Handlebars 文档笔记

作者：[GhostBot](http://www.ghostchina.com/author/ghostbot/) • 2014年11月2日

Handlebars 兼容 [Mustache 模板](https://github.com/janl/mustache.js)。

对比了几个 Node.js 常用模板，什么 EJS 、Jade 等等，还是感觉 Handlebars 比较顺手，模板只做数据展示，前端逻辑的东西通过 helper 实现，HTML 中没有掺杂太多 JS 的东西，看起来整洁一些。

Express 中引入 Handlebars 模板的话，需要引入[hbs 模块](https://github.com/donpark/hbs)

**handlebars 表达式**

<h1>{{title}}</h1>

在上下文中找 title 属性，获取它的值

**点分割表达式**

<h1>{{article.title}}</h1>

当前上下文找 article 属性，再找它的 title 属性

标识符可以是除了以下字符以外的 unicode 字符 Whitespace ! “ # % & ‘ ( ) \* + , . / ; < = > @ [ \ ] ^ ` { | } ~

**不合法的标识符用 “[]” 包装**

{{#each articles.[10].[#comments]}}

<h1>{{subject}}</h1>

<div>

{{body}}

</div>

{{/each}}

**不转义**

{{{foo}}}

**Helpers**

0 或多个参数，用空格分割，每个参数是个 handlebars 表达式

{{{link story}}}

link 是 helper 名字，story 是 helper 参数。

**注册 helper**

Handlebars.registerHelper('link', function(object) {

return new Handlebars.SafeString(

"<a href='" + object.url + "'>" + object.text + "</a>"

);

});

helper 返回 HTML ，不想被转义，用 Handlebars.SafeString()。

helper 把接收的上下文作为 this 上下文

<ul>

{{#each items}}

<li>{{agree\_button}}</li>

{{/each}}

</ul>

**上下文和 helper：**

var context = {

items: [

{name: "Handlebars", emotion: "love"},

{name: "Mustache", emotion: "enjoy"},

{name: "Ember", emotion: "want to learn"}

]

};

Handlebars.registerHelper('agree\_button', function() {

return new Handlebars.SafeString(

"<button>I agree. I " + this.emotion + " " + this.name + "</button>"

);

});

输出结果：

<ul>

<li><button>I agree. I love Handlebars</button></li>

<li><button>I agree. I enjoy Mustache</button></li>

<li><button>I agree. I want to learn Ember</button></li>

</ul>

也可以直接传字符串参数

{{{link "See more..." story.url}}}

等价于

{{{link story.text story.url}}}

Handlebars.registerHelper('link', function(text, url) {

return new Handlebars.SafeString(

"<a href='" + url + "'>" + text + "</a>"

);

});

helper 最后一个参数也可以接收可选的键值对序列（文档提到的 hash 参数）

{{{link "See more..." href=story.url class="story"}}}

hash 参数的 key 必须是简单的标识符，value 是 Handlebars 表达式，value 可以是简单的标识符，路径，或者字符串。

Handlebars.registerHelper('link', function(text, options) {

var attrs = [];

for(var prop in options.hash) {

attrs.push(prop + '="' + options.hash[prop] + '"');

}

return new Handlebars.SafeString(

"<a " + attrs.join(" ") + ">" + text + "</a>"

);

});

基础 Blocks

<div class="entry">

<h1>{{title}}</h1>

<div class="body">

{{#noop}}{{body}}{{/noop}}

</div>

</div>

Handlebars.registerHelper('noop', function(options) {

return options.fn(this);

});

noop helper 实际跟没有 helper 类似，只是传递上下文，返回字符串。Handlebars 把当前的上下文作为 this 。

内建 helper

**with helper**

根据模板传递的上下文解析模板

<div class="entry">

<h1>{{title}}</h1>

{{#with story}}

<div class="intro">{{{intro}}}</div>

<div class="body">{{{body}}}</div>

{{/with}}

</div>

当 JSON 对象包含嵌套属性时，不必再三重复父属性的名字。比如以下数据：

{

title: "First Post",

story: {

intro: "Before the jump",

body: "After the jump"

}

}

helper 接收参数，参数为 JSON 属性的 上下文。

Handlebars.registerHelper('with', function(context, options) {

return options.fn(context);

});

**简单迭代器 each helper**

Handlebars 内建了　each　迭代器

<div class="comments">

{{#each comments}}

<div class="comment">

<h2>{{subject}}</h2>

{{{body}}}

</div>

{{/each}}

</div>

实现原理如下：　把 comments 数组的每一个元素作为上下文解析模板

Handlebars.registerHelper('each', function(context, options) {

var ret = "";

for(var i=0, j=context.length; i<j; i++) {

ret = ret + options.fn(context[i]);

}

return ret;

});

