Juicer 中文文档

当前最新版本**: 0.6.8-stable**

Juicer 是一个高效、轻量的前端 (Javascript) 模板引擎，使用 Juicer 可以是你的代码实现数据和视图模型的分离(MVC)。 除此之外，它还可以在 Node.js 环境中运行。

你可以在遵守 MIT Licence 的前提下随意使用并分发它。Juicer 代码完全开源并托管在 Github 上，如果你在使用的过程中发现 什么 Bug 抑或是一些好的建议都欢迎在 Github Issue 上提交。

名字的由来

倘若我们把数据比作新鲜可口的水果，把模板看做是水，Juicer 就是把水果和水榨出我们需要的HTML代码片段的榨汁机。

Juicer 的引入

```
<script type="text/javascript" src="juicer-min.js"></script>
```

* 使用方法

> 编译模板并根据所给的数据立即渲染出结果.

```
juicer(tpl, data);
```

> 仅编译模版暂不渲染，它会返回一个可重用的编译后的函数.

```
var compiled_tpl = juicer(tpl);
```

> 根据给定的数据，对之前编译好的模板进行数据渲染.

```
var compiled_tpl = juicer(tpl);
var html = compiled_tpl.render(data);
```

> 注册/注销自定义函数（对象），在下边 ${变量} 中会有实例.

```
juicer.register('function_name', function);
juicer.unregister('function_name');
```

> 自定义模板语法边界符，下边是 Juicer 默认的边界符。你可以借此解决 Juicer 模板语法同某些后端语言模板语法冲突的情况.

```
juicer.set({
    'tag::operationOpen': '{@',
    'tag::operationClose': '}',
    'tag::interpolateOpen': '${',
    'tag::interpolateClose': '}',
    'tag::noneencodeOpen': '$${',
    'tag::noneencodeClose': '}',
    'tag::commentOpen': '{#',
    'tag::commentClose': '}'
});
```

默认参数配置

```
{
    cache:          true [false],
    strip:          true [false],
    errorhandling:  true [false],
    detection:      true [false]
}
```

默认配置是 Juicer 推荐的使用方式，如果你使用过程中的确需要更改这些参数，可以这么做：

逐条参数更改：

```
juicer.set('strip',false);
juicer.set('cache',false);
```

批量参数更改：

```
juicer.set({
    'strip': false,
    'cache': false
};
```

Juicer 默认会对编译后的模板进行缓存，从而避免同一模板多次数据渲染时候重复编译所耗的时间， 如无特殊需要，强烈不建议关闭默认参数中的 cache，这么做将会令 Juicer 缓存失效从而降低性能.

* 语法

a. ${变量}

使用 ${} 输出变量值，其中 _ 为对数据源的引用（如 ${_}，常用于数据源为数组的情况）。支持自定义函数（通过自定义函数你可以实现很多有趣的功能，类似 ${data|links} 就可以 通过事先定义的自定义函数 links 直接对 data 拼装出<a href=".." alt=".." />）.

```
${name}
${name|function}
${name|function, arg1, arg2}
```

让我们通过一个例子演示一下自定义函数的奇妙用法吧.

```
var json = {
    links: [
        {href: 'http://juicer.name', alt: 'Juicer'},
        {href: 'http://benben.cc', alt: 'Benben'},
        {href: 'http://ued.taobao.com', alt: 'Taobao UED'}
    ]
};

var tpl = [
    '{@each links as item}',
        '${item|links_build} <br />',
    '{@/each}'
].join('');

var links = function(data) {
        return '<a href="' + data.href + '" alt="' + data.alt + '" />';
};

juicer.register('links_build', links); //注册自定义函数
juicer(tpl, json);
```

上述代码执行后我们会发现结果是这样的：

```
&lt;a href=&quot;http://juicer.name&quot; alt=&quot;Juicer&quot; <br />
&lt;a href=&quot;http://benben.cc&quot; alt=&quot;Benben&quot; <br />
&lt;a href=&quot;http://ued.taobao.com&quot; alt=&quot;Taobao UED&quot; <br />
```

可以看得出，结果被转义了，如果我们上边使用 $${item|links} 就会得到我们预期的结果，这就是下边即将提到的避免转义。

转义/避免转义

出于安全角度的考虑，${变量} 在输出之前会对其内容进行转义，如果你不想输出结果被转义，可以使用 $${变量} 来避免这种情况。 例如：

```
var json = {
    value: '<strong>juicer</strong>'
};

var escape_tpl='${value}';
var unescape_tpl='$${value}';

juicer(escape_tpl, json); //输出 '&lt;strong&gt;juicer&lt;/strong&gt;'
juicer(unescape_tpl, json); //输出 '<strong>juicer</strong>'
```

b. 内联辅助函数 **{@helper} ... {@/helper}**

如果你需要在页面或者模板内定义辅助函数，可以像这样使用 `helper`，同时支持Node和Browser.

```
<!--
{@helper numberPlus}
    function(number) {
        return number + 1;
    }
{@/helper}
-->
```

