目录 Contents

4. 模板引擎的基本概念

4.1 渲染UI结构时遇到的问题

```
var rows = []
$.each(res.data, function (i, item) { // 循环拼接字符串
    rows.push('<li class="list-group-item">'+ item.content +'<span class="badge
cmt-date">评论时间：'+ item.time +'</span><span class="badge cmt-person">评论人：'+
item.username +'</span></li>')
})
$('#cmt-list').empty().append(rows.join('')) // 渲染列表的UI结构
```

上述代码是通过**字符串拼接**的形式，来渲染UI结构。

如果UI结构比较复杂，则拼接字符串的时候需要格外注意**引号之前的嵌套**。且一旦需求发生变化，**修改起来也非常麻烦**。

4. 模板引擎的基本概念

4.2 什么是模板引擎

模板引擎，顾名思义，它可以根据程序员指定的模板结构和数据，自动生成一个完整的HTML页面。

4. 模板引擎的基本概念

4.3 模板引擎的好处

① 减少了字符串的拼接操作

② 使代码结构更清晰

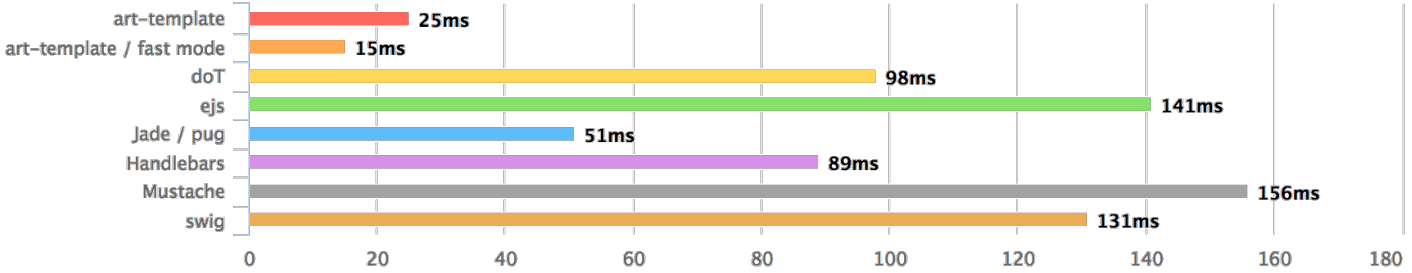
③ 使代码更易于阅读与维护

目录 Contents

5. art-template模板引擎

5.1 art-template简介

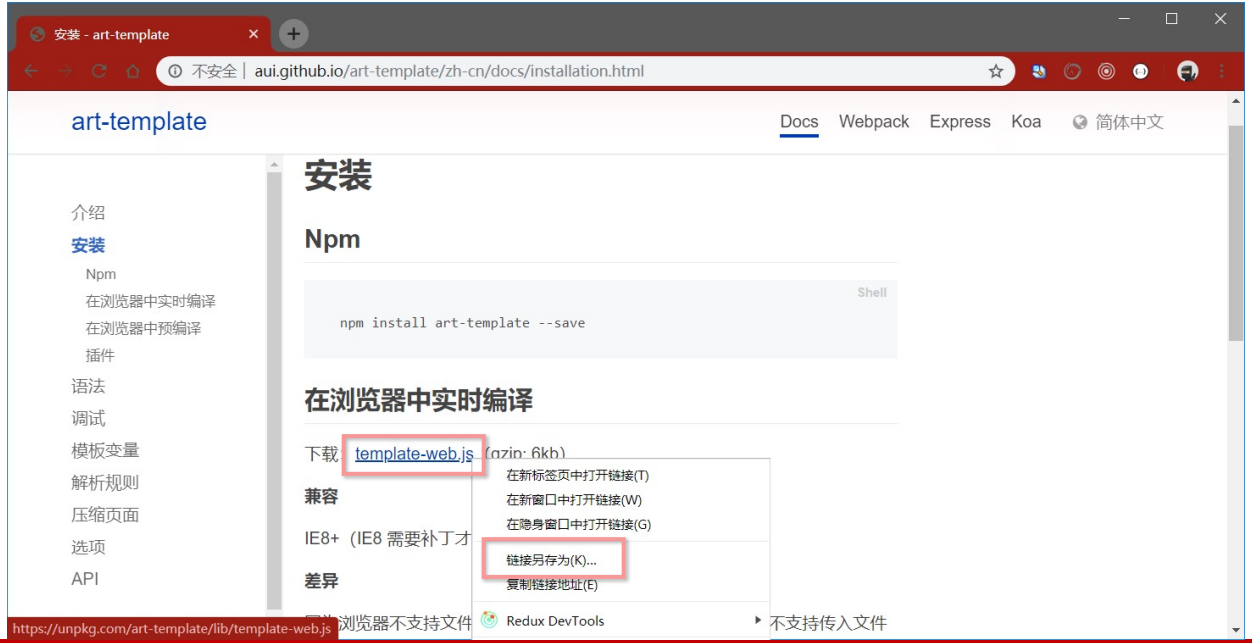
art-template 是一个简约、超快的模板引擎。中文官网首页为 http://aui.github.io/art-template/zh-cn/index.html



