (we will sum all the scores in the array and replace the score array with the total of the scores:

```
 var contextObj = [{firstName: "Kapil", lastName:"Manish", score:[22, 34, 45, 67]}, {firstName: "Bruce", l
astName:"Kasparov", score:[10, 34, 67, 90]}];
```

Here we setup the template with the userScore block helper, which we define below:

```
 <script id="shoe-template" type="x-handlebars-template">
 {{#userScore this}}
<div>{{firstName}} {{lastName}}, Your Total Score is {{score}} </div>
 {{/userScore}}
</script>
```

We register the userScore block helper with Handlebars.registerHelper.
Note the last item in the parameter is the options object, which Handlebars inserts automatically, and we use it here:

```
 Handlebars.registerHelper ("userScore", function (dataObject, options) {
var templateWithInterpolatedData = "";
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
for (var i = dataObject.length - 1; i >= 0; i--) {
    //Sum user scores from the score array and replace the array with the total
    dataObject[i].score = dataObject[i].score.reduce(function (prev, cur, index, array) {
                return prev + cur;
            });

// Each object in the data object array is interpolated with the options.fn method, which processes all
the HTML from the template and insert the values from the object in their respective positions.

// Just so you understand the purpose of the options.fn method: it does exactly what the regular handl
ebars template function does when we pass the data object to the function, to  retrieve the values fro
m the object and insert them into the HTML from the template.

// Without the options.fn object in this example, the raw objects (instead of the interpolated values) w
ould have been returned

templateWithInterpolatedData += options.fn (dataObject[i]);

}

// We return the full string of HTML with all the values from the data object inserted into place.

return templateWithInterpolatedData;
});
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

And this is the HTML output:
Bruce Kasparov, Your Total Score is **201**
Kapil Manish, Your Total Score is **168**

— It is also important to know that a custom block helper can be inserted anywhere in the template, and we can pass any number of parameters in the template for the custom block helper.

The **options.inverse** method:
The inverse method is used as the else section of any block statement. So, you would use options.fn to return when the expression in the callback evaluates to a truthy value. But you would use options.inverse when the expression evaluates to falsy (to render content in the else part of the block).

The **options.hash** object:
Handlebars expressions take not only strings and variables as arguments, but you can pass key-value pairs separated by spaces as well.
For example:
(Note there is no comma separating the key-value pairs arguments.)

```
 {{#myNewHelper score=30 firstName="Jhonny" lastName="Marco"}}
Show your HTML content here.
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
{{/myNewHelper}}
```

And that invocation of the Handlebars expression with the key-value pairs as parameters will be added automatically onto the hash object in the helper callback function. Thus:

```
 Handlebars.registerHelper ("myNewHelper", function (dataObject, options) {
//JSON.stringify used to serialize the object (to a string)
console.log(JSON.stringify (options.hash));
//The output is: {score:30, firstName:"Jhonny", lastName:"Marco"}
});
```

Four Ways to Load/Add Templates

There are four main ways you can add your Handlebars.js templates to your pages:

1. **Use Script Tags**
   The first and simplest, indeed, the least desirable, way to add the Handlebars template to the page is by adding the script tag, as we have done throughout this tutorial so far. You add the entire script tag with the template, like the example below, on the HTML page:

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
<script id="shoe-template" type="x-handlebars-template">
{{#userScore this}}
<div>{{firstName}} {{lastName}}, Your Total Score is {{score}} </div>
{{/userScore}}
</script>
```

Pros

- Quick to setup and use, if you have a very simple application or page
- Little to no ramp-up time (time to learn and implement is negligible).

Cons

- All the templates are on the page in script tags, which becomes a nightmare to maintain if you have many templates.
- It can be problematic to manage memory effectively in medium to large applications, since all the templates are always on the page and you cannot easily add and remove them, for example in a single-page web application.
- It is difficult to create highly-responsive interfaces with these static templates that have a full life cycle in the application.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

- The templates cannot be precompiled. Precompiled templates are executed considerably faster than templates that have to be compiled in the browser before being rendered. Compiling templates is the most expensive operation for JavaScript templating engines. So the inability to precompile the templates would result in avoidable latency in your application.

2. **Use Custom Function**
   You can place all of your templates in HTML files (without the script tag) and load them and compile in one go. Then you can add the compiled templates to the page whenever they are needed.

   A StackOverflow user, koorchik, has created a custom function to handle the loading and the compiling of templates in a concise manner. It is below.

   Pros

   - All the templates can be kept in separate HTML files, which allows for easy maintenance.

