Built-in Helpers

BUILT-IN HELPERS – HANDLEBARS.JS TUTORIAL

Handlebars comes with built-in helpers to make executing complex logic simple. They're conditionals and loops. Let's take a look at some of the built-in helpers that come with it.

#Each Helper

Right now, it's required that we hard code each menu item in our HTML document, but what if the list was updated? Looping through a menu list is easier than typing each name and link. Let's update our template to this.

```
<script id="menu-template" type="text/x-handlebars-template">
    <ul>
        {{#each menu}}
            <li><a href="{{link}}">{{name}}</a></li>
        {{/each}}
    </ul>
</script>
```

Our template was greatly reduced to a few simple lines. The ***#each*** expression is a built-in helper from handlebars. You have to type ***#each*** before the object or array name in order for it to work. To close our loop, you use the same name, but with a forward slash. It basically loops through an array. However, we don't have an array called ***menu***. So, let's update our ***menuData*** variable to this.

```
var menuData = {
    menu: [
        { name: "Link 1", link: "http://google.com" },
        { name: "Link 2", link: "http://jaskokoyn.com" },
        { name: "Link 3", link: "http://youtube.com" },
        { name: "Link 4", link: "http://twitter.com" },
        { name: "Link 5", link: "http://yahoo.com" },
    ]
};
```

We now have a property called ***menu***. This property contains an array of objects. Each time the ***#each*** loop runs, then it'll go through each object. From there, we can then use each property inside each object in the array.

#If Helper

SUBSCRIBE
To RSS Feed
12
Fans
2
Subscribers

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

There are conditional helpers in handlebars. However, they don't work **EXACTLY** like conditional statements like most programming languages. They just check whether or not an object or property is empty. If it is, then the code wrapped inside the ***#if*** block isn't executed. If it is, then the code runs. Let's update our template to this.

```
<script id="menu-template" type="text/x-handlebars-template">
    {{#if menu}}
        <ul>
            {{#each menu}}
                <li><a href="{{link}}">{{name}}</a></li>
            {{/each}}
        </ul>
    {{/if}}
</script>
```

We wrap our ***#each*** loop with the ***#if*** condition. Inside our ***menuData***, we search for the ***menu*** property. Handlebars will look for this property and makes sure it has a value. If you refresh your page, you should see your menu still displaying. There is also an else helper. Let's go to our script and update our ***menuData*** variable to this.

```
var menuData = {};
```

As you can see, we just empty it. Now, let's update our template to use the ***else*** helper.

```
<script id="menu-template" type="text/x-handlebars-template">
    {{#if menu }}
        <ul>
            {{#each menu}}
                <li><a href="{{link}}">{{name}}</a></li>
            {{/each}}
        </ul>
        {{else}}
        There is no menu to display.
    {{/if}}
</script>
```

You'll notice we don't have to type the ***#*** before the else condition. Also, unlike most programming languages, Handlebars requires that the ***else*** condition is placed inside the ***#if*** condition before closing it. This is how handlebars reads our template.

If the ***menu*** property is declared and contains values, then it'll execute our loop, else display that there is no menu.

#unless Helper

The #unless helper works like the ***#if*** helper. Except, it checks for a false value. Let's see how this works. Update your template to this.

```
<script id="menu-template" type="text/x-handlebars-template">
    {{#unless menu }}
        There is no menu to display.
    {{/unless}}
</script>
```

If you refresh your page, then you should see this message displayed. Unlike our ***#if*** condition, the ***#unless*** condition searches for the ***menu*** property. If it doesn't exist, then this block of code executes. If it does find menu, but menu is empty, then it'll also execute this block of code.

#with Helper

The ***#with*** helper is a special kind of helper. In some situations, you may have nested objects. Handlebars handles only the current object. To access objects inside of an object, you would use dot notation. However, the ***#with*** helper can simplify this process. Let's update our template to this.

```
<script id="menu-template" type="text/x-handlebars-template">
    {{#with titles}}
        <h1>{{main}}</h1>
        <h2>{{sub}}</h2>
    {{/with}}
</script>
```

Let's update our ***menuData*** variable to this.

```
var menuData = {
    titles: {
        main: "Main Menu",
        sub: "Sub title"
    },
    menu: [
        { name: "Link 1", link: "http://google.com" },
```

```
        { name: "Link 2", link: "http://jaskokoyn.com" },
        { name: "Link 3", link: "http://youtube.com" },
        { name: "Link 4", link: "http://twitter.com" },
        { name: "Link 5", link: "http://yahoo.com" }
    ]
};
```

As you can see, the ***titles*** property contains an object. To access the properties of this object you would do something like this ***{{titles.main}}***. With the ***#with*** helper however, you just specify the property with the object and then handlebars changes it's scope to that property. Now, you can just access that object's properties by their names instead of using dot notation. It's almost like a loop except for objects and it doesn't bother looping through multiple objects.

Conclusion

The ***#with*** helper is one of those things you won't use often. We'll soon learn about paths and how they can be useful in another tutorial. In the next tutorial, we'll discuss how to create our own custom helpers. More information about Handlebars can be found here.

http://handlebarsjs.com/

Download This Free Handlebars.js Example Template Free

SHARE ! Tweet 0 +1 0 Share

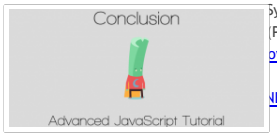
NEB
User
Group
IP Address
Url
Policy
Date
© 2012 | All Rights Reserved by Nebero

ABOUT JASKO KOYN

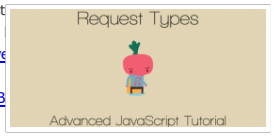
« Previous: Basic Expressions – Handlebars.js Tutorial

Next: Custom Helpers – Handlebars.js Tutorial »

RELATED ARTICLES



Conclusion – Advanced JavaScript Tutorial
September 6, 2013

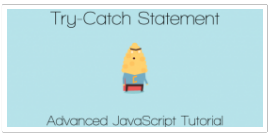


est Types – Advanced cript Tutorial
ber 6, 2013

HTTP Headers – Advanced JavaScript Tutorial
September 6, 2013

AJAX – Advanced JavaSc Tutorial
September 6, 2013

ole Object – Advanced cript Tutorial
ber 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