Custom Helpers

CUSTOM HELPERS – HANDLEBARS.JS TUTORIAL

You can extend handlebars with your own custom helpers. This allows you to create complex logic using handlebar's expression system. There are 2 kinds of helpers, a function helper and a block helper. The difference is that one is meant for a single expression and the other uses a block expression.

Custom Function Helpers

Creating function helpers is relatively easy. To create one, we have to register it using the ***registerHelper()*** method. Inside your script, add this bit of code at the very top.

```
//Create a custom function helper to check the status.
Handlebars.registerHelper( "checkStatus", function ( status ){
    if (status == "leaving" )
    {
        return 'leave';
    }
    else
    {
        return 'stay';
    }
});
```

The first parameter is the name of the expression the user must type in order to use this function helper. The second parameter is the function that will execute when the user uses this function helper.

Our function helper has 1 parameter and that's the status of our user. Depending on the status, we'll return different messages. By doing this, we can have custom conditional logic in our templates. You don't have to stick to the ***#if*** conditions. Let's see this in action.

First, let's update our ***menuData*** variable to this.

```
var menuData = {
    name: "jack",
    status: "leaving",
};
```

Next, let's update our template that will use our function helper.

```
<script id="menu-template" type="text/x-handlebars-template">
    Welcome {{name}}, <a href="#">Click here to {{checkStatus status}}</a>
```

ABOUT ME

Hey! I'm Jack and this is my personal blog. I love playing guitar and programming. I like to share my knowledge for free.

Popular Recent Comments Tags

External JSON File – JSON Tutorial
July 24, 2013

Shortcode Attributes – WordPress Plugin Tutorial
March 19, 2013

WordPress Admin Forms – WordPress Plugin Tutorial
March 26, 2013

Getting Started – PHP OOP Tutorial
May 23, 2013

Conclusion – Advanced JavaScript Tutorial
September 6, 2013

```
</script>
```

To use our function helper, we just type in the name in our expression. To pass in the parameters, you just type them in after our function helper. If we had multiple parameters, you would just separate them with spaces.

You can now open the file in our browser and you should see a message. Unlike previous helpers, this is all just one line. Function helpers are just one line of code, but what if we wanted to create a helper for a block of code? This is possible with block helpers.

Custom Block Helpers

Creating custom block helpers is the same process of creating function helpers. First, let's update our template.

```
<script id="menu-template" type="text/x-handlebars-template">
    {{#checkStatus this}}
        You are {{status}}
    {{/checkStatus}}
</script>
```

In order to use block helpers, you must use the ***#*** sign to tell Handlebars you're using a block helper. You'll notice we're using something called ***this***. The ***this*** object is the object you use when you process the template. Then, you give it the name and the parameters. Let's create our custom block helper in JavaScript now. Update the ***checkStatus*** helper to this.

```
Handlebars.registerHelper( "checkStatus", function ( data, options ){
    return options.fn( data.info[0] );
});
```

Creating block helpers is the same for creating function helpers. You use the ***registerHelper()*** method to create one. However, the parameters are a bit different for your function. The first parameter is the object that we need in order to use our template. The next parameter is an object called ***options***.

When we use our ***checkStatus*** block helper, you don't have to pass in the options object. Handlebars will automatically add this for you. Before I explain what this function does, let's update our ***menuData*** variable.

```
var menuData = {
    name: "jack",
    info: [
        { status: "leaving" }
    ]
};
```

You'll notice our status is in an array that's stored in a property called ***info***. If we look at our template, we're directly trying to access ***status***, but we the ***status*** property is nested in our object. So, we'll get an error. Luckily, there's a solution to this.

In our function, the ***options*** object has a method called **fn()**. The ***fn()*** method allows you to change the context of the object temporarily to access a certain property. In our case, we want to access our ***info*** property and use it's objects nested inside. We then return this so our block of code can use it.

Refresh your page and you should be able to see everything working properly.

Conclusion

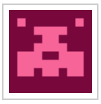
The options object comes with a couple of handy methods. You can check them out at the handlebars website. Found here

http://handlebarsjs.com/

Download This Free Handlebars.js Example Template Source Code Free

SHARE ! Tweet 0 +1 0 Share Pin it

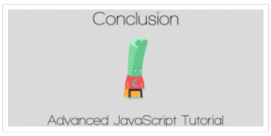
ABOUT JASKO KOYN



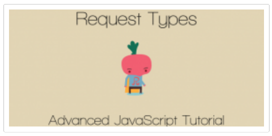
« Previous: Built-in Helpers – Handlebars.js Tutorial

Next: Paths and More Expressions – Handlebars.js Tutorial »

RELATED ARTICLES



Conclusion – Advanced JavaScript Tutorial
September 6, 2013

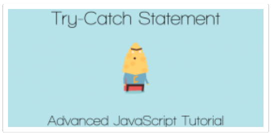


Request Types – Advanced JavaScript Tutorial
September 6, 2013

HTTP Headers – Advanced JavaScript Tutorial
September 6, 2013

AJAX – Advanced JavaScript Tutorial
September 6, 2013

Console Object – Advanced JavaScript Tutorial
September 5, 2013



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

LEAVE A REPLY

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

Name *

Email *

Website

Post Comment