   - No ramp-up time to learn and implement the function: after studying the function for a bit, you will understand it and will be ready to use it in a few minutes.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

- The function is very lightweight.
- It is versatile, because it facilitates the use of both precompiled templates and templates that haven't been compiled; and you can even pass preloaded templates to the function.

Con

- None as yet :), but feel free to suggest any in the comments below.

The Custom Function

```
// And this is the definition of the custom function
function render(tmpl_name, tmpl_data) {
    if ( !render.tmpl_cache ) {
        render.tmpl_cache = {};
    }

    if ( ! render.tmpl_cache[tmpl_name] ) {
        var tmpl_dir = '/static/templates';
        var tmpl_url = tmpl_dir + '/' + tmpl_name + '.html';
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
        var tmpl_string;
        $.ajax({
            url: tmpl_url,
            method: 'GET',
            async: false,
            success: function(data) {
                tmpl_string = data;
            }
        });

        render.tmpl_cache[tmpl_name] = _.template(tmpl_string);
    }

    return render.tmpl_cache[tmpl_name](tmpl_data);
}
```

This is how you would use the custom function to load a mytemplate.html file; pass a data object as second parameter

```
var rendered_html = render('mytemplate', {});
```

3. **Use AMD and Require.js**
   AMD is a specification that outlines a mechanism for loading modules and their dependencies asynchronously. We will leave the particulars of the AMD specification

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

for another article.

For the purpose of this Handlebars templating, the bottom line is that we use a define () function to register our modules and dependencies (including templates) and Require.js (a file and module loader) will load our templates so we can assign them to a variable and use them in our project. More on AMD and Requires.js in the Backbone.js application later.

Pros

- With the AMD module mechanism, your code will be well organized.
- You can easily and dynamically load templates in a structured manner.
- You can keep all the templates in separate HTML files for easy maintenance.
- Different developers can edit or create different HTML template files.
- Require.js can concantenate all JS files into one file JS file, thus reducing the number of HTTP requests and download time for the application.

Con

- Steep Ramp-up time: AMD and Require.js have a moderate to steep learning

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

curve, and you might likely have to learn Backbone.js or another JS framework that utilize these technologies in an abstracted manner.

Handlebars template with AMD and Require.js Example

The templates are in separate HTML files, and they don’t need the script tag anymore, so the user_account.html file looks like this:

```
 // The define function is part of the AMD mechanism for loading
define([
    'jquery',
    'underscore',
    'handlebars',
// Require.js text plugin loads the HTML template pages
    'text!templates/user_account.html',
    'text!templates/user_profile.html'],
function ($, _, Backbone, HandleBars,
UserAccount_Template, UserProfile_Template) {

// These are the two objects we will be using
userAccountDataObject: {accountNumber:85444, balance: $120.00},
userProfileDataObject: {firstName:”Michael, lastName:”Harrington”},

// Compile the Handlebars template that we loaded in (the user_account.html and user_profile.ht
ml) and assign them to the following two variables.
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
userAccount_compiledTemplate:
HandleBars.compile(UserAccount_Template),
userProfile_compiledTemplate:
HandleBars.compile(UserProfile_Template),

// This function will add the Handlebars compiled templates on the page
render: function () {
// Use the compiled function (userAccount_compiledTemplate) and pass to it the context (the da
ta object). The result will be the HTML from the template page with the interpolated values from
the data object.

this.$(".user-account-div").html
(this.userAccount_compiledTemplate(userAccountDataObject);

// Ditto for this template function
this.$(".user-profile-div").html
(this.userProfile_compiledTemplate(userProfileDataObject);
});
```

4. **Precompile The Templates**
   With the script tag and the AMD/Requires.js methods of adding the templates to the page, Handlebars has to compile the templates on the client side, and this is the most expensive operation for JavaScript Template engines.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

To reduce latency and speed up page execution, Handlebars has a node.js module that precompiles your templates. And because the templates are precompiled, the execution time is considerably faster and you only include a runtime Handlebars.js file on the page.

I do not include an example code to show how to precompile Handlebars template, because the code will be best understood within the context of a complete application, and we will build a complete application with Backbone.js and precompile the Handlebars template in part 2 of this post.

Pros

- Because the precompiled template files are in JavaScript, they can be minified and combined into one JavaScript file for the entire application, including all the libraries and other JS files. This will significantly reduce the number of HTTP requests to the server on startup, and thus reduce the download time for the application.

- The application will execute considerably faster, because the expensive compile operation has been completed.

- All the advantages with the AMD/Require.js option, if used with AMD and

Require.js.

