参考：<https://segmentfault.com/a/1190000008367747>

[handlebars.js模板引擎](https://segmentfault.com/a/1190000008367747)

* 623 次浏览

前言：常用的模板引擎有很多，但写法都大同小异。  
handlebars.js就是一个纯JS库，因此你可以像其他脚本一样用script包起来。调用内部封装好的功能。

1.基于Mustache，可以在handelbars中导入Mustache模板。  
利用{{ 对象 函数 }} 替换对象 或者 运行函数  
支持.点语法 可以对象.title等属性值  
2.使用时，直接script标签引入handlebars.js文件。

首先github上下载自新版本的handelbars.js [http://handlebarsjs.com](http://handlebarsjs.com/)

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

<script type="text/javascript" src="script/handlebars-1.0.0.beta.6.js"></script>

3.基本语法  
handlebarsjs 是模块中的最基本的单元，使用时用两个花括号{{ }} 包裹。eg：{{ value }} ,handlebars模块会自动匹配相应的数值，对象或者是函数。

<div class="demo">

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

<p>{{content}}</p>

</div>

也可以单独建立一个模板，  
id（或者class）可以用来唯一确定一个模板，type是固定写法，不可或缺。

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

<div class="demo">

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

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

</div>

</script>

4.js 中使用handelbars.compile（）预编译模板 ：

//用jquery获取模板, 使用其他方式也可以，只要是获取到模板的源代码就可以

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);

//输出模板

$(body).html(html);

5.blocks表达式

* 表达式的{{后面跟一个# 表示blocks
* {{ 表达式 }} 结束blocks
* 如果当前表达式是数组则handelbars会展开数组。并将blocks的上下文设为数组元素）

比如：

<ul>

{{#programme}}

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

{{/programme}}

</ul>

对应json数据：

{

programme: [

{language: "JavaScript"},

{language: "HTML"},

{language: "CSS"}

]

}

渲染后：

<ul>

<li>JavaScript</li>

<li>HTML</li>

<li>CSS</li>

</ul>

6.Handelbars内置表达式（Block helper）

* each：

利用{{#each name}}来遍历列表块的内容，  
用this来引用遍历的元素，指数组里的每一个元素。 name 是数组。

<ul>

{{#each name}}

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

{{/each}}

</ul>

对应json是：

{

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

};

编译后：

<ul>

<li>JavaScript</li>

<li>HTML</li>

<li>CSS</li>

</ul>

if else :

指定条件渲染dom；如果 {{ #if list }}，即if后的参数存在， 则渲染{{ #else }}后面的语句；否则将不会渲染都dom，将执行{{ else }}后的error语句；

{{#if list}}

<ul id="list">

{{#each list}}

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

{{/each}}

</ul>

{{else}}

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

{{/if}}

对应的json：

var data = {

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

"error":"数据取出错误"

}

unless

* {{ #**unless** }}反向的一个**if**语句；**unless**后的参数不存在 为false时，渲染dom;

with

{{#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。  
即：在使用json数据较为复杂时，我们用这种方式来确定模板里填写的内容是json对象的哪一个部分！

<div class="entry">

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

{{#with author}}

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

{{/with}}

</div>

对应json数据：

{

title: "My first post!",

author: {

firstName: "Charles",

lastName: "Jolley"

}

}

7.handlebar的注释（comments）

7.handlebar的注释（comments）

写法：

{{! handlebars comments }}

8.handlebar的访问（path）

可以通过 . 语法访问子属性；  
也可以通过 ../ 来访问父级属性。

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

{{#with person}}

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

{{/with}}

9.自定义helper

用Handlebars.registerHelper ( )方法来注册一个helper

10.handelbars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" });