Handlebars.js 模板引擎

介绍

Handlebars 是 JavaScript 一个语义模板库，通过对 view 和 data 的分离来快速构建 Web 模板。它采用"Logic-less template"（无逻辑模版）的思路，在加载时被预编译，而不是到了客户端执行到代码时再去编译， 这样可以保证模板加载和运行的速度。Handlebars 兼容 Mustache，你可以在 Handlebars 中导入 Mustache 模板。

使用与安装

Handlebars 的安装非常简单，你只需要从 Github 下载最新版本，你也可访问下面网址获取最新信息：http://handlebarsjs.com。
目前 handlebars.js 已经被许多项目广泛使用了，handlebars 是一个纯 JS 库，因此你可以像使用其他 JS 脚本一样用 script 标签来包含 handlebars.js

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

基本语法

Handlebars expressions 是 handlebars 模板中最基本的单元，使用方法是加两个花括号`{{value}}`, handlebars 模板会自动匹配相应的数值，对象甚至是函数。
例如：

```
<div class="demo">

  <h1>{{name}}</h1>

  <p>{{content}}</p></div>
```

你可以单独建立一个模板,ID（或者 class）和 type 很重要，因为你要用他们来获取模板内容 例如：

```
<script id="tpl" type="text/x-handlebars-template">  <div class="demo">

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

       <p>{{content.title}}</p>
```

```
    </div></script>
```

handlebars 会根据上下文来自动对表达式进行匹配，如果匹配项是个变量，则会输出变量的值，如果匹配项是个函数，则函数会被调用。
如果没找到匹配项，则没有输出。表达式也支持点操作符，因此你可以使用`{{content.title}}`这样的形式来调用嵌套的值或者方法， handlebars 会根据当前上下文输出 content 变量的 title 属性的值。

在 JavaScript 中，使用 `Handlebars.compile()`方法来预编译模板 例如：(这是一套规则)

```
    //用 jquery 获取模板

    var tpl   =  $("#tpl").html();

    //原生方法

    var source = document.getElementById('#tpl').innerHTML;

    //预编译模板

    var template = Handlebars.compile(source);

    //模拟 json 数据

    var context = { name: "zhaoshuai", content: "learn Handlebars"};

    //匹配 json 内容

    var html = template(context);

    //输入模板

    $(body).html(html);
```

Handlebar 的表达式

Block 表达式

有时候当你需要对某条表达式进行更深入的操作时，Blocks 就派上用场了，在 Handlebars 中，你可以在表达式后面跟随一个#号来表示 Blocks，然后通过`{{/表`

达式}}来结束 Blocks。 如果当前的表达式是一个数组，则 Handlebars 会“自动展开数组”，并将 Blocks 的上下文设为数组中的元素。 例如：

```
<ul>

{{#programme}}

    <li>{{language}}</li>

{{/programme}}</ul>
```

有以下 json 数据

```
{

  programme: [

    {language: "JavaScript"},

    {language: "HTML"},

    {language: "CSS"}

  ]

}
```

编译模板代码同上…… 上面的代码会自动匹配 `programme` 数据并展开数据，渲染 DOM 后就是这样的

```
<ul>

  <li>JavaScript</li>

  <li>HTML</li>

  <li>CSS</li></ul>
```

Handlebars 的内置块表达式（Block helper）

1.each block helper

你可以使用内置的`{{#each}}` helper 遍历列表块内容，用 `this` 来引用遍历的元素
例如：

```
<ul>

    {{#each name}}

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

    {{/each}}</ul>
```

对应适用的 json 数据

```
{

    name: ["html","css","javascript"]

};
```

这里的 `this` 指的是数组里的每一项元素，和上面的 Block 很像，但原理是不一样的这里的 name 是数组，而内置的 each 就是为了遍历数组用的，更复杂的数据也同样适用。

2.if else block helper

`{{#if}}`就你使用 JavaScript 一样，你可以指定条件渲染 DOM，如果它的参数返回 `false，undefined，null，"" 或者 [] (a "falsy" value)`, Handlebar 将不会渲染 DOM，如果存在`{{#else}}`则执行`{{#else}}`后面的渲染
例如：

```
{{#if list}}

<ul id="list">

    {{#each list}}

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

    {{/each}}

</ul>
```

```
{{else}}

    <p>{{error}}</p>

{{/if}}
```

对应适用 json 数据

```
var data = {

    info:['HTML5','CSS3',"WebGL"],

    "error":"数据取出错误"}
```

这里`{{#if}}`判断是否存在 list 数组，如果存在则遍历 list，如果不存在输出错误信息

3.unless block helper

`{{#unless}}`这个语法是反向的 if 语法也就是当判断的值为 `false` 时他会渲染 DOM 例如：

```
{{#unless data}}<ul id="list">

    {{#each list}}

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

    {{/each}}</ul>

{{else}}

    <p>{{error}}</p>

{{/unless}}
```

4.with block helper

`{{#with}}`一般情况下，Handlebars 模板会在编译的阶段的时候进行 context 传递和赋值。使用 with 的方法，我们可以将 context 转移到数据的一个 section 里面（如果你的数据包含 section）。 这个方法在操作复杂的 template 时候非常有用。

```
<div class="entry">
```

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

  {{#with author}}

  <h2>By {{firstName}} {{lastName}}</h2>

  {{/with}}</div>
```

对应适用 json 数据

```
{

  title: "My first post!",

  author: {

    firstName: "Charles",

    lastName: "Jolley"

  }

}
```

Handlebar 的注释（comments）

Handlebars 也可以使用注释写法如下

```
{{! handlebars comments }}
```

Handlebars 的访问（Path）

Handlebar 支持路径和 `mustache`,Handlebar 还支持嵌套的路径，使得能够查找嵌套低于当前上下文的属性
可以通过`.`来访问属性也可以使用`../`,来访问父级属性。 例如:（使用`.`访问的例子）

```
<h1>{{author.id}}</h1>
```

对应 json 数据

```
{
```

```
  title: "My First Blog Post!",

  author: {

    id: 47,

    name: "Yehuda Katz"

  },

  body: "My first post. Wheeeee!"

  };
```

例如：（使用`../`访问）

```
{{#with person}}

    <h1>{{../company.name}}</h1>

{{/with}}
```

对应适用 json 数据

```
{

    "person":

    { "name": "Alan" },

        company:

    {"name": "Rad, Inc." }

};
```

自定义 helper

Handlebars，可以从任何上下文可以访问在一个模板，你可以使用`Handlebars.registerHelper()`方法来注册一个 helper。

调试技巧

把下面一段"debug helper"加载到你的 JavaScript 代码里,然后在模板文件里通过`{{debug}}`或是`{{debug someValue}}`方便调试数据

```
Handlebars.registerHelper("debug", function(optionalValue) {

  console.log("Current Context");

  console.log("====================");

  console.log(this);

  if (optionalValue) {

    console.log("Value");

    console.log("====================");

    console.log(optionalValue);

  }

});
```

handlebars 的 jquery 插件

```
(function($) {

    var compiled = {};

    $.fn.handlebars = function(template, data) {

        if (template instanceof jQuery) {

            template = $(template).html();

        }

    compiled[template] = Handlebars.compile(template);

    this.html(compiled[template](data));

    };
```

```
})(jQuery);

$('#content').handlebars($('#template'), { name: "Alan" });
```