Cons

- The Handlebars precompile script is a Node.js module, which means you need to have Node.js installed on your development machine to precompile the templates.
- Since all the templates files are now JavaScript files, the templates cannot easily be edited on the server, to make quick changes.
- Making changes to files is a two-step process: You change the HTML files, then you have to run the precompile step to compile the templates to JavaScript. However, this last step can be enhanced with watch folders that automatically notices changes to the templates files in the template folder, and precompile the templates immediately.

Handlebars.js with Backbone.js, jQuery, Ember.js, and Meteor.js

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Part 2 (coming soon)
In a follow-up post, we get to use Handlebars.js with some of the most popular JavaScript frameworks. We will build 4 quiz applications with 4 different frameworks (jQuery, Ember.js, Backbone.js, and Meteor.js), and we will use Handlebars in all the applications. Furthermore, we will load the templates in each of the 4 different ways described above.

Try Handlebars Online in your Browser

The website at the link below has a useful utility that you can use to try Handlebars online in your browser.
Here is the link:
http://tryhandlebarsjs.com/

Notes

1. http://handlebarsjs.com/

Our Career Paths and Courses Website Is Now Live

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Learn.Modern Developer Launched

Our first cohort is in session: 97% of our first cohort on target to graduate. Enroll in the second cohort. Career Path 1: JavaScript Developer and Career Path 3: Modern Frontend Developer **usually fill up quickly**.

https://learn.moderndeveloper.com

Posted in: JavaScript Libraries / Tagged: HandleBars.js

Richard

Thanks for your time; please come back soon. Email me here: javascriptissexy at gmail email, or use the contact form.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

123 Comments

Rameş Aliyev

February 19, 2013 at 11:50 am / Reply

Thanks for a lovely article! That four ways to add / load template part is so creative i think, these ways broadened my mind!

Richard Bovell (Author)

February 19, 2013 at 11:57 am / Reply

You are welcome, Rameş, and thank you 🙂

I just finished cleaning up the post and making some changes about 20 minutes ago, so if you read it more than an hour ago, you might want to take another peep. Although, I did not make much changes to the section you referenced.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Rameş Aliyev

February 19, 2013 at 2:51 pm / Reply

Thanks for reply and inform, i've checked quickly and read the new ones 🙂 This was lovely, but now become that woman which i fall in love with 😃 Thanks again, Cya!

Tlg B

February 22, 2013 at 4:24 pm / Reply

Great post dude! That's one of the things that i wanna learn but couldn't find a proper tutorial since i found that one. thanks again

Sergi

March 1, 2013 at 12:14 pm / Reply

Very concise, Rames, and the flow is wonderful too. One

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

note is that you've made an error in the "partials" explanation. The object you ask us to cut/paste doesn't include the object allShoes, so we get an error when we try to load it into the block.

Sergi

March 5, 2013 at 10:21 am / Reply

Oops! I meant to say "Richard", not Rames. So, great tutorial Richard!

Richard Bovell (Author)

March 7, 2013 at 3:26 pm / Reply

Thanks for pointing that out, Sergi. I added the correct data for partials. Here it is:
var shoesData = {allShoes:[{name:"Nike", price:199.00, color:"black", size:10 }, {name:"Loafers", price:59.00, color:"blue", size:9 }, {name:"Wing Tip", price:259.00,

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

color:"brown", size:11 }]};

Kranky

March 7, 2013 at 6:52 am / Reply

Nice article Richard...!!! Loved the Four ways to add/load Templates.. which i was looking at...

You have referred about custom function posted on stack overflow. does it work with handlebars.js.? does it require any other file/library to be included.?

I am struck with loading the templates part.?

Cheers

Richard Bovell (Author)

March 7, 2013 at 12:45 pm / Reply

You have referred about custom function posted on

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

> stack overflow. does it work with handlebars.js.?
> does it require any other file/library to be included.?

Yes, the custom function work with Handlebars.js.

> I am struck with loading the templates part.?

What specifically you are having trouble with?

