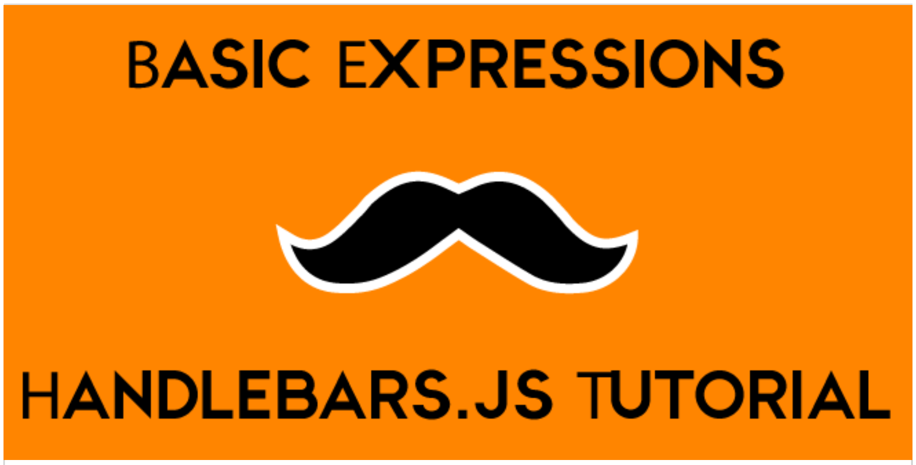


Basic Expressions

BASIC EXPRESSIONS – HANDLEBARS.JS TUTORIAL

Let's kick things off with basic expressions. Expressions are the base of handlebars. They're basically placeholders for your data. You insert them into your template and then handlebars will search and replace for the data that corresponds with your expression. Let me show you how this is done.

Setting Up The HTML Document

Create a file called index.html. Inside this file, add this bit of code.

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Handlebars.js Example Template</title>
        <script type="text/javascript" src="js/handlebars.js"></script>
    </head>
    <body>
    </body>
</html>
```

This is a pretty basic HTML template. The first thing you must do before using handlebars is to include it. You can download the latest version of handlebars.js at the following website

http://handlebarsjs.com/

Once you have it downloaded, save it in a folder called ***js***. Including handlebars is pretty similar to including any other JavaScript library. It's absolutely important that you include this file before you start templating anything.

Inside your ***<body></body>*** tags, insert this bit of code.

```
<div id="menu-placeholder"></div>
<script id="menu-template" type="text/x-handlebars-template">
    <ul>
        <li><a href="{{linkURL1}}">{{linkName1}}</a></li>
        <li><a href="{{linkURL2}}">{{linkName2}}</a></li>
        <li><a href="{{linkURL3}}">{{linkName3}}</a></li>
        <li><a href="{{linkURL4}}">{{linkName4}}</a></li>
        <li><a href="{{linkURL5}}">{{linkName5}}</a></li>
    </ul>
</script>
<script type="text/javascript" src="js/script.js" /></script>
```

```
<script type="text/javascript" src="js/script.js" ></script>
```

There are a couple of things going here. First, we have a ***<div>*** tag with ID ***menu-placeholder***. When our template is finished compiled and processed, then this is where the final output will be displayed. We will soon do this in our script.

Next, we create ***<script>*** tags. There are a couple of attributes we **MUST** define in order for it to work. You must give it an ID. This is so we can grab the template to compile and process. Then, you set the ***type*** attribute to ***text/x-handlebars-template***. By doing this, you're telling your browser that this is not JavaScript code. So, your browser won't bother executing it and displaying it. You also allow handlebars to know that this is a handlebars template.

Inside these tags, you can insert regular HTML, I'll explain what's going on in here later.

Lastly, we include a separate script called ***script.js*** inside our ***js*** folder. Let's create this file now and add this bit of code.

```
// Grab the HTML source that needs to be compiled
var menuSource = document.getElementById( 'menu-template' ).innerHTML;
```

We're grabbing the HTML inside our script. This is what we we would call the source. We need to grab this so we can compile it and process it. Add this next bit of code next.

```
// Compiles the source
var menuTemplate = Handlebars.compile( menuSource );
```

Handlebars is a global object created by the ***handlebars.js*** script when you first linked to it. We can now access it's properties and methods. The first method we access is the ***compile()*** method.This method takes in some HTML code and then translates it into something handlebars can understand and manipulate. In our case, we pass in our source from the script tags.

Creating Basic Expressions

The last thing we need to do is inject our data with the expressions found in our template. Let's take a look at this now. Inside your HTML document, take a look at this code.