可以用 this 引用迭代元素

<ul class="people\_list">

{{#each people}}

<li>{{this}}</li>

{{/each}}

</ul>

上下文：

{

people: [

"Yehuda Katz",

"Alan Johnson",

"Charles Jolley"

]

}

结果：

<ul class="people\_list">

<li>Yehuda Katz</li>

<li>Alan Johnson</li>

<li>Charles Jolley</li>

</ul>

当某一项为空时，可以用

{{else}}

表达式

{{#each paragraphs}}

<p>{{this}}</p>

{{else}}

<p class="empty">No content</p>

{{/each}}

通过

{{@index}}

可以引用当前的循环索引

{{#each array}}

{{@index}}: {{this}}

{{/each}}

用

{{@key}}

引用当前的键名：

{{#each object}}

{{@key}}: {{this}}

{{/each}}

数组迭代的第一步和最后一步用 @first 和 @last 变量表示， 对象迭代时仅 @first可用。

**条件语句 if helper**

如果条件参数返回 false, undefined, null, "" 或 []（非真的值）时，Handlebars 将不渲染该块

Handlebars 内建了 if 和 unless 语句

{{#if isActive}}

<img src="star.gif" alt="Active">

{{/if}}

实现原理：根据传入的条件参数，判断是否解析模板

Handlebars.registerHelper('if', function(conditional, options) {

if(conditional) {

return options.fn(this);

}

});

Handlebars 还提供了 else 语句

{{#if isActive}}

<img src="star.gif" alt="Active">

{{else}}

<img src="cry.gif" alt="Inactive">

{{/if}}

**unless helper**

unless 跟 if 正好相反，如果表达式返回 false ，模板将被渲染。

<div class="entry">

{{#unless license}}

<h3 class="warning">WARNING: This entry does not have a license!</h3>

{{/unless}}

</div>

当 license 返回 false，Handlebars 将渲染 warning 。

**log helper**

记录上下文状态

{{log "Look at me!"}}

**JavaScript 编译模板**

模板可以包含在特殊的<script> 里：

<script id="entry-template" type="text/x-handlebars-template">

template content

</script>

然后用 Handlebars.compile 编译模板

var source = $("#entry-template").html();

var template = Handlebars.compile(source);

获取编译后的 HTML 模板，用 JSON 数据填充模板

var context = {title: "My New Post", body: "This is my first post!"}

var html = template(context);

最终结果：

<div class="entry">

<h1>My New Post</h1>

<div class="body">

This is my first post!

</div>

</div>

**HTML 转义**

不想转义用

{{{

模板：

<div class="entry">

<h1>{{title}}</h1>

<div class="body">

{{{body}}}

</div>

</div>

上下文数据：

{

title: "All about <p> Tags",

body: "<p>This is a post about <p> tags</p>"

}

最终结果：

<div class="entry">

<h1>All About <p> Tags</h1>

<div class="body">

<p>This is a post about <p> tags</p>

</div>

</div>

Handlebars.SafeString 方法不做转义，通常返回 new Handlebars.SafeString(result)。此种情形，你可能想手动转义参数：

Handlebars.registerHelper('link', function(text, url) {

text = Handlebars.Utils.escapeExpression(text);

url = Handlebars.Utils.escapeExpression(url);

var result = '<a href="' + url + '">' + text + '</a>';

return new Handlebars.SafeString(result);

});

模板注释

{{! }}

或者

{{!-- --}}

<div class="entry">

{{! only output this author names if an author exists }}

{{#if author}}

<h1>{{firstName}} {{lastName}}</h1>

{{/if}}

</div>

模板注释不会输出，HTML 注释会输出

<div class="entry">

{{! This comment will not be in the output }}

<!-- This comment will be in the output -->

</div>

**Partials 局部模板**

用

{{> partialName}}

引入局部模板，局部模板可以使字符串，也可以是编译模板的函数。

var source = "<ul>{{#people}}<li>{{> link}}</li>{{/people}}</ul>";

Handlebars.registerPartial('link', '<a href="/people/{{id}}">{{name}}</a>')

var template = Handlebars.compile(source);

var data = { "people": [

{ "name": "Alan", "id": 1 },

{ "name": "Yehuda", "id": 2 }

]};

template(data);

// Should render:

// <ul>

// <li><a href="/people/1">Alan</a></li>

// <li><a href="/people/2">Yehuda</a></li>

// </ul>

内建工具

**转义字符串**

Handlebars.Utils.escapeExpression(string)

**判断空值**

Handlebars.Utils.isEmpty(value)

**扩展对象**

Handlebars.Utils.extend(foo, {bar: true})

**转字符串**

Handlebars.Utils.toString(obj)

**判断数组**

Handlebars.Utils.isArray(obj)

**判断函数**

Handlebars.Utils.isFunction(obj)