Keith Moore

May 2, 2013 at 10:00 am / Reply

Great article! Thanks for posting. I just wanted to point out that the "custom function" from the stackoverflow user does not use handlebars.js. It uses underscore templates. If you want handlebars.js/mustache.js like variables, you can add the following to the custom function:

```
_.templateSettings = {
interpolate : /\{\{(.+?)\}\}/g
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

};

This allows you to do something like this:
var rendered_html = render('mytemplate', { name: "Keith"});

Where name would be defined in the template as {{name}}

Richard Bovell (Author)

May 2, 2013 at 11:10 am / Reply

Keith,
Thanks very much for your comment. I will update the post accordingly, which I plan to do soon when I add part 2. Thanks for taking the time to point out the extra, missing detail.

Fabio

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

May 3, 2014 at 10:39 pm /

to use it in Handlebarsjs i've edited the line that use underscore into this:

render.tmpl_cache[tmpl_name] = Handlebars.compile(tmpl_string);

vps

March 14, 2013 at 11:09 am / Reply

This blog is amazing. I realy like it!

Richard Bovell (Author)

March 15, 2013 at 10:34 pm / Reply

Thank you, vps,

Encouraging comments like yours keep me motivated 🙂

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Ashwini Mohan

April 9, 2013 at 2:31 am / Reply

Awesome tutorial thank uuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuu so much.... good work 🙂

Richard Bovell (Author)

July 31, 2013 at 1:39 am / Reply

Thanks very much, Ashwini.

Ashwini Mohan

April 9, 2013 at 2:31 am / Reply

good tutorial

Coleman Foley

April 28, 2013 at 8:23 pm / Reply

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Excellent tutorial. I was having a hard time figuring Handlebars out just by looking at the example on the official page, and this is just the kind of step-by-step tutorial I was looking for.

Also, I think you forgot to put a line including handlebars in the html in step 3 of "A Little Non-Handlebars Project". You said to include it before that, but it's easy to forget that once you actually write the HTML.

Richard Bovell (Author)

April 29, 2013 at 6:47 am / Reply

Coleman,
I meant to include the Handlebars.js file in the folder. The reason it is not included in the HTML at that point is because that specific example is a **non-Handlebars** example.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Jerimiah

April 9, 2014 at 4:01 pm / Reply

The line that is missing from the non-handlebars index.html example is:

For me, main.js does not run automagically.

Jerimiah

April 9, 2014 at 4:02 pm / Reply

Hmm, code got eaten. In plain english, duplicate the line for the embedded jquery, and change the filename in that line to ‘main.js’.

Salamat Ali

May 5, 2013 at 9:11 am / Reply

This tutorial along with everything else on your site is excellent.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

I've been learning javascript for just over a month(following your learn JS completely) and have been dipping in and out of tutorials such as this.

Your style of explaining is so clear and precise that someone like myself who has never seen a line of code let alone write any is already building small web apps after just over a month.

I look forward to part 2 with great enthusiasm.

Keep up the great work!

Richard Bovell (Author)

May 5, 2013 at 4:37 pm / Reply

Thank you very much, Salamat.

aditya menon

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

May 5, 2013 at 11:44 am / Reply

For some reason, $(document).append did not work for me, I needed to use $('#id_of_target').append;

Richard Bovell (Author)

May 5, 2013 at 5:46 pm / Reply

Either of the following should resolve this:
1. Put the JavaScript code at the bottom of the file, as the last item in the closing tag.
2. Put the code inside the jQuery on document ready function:
$(document).ready(function() {
// Add code here
});

peacelion

June 8, 2013 at 9:17 am / Reply

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

$(document).append doesn't work, see : http://stackoverflow.com/questions/15221475/why-document-append-doesnt-work-in-jquery-1-9-1

Richard Bovell (Author)

June 21, 2013 at 2:09 am / Reply

Hi Peacelion,
Thanks much, Mate. Because of your comment, I just learned something new 🙂

I have updated the code accordingly.

Josh

May 9, 2013 at 2:37 pm / Reply

Thank you for the detailed tutorial! What is the plan for part 2? Would love to see more info regarding Meteor. The Telescope guys have their book out now "Discover Meteor" and I'm very interested in the framework.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Richard Bovell (Author)

May 9, 2013 at 5:16 pm / Reply

I plan to write part 2 in a few weeks.

Regarding Meteor, I do plan to write about it as soon as complete developing my first Meteor.js application, which should be in a few weeks.

Waikit

May 9, 2013 at 4:34 pm / Reply

Awesome tutorial, actually pretty much all your posts have been really helpful! Awesome site!