```
<script id="menu-template" type="text/x-handlebars-template">
    <ul>
        <li><a href="{{linkURL1}}">{{linkName1}}</a></li>
        <li><a href="{{linkURL2}}">{{linkName2}}</a></li>
        <li><a href="{{linkURL3}}">{{linkName3}}</a></li>
        <li><a href="{{linkURL4}}">{{linkName4}}</a></li>
        <li><a href="{{linkURL5}}">{{linkName5}}</a></li>
    </ul>
</script>
```

Inside our script tags, you just input regular HTML. You don't have to worry about any errors because of the ***type*** attribute. The point of handlebars is to be able to have dynamic content. Inside our HTML, you'll notice we have things like ***{{linkURL1}}*** and ***{{linkName1}}***.

This is what's called an expression in Handlebars. This is most basic form of an expression. Basically, it's a name of a property surrounded by 2 opening and closing curly brackets. Handlebars will look for things like these and then look for a property to replace this expression with.

If you try opening your ***index.html*** file inside your browser, then you won't see anything. Remember, anything placed inside your ***<script>*** tags won't be displayed into your browser. So, we have to process and display it now inside our script.

Add this bit of code next.

```
//Data that will replace the handlebars expressions in our template
var menuData = {
    linkName1 : "Link 1",
    linkName2 : "Link 2",
    linkName3 : "Link 3",
    linkName4 : "Link 4",
    linkName5 : "Link 5",
    linkURL1 : "http://google.com",
    linkURL2 : "http://jaskokoyn.com",
    linkURL3 : "http://yahoo.com",
    linkURL4 : "http://youtube.com",
    linkURL5 : "http://twitter.com"
};
```

We use JSON to contain all our properties. You can use an external JSON file and use an AJAX call to grab it, but we'll keep things simple. This is the data that will replace our expressions in our Handlebars template. Let's add one more line of code.

```
// Process Template with Data
document.getElementById( 'menu-placeholder' ).innerHTML = menuTemplate( menuData );
```

What' were doing here is processing the template with our data. The ***menuTemplate*** becomes a usable function. It has 1 parameter which is the data. Handlebars will then begin the process of updating all the expressions inside our template with the data you provided.

From there, our empty ***<div>*** tag with the ID ***menu-placeholder*** will hold the final result. That's it! You can now refresh your page and see list being displayed with the proper data and links. Let me repeat everything for you so can get a good understand of what's going on.

1. Include handlebars before setting up your template.
2. Create your template and place inside ***<script>*** tags with the ***text/x-handlebars-template*** attribute to prevent your browser from executing it as normal JavaScript and getting any errors.
3. Compile your template into a handlebars template.
4. Create your data and then have Handlebars replace the basic expressions in your template with that data

Now, you may be wondering, why place the templates inside **<script>** tags since it's already basic HTML? Well, here's the thing. This is a pretty simple project, but you may end up having a complex design. Even though handlebars is fast, you don't want to display anything right away until 'it's compiled and processed first. If you do, then your browser will end up displaying the basic expressions before it updates them with the actual values.

Everything here may seem complex, but it really does separate design and code. If you have a designer, all you have to tell them is to place the expressions inside their template and then you can do the behind-the-scenes thing in your code. This way don't have to worry about the HTML markup in the design. It also comes in handy when working with a large project. You should also know you may end up having multiple templates in 1 file. That's fine!

Conclusion

In our next tutorial, we'll focus on executing block codes. They way we're doing things now is actually inefficient and there's a better way to display data with complex expressions.

Download This Free Handlebars.js Example Template Free

SHARE ! Tweet 0 g+1 0 Share Pin it

NEB

User

Group

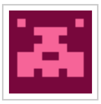
IP Address

Url

Policy

Date

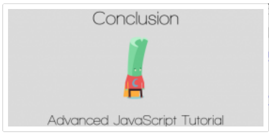
ABOUT JASKO KOYN



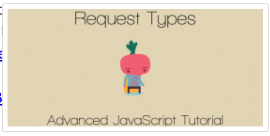
« Previous: External JSON File – JSON Tutorial

Next: Built-in Helpers – Handlebars.js Tutorial »

RELATED ARTICLES



Conclusion – Advanced JavaScript Tutorial
September 6, 2013

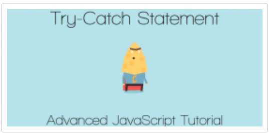


est Types – Advanced ript Tutorial
ber 6, 2013

HTTP Headers – Advanced JavaScript Tutorial
September 6, 2013

AJAX – Advanced JavaSc Tutorial
September 6, 2013

ole Object – Advanced ript Tutorial
ber 5, 2013



Try-catch Statement – Advanced JavaScript Tutorial
September 5, 2013

ONE COMMENT

Steve Lim
February 20, 2014 at 8:07 pm

I just want to say thank you for a well written tutorial. More practical examples would be highly appreciated.

Reply

LEAVE A REPLY

Your email address will not be published. Required fields are marked *

Name *

Email *

Website

Post Comment

A site created by Jasko Koyn