Javascript 代码：

```
var tpl = 'Number: ${num|numberPlus}';

juicer(tpl, {
    num: 123
});

//输出 Number: 124
```

c. 循环遍历 **{@each} ... {@/each}**

如果你需要对数组进行循环遍历的操作，就可以像这样使用 `each` .

```
{@each list as item}
        ${item.prop}
{@/each}
```

如果遍历过程中想取得当前的索引值，也很方便.

```
{@each list as item, index}
        ${item.prop}
        ${index} //当前索引
{@/each}
```

d. 判断 **{@if} ... {@else if} ... {@else} ... {@/if}**

我们也会经常碰到对数据进行逻辑判断的时候.

```
{@each list as item,index}
    {@if index===3}
        the index is 3, the value is ${item.prop}
    {@else if index === 4}
        the index is 4, the value is ${item.prop}
    {@else}
```

```
        the index is not 3, the value is ${item.prop}
    {@/if}
{@/each}
```

e. 注释 {# 注释内容}

为了后续代码的可维护性和可读性，我们可以在模板中增加注释.

```
{# 这里是注释内容}
```

f. 辅助循环 {@each i in range(m, n)}

辅助循环是 Juicer 为你精心设置的一个语法糖，也许你会在某种情境下需要它.

```
{@each i in range(5, 10)}
    ${i}; //输出 5;6;7;8;9;
{@/each}
```

g. 子模板嵌套 {@include tpl, data}

子模板嵌套可以让你更灵活的组织你的模板代码，除了可以引入在数据中指定的子模板外，当然你也可以通过指定字符串`#id`使用写在`script`标签中的模板代码.

HTML代码：

```
<script type="text/juicer" id="subTpl">
        I'm sub content, ${name}
</script>
```

Javascript 代码：

```
var tpl = 'Hi, {@include "#subTpl", subData}, End.';

juicer(tpl, {
        subData: {
                name: 'juicer'
        }
});

//输出 Hi, I'm sub content, juicer, End.
//或者通过数据引入子模板，下述代码也将会有相同的渲染结果：

var tpl = 'Hi, {@include subTpl, subData}, End.';

juicer(tpl, {
    subTpl: "I'm sub content, ${name}",
    subData: {
        name: 'juicer'
    }
});
```

* 在 **Node.js** 环境中运行

```
在命令行中执行:
npm install juicer

在代码中这么引入:
var juicer = require('juicer');
var html = juicer(tpl, data);
```

在 **express.js** 框架中使用

在 express 2.x 系列版本中：

```
npm install juicer
var juicer = require('juicer');
app.set('view engine', 'html');
app.register('.html', {
    compile: function(str, options) {
        return juicer.compile(str, options).render;
    }
});
```

在 express 3.x 系列版本中：

```
npm install juicer
var juicer = require('juicer');
var fs = require('fs');
app.set('view engine', 'html');
app.engine('html', function(path, options, fn){
    fs.readFile(path, 'utf8', function(err, str){
        if (err) return fn(err);
        str = juicer(str, options);
        fn(null, str);
    });
});
```

或者也可以借助 juicer-express-adapter 中间件在Node端使用Juicer以及简单的include功能。

```
npm install juicer-express-adapter
```

在命令行预编译模板文件：

```
npm install -g juicer
juicer example.juicer.tmpl -f example.js

// type `juicer` after install for more help.
// 全局模式安装 `juicer` 后，在命令行下输入 `juicer` 可以获得更多帮助信息。
```

* 一个完整的例子

```
HTML 代码:

<script id="tpl" type="text/template">
    <ul>
        {@each list as it,index}
            <li>${it.name} (index: ${index})</li>
        {@/each}
        {@each blah as it}
            <li>
                num: ${it.num} <br />
                {@if it.num==3}
                    {@each it.inner as it2}
                        ${it2.time} <br />
                    {@/each}
                {@/if}
            </li>
        {@/each}
    </ul>
</script>
```

Javascript 代码:

```
var data = {
    list: [
        {name:' guokai', show: true},
        {name:' benben', show: false},
        {name:' dierbaby', show: true}
    ],
    blah: [
        {num: 1},
        {num: 2},
        {num: 3, inner:[
            {'time': '15:00'},
            {'time': '16:00'},
            {'time': '17:00'},
            {'time': '18:00'}
        ]},
        {num: 4}
    ]
};

var tpl = document.getElementById('tpl').innerHTML;
var html = juicer(tpl, data);
```