Looking forward to more. Double thumbs up! d ^-^ b

peter

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

May 20, 2013 at 6:20 pm / Reply

ok so am I missing something –
i have handlebars.js registered in the html file before my app.js and in app.js i'm using the template but am getting the Reference Error
ANy idea

Armando Andrade

June 3, 2013 at 7:47 pm / Reply

I'm following everything on the tutorial, but something SUPER weird is happening. After calling the Handlebars.compile(), nothing else gets executed. I even tried with a simple alert('Hi'), and nothing happens. I used bothe handlebars.js and handlebars.runtime.js with the same result. Any help is appreciated, it's driving me nuts.

Richard Bovell (Author)

June 5, 2013 at 11:33 pm / Reply

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Hi Armando, Post your code on JSFiddle or JSBin, so I can take a look and help you out.

Felipe

June 24, 2013 at 1:10 pm / Reply

Nice tutorial!

BTW: South America… not a country…

Richard Bovell (Author)

June 25, 2013 at 1:12 am / Reply

Thanks much, Felipe. good catch. That was embarrassing 🙁
I just changed it.

John Gary

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

June 25, 2013 at 8:13 pm / Reply

Excellent post! It gave me a great understanding of Handlebars in such a short while. Looking forward to your next post!

nicofrand

July 5, 2013 at 5:39 am / Reply

In the « Compare a Non-Handlebars Project With a Handlebars.js Project » part, you are using the shoe name for both the URL and name. However, in the non-handlebar example, you use a lowercase name as URL and it seems you are using the unmodified name in the handlebar example.

Maybe you should show how to lowercase any content in handlebar too to be consistent ? Or maybe I missed something ?

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Chris

July 10, 2013 at 7:25 pm / Reply

Great tutorial, thanks!
I've been meaning to look in to Mustache / Hnadlebars for a while but not found the time. Now I have just started a personal project that I think could potentially benefit from this approach to design and this article has given me everything I need.

Although I have one question and one suggestion…

Q. Why does it compile a function that then has to be called to get the desired output HTML? — Why not just process the data in to the template and return the resulting HTML in one go?

(Okay so that's kind of 2 questions, and of course it's better aimed at the Mustache/Handlebar dev's)

Now my suggestion…and really it was only a thought I had as I read through the article…

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In the section where you compare a non-handlebars and a handlebars solution I though it odd that you had such a horrible mix of JS and HTML (as you state, it's not good). I think the 'proper' JS solution is to use native functions to create the elements and textNodes required in this section. Yes it's a slightly more long winded way of doing things but it's a LOT neater and really quite easy, even without jQuery!

Anyway, just my 2 cents worth.

Thanks again for the detailed and versatile examples. I look forward to reading more of your tutorials!

Boaz

July 21, 2013 at 4:11 pm / Reply

Yet another great article!
thank you so much Richard Bovell

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Richard Bovell (Author)

July 31, 2013 at 1:08 am / Reply

Glad to hear. You are welcome 🙂

Julian Kigwana

July 23, 2013 at 6:12 am / Reply

Thank you for another the great article.
I ran into problems when it came to building Koorchik's custom function for rendering templates. The referenced Stackoverflow thread refers to templating with Underscore, rather than Handlebars and when I add this code into my project, I get an '_' not defined error.
Am I missing something?

Richard Bovell (Author)

July 31, 2013 at 1:42 am / Reply

Julian,

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

As Keith Moore pointed in his comment above, you can use the custom function with Handlebars by adding this bit (I am quoting him here):

> _.templateSettings = {
> interpolate : /\{\{(.+?)\}\}/g
> };
>
> This allows you to do something like this:
> var rendered_html = render('mytemplate', { name: "Keith"});
>
> Where name would be defined in the template as {{name}}

Try this and let me know how it goes.

Dustin

September 17, 2013 at 3:51 pm / Reply

This is great code! I'm having a bit of an issue with

"_.templateSettings" and "_.template" saying the "_" underscore is not defined when I attempt to use it with "handlebars.js". Any quick ideas on how to fix this?

Dustin

September 17, 2013 at 4:02 pm / Reply

I'm also not using underscore.js. hopefully, this can be done without it.

Harry

September 24, 2013 at 7:58 am /

Hi Dustin,

Are you able to solve this issue I am having the same error.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Harry

September 24, 2013 at 9:41 am /

I replaced _.template with
Handlebars.compile(tmpl_string);
It seems working.

YuC

July 25, 2013 at 3:49 am / Reply

Thank you for such a great article, it helps me a lot, waiting for the 2nd part. 🙂

Richard Bovell (Author)

July 31, 2013 at 2:16 am / Reply

I will likely add the second part to the book I am writing.
I am glad to hear the article was helpful.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

the diz

July 25, 2013 at 2:36 pm / Reply

thanks for the post. great explanation of custom helpers. helped me out a lot!

Richard Bovell (Author)

July 31, 2013 at 2:16 am / Reply

Sweet! I am glad to hear it helped out.

Alejandro

September 4, 2013 at 1:52 pm / Reply

Thank you so much, i need this class the examples!

Richard Bovell (Author)

September 4, 2013 at 9:38 pm / Reply

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

I am glad to hear, Alejandro. Good luck.

thetrystero

September 6, 2013 at 9:02 pm / Reply

Under the Little non-Handlebars project, I was unable to get the list to show up in the webpage. I cut and pasted Index.html and main.js exactly as shown. The only difference is that I used jquery-2.0.3.min.js instead of jquery-1.9.1.min.js.

Any ideas as to what might have gone wrong?

Richard Bovell (Author)

September 8, 2013 at 2:25 pm / Reply

I am not sure why it did not work for you. I think it must have been the way you copied it. I just copied the non-handlebars example code and pasted it in JSBin, and here is the working example (I also used jQuery 2.0):

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

http://jsbin.com/ABIBATU/1/edit

thetrystero

September 7, 2013 at 8:22 am / Reply

the version of the example with handlebars does not work either. it returns

The List of Shoes:
â€¨

Richard Bovell (Author)

September 8, 2013 at 2:25 pm / Reply

And here is the handlebars example after I copied and pasted the code in JSBin:
http://jsbin.com/esOsarE/1/edit

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

prateek

September 9, 2013 at 12:29 pm / Reply

Thanks ! Finally an awesome tutorial for Handlebars

DC

October 18, 2013 at 6:18 am / Reply

Great, and comprehensive, tutorial thanks. I was reading it for number 4 of ways to compile templates and the best way to organize the files. So this line was a bit frustrating:
"I do not include an example code to show how to precompile Handlebars template"
It'd be great if you can include a minimal example (without the backbones coding occluding it). Then it truly would the one-stop shop for Handlebars information!

Richard Of Stanley (Author)

October 23, 2013 at 3:16 pm / Reply

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Sorry about that, DC. That does suck. I agree. I must have planned to add that in part 2, which I am yet to complete. I WILL get to part 2 eventually, but it will be after I complete the work I am doing on MDTk.

jagadish

October 25, 2013 at 2:57 am / Reply

Really very usefull article to learn Handlebars.js.

Thanks,
Jagadish N

Leslie

November 3, 2013 at 9:45 pm / Reply

Thanks a lot!

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Nat

December 29, 2013 at 6:41 am / Reply

Awesome article! When's part two coming I'm looking to make some quiz apps 🙂

Richard Of Stanley (Author)

January 7, 2014 at 12:34 am / Reply

I am hopeful I will be able to write part two early this year.

George

December 30, 2013 at 12:29 pm / Reply

Just beginning to read the article and found this paragraph worthy of a "pre-emptive" comment:

"Handlebars expressions are comprised of Handlebars expression and any HTML content or Handlebars

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

expressions within the expression (if the expression is a block)."

Perhaps the above paragraph could be changed or simply removed.

Otherwise, so far so good. 🙂

Richard Of Stanley (Author)

January 7, 2014 at 12:30 am / Reply

Thanks very much, George. I just fixed the issue. I always appreciate it when users take the time to point out typos, grammar errors, bugs, and the like.

jordan

January 9, 2014 at 3:07 am / Reply

It's Jan 2014, did I miss Part 2? Where is it? Or is it not out yet? Anxiously waiting for another great tutorial… nice work!

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

vanessa

January 22, 2014 at 9:57 am / Reply

excellent tutorial… thank you

Manuel

February 7, 2014 at 8:16 pm / Reply

Hey Richard,

Thanks for all the tutorials, it's been a great help!
Just wanted to clarify for everyone that you should add a

in the HTML file, or neither the handlebars or
nonhandlebars shoe project would work!

Manuel

February 7, 2014 at 8:18 pm / Reply

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Hmm, code didn't show, but I was saying you need to add a reference (script src) to the main.js, in your index.html file.

sws

February 13, 2014 at 2:44 pm / Reply

lovely very good tutorial
when can we expect part 2?

MrPfister

February 21, 2014 at 6:26 pm / Reply

Great article! I am a regular here, but I was wondering when part 2 is coming out? I ask, because I am building a web app with require.js backbone and handlebars and I am getting errors when trying to use handlebars. Thought the next issue might solve this problem for me. Thanks!

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Albert

March 7, 2014 at 5:12 am / Reply

Awesome article! Thanks!

Mike

March 11, 2014 at 7:21 am / Reply

Great article! Definitely loving the power of js templating. I have been trying to figure out if handlebars.js does support indirect lookups in some fashion. For example:

context = {
colors: { red: '#ff0000', green: '#00ff00', blue: '#0000ff' },
rainbow: ['red', 'green', 'blue']
}

I can use {{#each rainbow}} and "this" will have the values of red, green, and blue. What if I wanted to output the color code stored in colors? I tried using colors[this], but it just caused a compile error. Is this concept not supported in

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

handlebars.js?

Roman

September 18, 2014 at 2:19 am / Reply

{{lookup ../colors this}}

Piotr

March 25, 2014 at 9:08 am / Reply

Very useful. One thing: the link to main.js is missing in the non-handlebars example index.html as well as Handlebars.js is missing in the handlebars example.

Ashish

March 27, 2014 at 6:54 am / Reply

Really nice article, helped me much. Just one type in when to use section. it should be backbone.js. 🙂

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Akrur

April 10, 2014 at 2:21 am / Reply

Very nice article thanx buddy.. This really helped me alot..

Joe C

April 10, 2014 at 10:42 am / Reply

Is it possible to style the different sections of a handlebars template? For example, if I have a collection of 4 models, how would I use a handlebars template to style the first model for 100% width, the 2nd and 3rd models to have 50% width each, and the 4th to have a 100% width?

Please see https://stackoverflow.com/questions/22990560/is-it-possible-to-style-different-sections-of-handlebars-js-moustache-js-templat for background.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Sandeep

July 24, 2014 at 9:46 am / Reply

This should be the official documentation for Handlebars! :). Thanks.

max

July 27, 2014 at 12:57 pm / Reply

>The output will be the following (if the customerName variable has the value "Richard"):

Richard

won't the output be "Name: Richard"?

Arvind

August 29, 2014 at 2:15 pm / Reply

Great Article!! Cant wait for teh second part.When is it

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

coming?

Leslie C

September 8, 2014 at 6:38 pm / Reply

Great article and very clear! Thank you for the time you put into this and how it's helped people. I've been looking for a comprehensive intro and this was perfect, especially with the pros and cons discussed.

If you have time to polish to perfection:

1) The Custom Function : Change "_.template(tmpl_string)" to "Handlebars.compile(tmpl_string);"

2) On "options.fn" section, I was a bit stuck rereading that section. It helped me once I made a variable name change from "templateWithInterpolatedData" to "templateOutput". Probably obvious to others, but an easy a-ha after realizing it wasn't a handlebars template anymore but output.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Swamp Attack Hack Cheats Tool v 3.04

October 7, 2014 at 2:44 am / Reply

Hi there just wanted to give you a brief heads upp and let you know
a few of the pictures aren't loading correctly.
I'm nnot sure why buut I think its a linking issue.
I'vetried it inn two different internet browsers
and both show the same outcome.

vinay

October 21, 2014 at 5:23 am / Reply

its real proof of concepts, simple yet clear

oscar

October 22, 2014 at 2:59 am / Reply

Hi, I found https://github.com/cultofmetatron/jquery-

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

autobars for load one or more template, but it work fine in order to load a single template into a single file. it don’t work fine if into a file there are more templates… do exist another class that load more templates from one file? thanks

Daniel

November 3, 2014 at 6:26 am / Reply

Custom Function requires to use JQuery $.ajax (So you need to add jQuery as a dependency to your project for that method).

Is there a cleaner way of doing this? Without adding dependencies?

Dave

November 10, 2014 at 2:32 pm / Reply

Hey, Richard.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

I've enjoyed your other tutorials, but I am pretty stumped on this one.

I cannot get the "Here are the three pieces together" part to work. I feel like I have the correct code, but it's not working. Can you show me what the entire HTML page would look like for that demo? Or, tell me what is in the and where you load the external .js file?

p.s. There's a typo at the start of the section where you say "age" and mean "page"

Dave

November 10, 2014 at 2:35 pm / Reply

edit: that should say "Or, tell me what is in the head and where you load the external .js file?"

Chad

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

December 1, 2014 at 2:49 pm / Reply

Your example for using handlebars with an AMD approach has some formatting issues. Here's an updated version:

```
//The define function is part of the AMD mechanism for
loading
define([
'jquery',
'underscore',
'handlebars',
// Require.js text plugin loads the HTML template pages
'text!templates/user_account.html',
'text!templates/user_profile.html'],
function (HandleBars, UserAccount_Template,
UserProfile_Template){
return{
// These are the two objects we will be using
userAccountDataObject : {
accountNumber : "85444",
balance : "$120.00"
},
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
userProfileDataObject : {
firstName : “Michael”,
lastName : “Harrington”
},
// Compile the Handlebars template that we loaded in (the
user_account.html and user_profile.html) and assign them
to the following two variables
userAccount_compiledTemplate :
HandleBars.compile(UserAccount_Template),
userProfile_compiledTemplate :
HandleBars.compile(UserProfile_Template),

// This function will add the Handlebars compiled templates
on the page
render : function (){
// Use the compiled function
(userAccount_compiledTemplate) and pass to it the context
(the data object).
// The result will be the HTML from the template page with
the interpolated values from the data object.
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
this.$(".user-account-
div").html(this.userAccount_compiledTemplate(userAccountDataObject));
// Ditto for this template function
this.$(".user-profile-
div").html(this.userProfile_compiledTemplate(userProfileDataObject));
}
};
});
```

samrat

January 7, 2015 at 12:17 pm / Reply

Nice

Joe

May 25, 2015 at 12:47 pm / Reply

I am late to the party, but still having fun learning hanglebars. It seems handlebars has moved. Adding the -L

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

flag to the download code allows curl to goto the rediretced location. I just learned that flag yesterday 🙂

`curl -L https://github.com/downloads/wycats/handlebars.js/handlebars-1.0.rc.1.min.js > Handlebars.js`

Thomas

May 28, 2015 at 3:16 am / Reply

Hey, Thanks for the awesome article. I have implemented it using the first of four methods you have given here. I would like to implement it using the second method as I have lots of views. I have a question though: What do I place inside the separate HTML files (2nd method) ? Should I be including the script tag there also, or just HTML with handlebar expressions? If you had added the HTML part for the second method or link to an article which does so, will be very helpful for me. Thanks in advance.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

html tutorials

June 22, 2015 at 5:13 am / Reply

Thank you so much, this is what I looking for a long time.
It's useful.
Once again, thank you so much for easy explanation

Khushboo

July 8, 2015 at 2:42 am / Reply

If possible can you please add an article on Angular js.

Drake

August 9, 2015 at 11:49 am / Reply

Good tutorial~ How about the 'Part 2', I cant wait to it~

saurabh

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

October 19, 2015 at 10:35 am / Reply

Awesome…Good work man

zzr994

October 28, 2015 at 12:07 am / Reply

Can you please provide some examples that use events. For example, create a button and provide a click handler.

raza

April 4, 2016 at 3:20 am / Reply

Thanks you so much !
very good article i had a question.
is there any way to concatenate the string with variable like other i.e in angular {{'my name is' name}}
i also tried value="'my name is' {{name}}" but it takes the name as string and out put is "my name is name"

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

sparrow

April 19, 2016 at 6:13 am / Reply

Hi,
As a beginner I couldnt get the "A Little Non-Handlebars Project" working, even after copying it exactly from your example (and no errors in console). Trying the handlebars version i had to change your example from " $(".shoesNav").append (theTemplate(shoesData));" to " $(".shoesNav").append (theTemplateScript(shoesData));" to get it to work – assume that was what it was supposed to be!
Just a thought for any other newbies trying to get it working..
Thanks

Richard Of Stanley (Author)

January 7, 2014 at 12:33 am / Reply

You made my day, Zell 🙂

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Thanks for the information about underscore.js and the gist. I am sure users will find both useful.

Zell Liew

January 7, 2014 at 12:42 am / Reply

No problem!

Looking forward to more great articles and tutorials.

Hope that snippet was useful.

Trackbacks for this post

1. Learn Backbone.js Completely | JavaScript is Sexy
2. Learn Node.js Completely and with Confidence | JavaScript is Sexy
3. How to Learn JavaScript Properly | JavaScript is Sexy
4. Learning JS Properly – End Week 6, Week 7 Assignments (Last week!) | coding(isBananas);

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

5. JS is Sexy- Week 7 & 8 | B's Blog
6. Handlebars.js Tutorial: Learn Everything About …
7. A Thought Experiment with Handlebars.js and Edge Animate
8. Handlebars | Pearltrees
9. □□□□□□□□□JavaScript – □□□ -
10. □□□□□□JavaScript - javascript - □□□□
11. □□□□□□Javascript | 17khba
12. [□] □□□□□□□ JavaScript — □JSER
13. □□□□□□JavaScript — □JSER
14. javascript□□□□ | w3croad
15. □□□□□□JavaScript – web□□□
16. □□□□□□□□□JavaScript - YelloWish
17. □□□□□□□□□JavaScript | □□□
18. Introduction à handlebars | handlebarstutoriel
19. Javascript Template – Handlebar.js | Jay Jin, Programmer&Designer
20. Handlebars | Pearltrees
21. Status and research report | GM Code Blog
22. How To Learn JavaScript Properly — Introduction | My Blog

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Leave a Reply

Name

Email (hidden)

Website

Comment:

Submit Comment

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com