5. art-template模板引擎

5.2 art-template的安装

在浏览器中访问 http://aui.github.io/art-template/zh-cn/docs/installation.html 页面，找到下载链接后，鼠标右键，选择“**链接另存为**”，将 art-template 下载到本地，然后，通过 <script> 标签加载到网页上进行使用。



5.3 art-template模板引擎的基本使用

1. 使用传统方式渲染UI结构

```
var data = {
    title: '<h3>用户信息</h3>',
    name: 'zs',
    age: 20,
    isVIP: true,
    regTime: new Date(),
    hobby: ['吃饭', '睡觉', '打豆豆']
}
```

用户信息

姓名：zs
年龄：20
会员：否
注册时间：2019-10-28
爱好：

- 吃饭
- 睡觉
- 打豆豆

5.3 art-template模板引擎的基本使用

2. art-template的使用步骤

① 导入 art-template

② 定义数据

③ 定义模板

④ 调用 template 函数

⑤ 渲染HTML结构

5.4 art-template标准语法

1. 什么是标准语法

art-template 提供了 **{{ }}** 这种语法格式，在 {{ }} 内可以进行**变量输出**，或**循环数组**等操作，这种 {{ }} 语法在 art-template 中被称为标准语法。

5.4 art-template标准语法

2. 标准语法 - 输出

```
{{value}}
{{obj.key}}
{{obj['key']}}
{{a ? b : c}}
{{a || b}}
{{a + b}}
```

在 {{ }} 语法中，可以进行变量的输出、对象属性的输出、三元表达式输出、逻辑或输出、加减乘除等表达式输出。

5.4 art-template标准语法

3. 标准语法 – 原文输出

```
{{@ value }}
```

如果要输出的 value 值中，包含了 HTML 标签结构，则需要使用**原文输出**语法，才能保证 HTML 标签被正常渲染。

5.4 art-template标准语法

4. 标准语法 – 条件输出

如果要实现条件输出，则可以在 {{ }} 中使用 **if** ... **else if** ... **/if** 的方式，进行按需输出。

```
{{if value}} 按需输出的内容 {{/if}}

{{if v1}} 按需输出的内容 {{else if v2}} 按需输出的内容 {{/if}}
```

5.4 art-template标准语法

5. 标准语法 – 循环输出

如果要实现循环输出，则可以在 {{ }} 内，通过 each 语法循环数组，当前循环的索引使用 **$index** 进行访问，当前的循环项使用 **$value** 进行访问。

```
{{each arr}}
    {{$index}} {{$value}}
{{/each}}
```

5.4 art-template标准语法

6. 标准语法 – 过滤器

过滤器的本质，就是一个 function 处理函数。

5.4 art-template标准语法

6. 标准语法 – 过滤器

```
{{value | filterName}}
```

过滤器语法类似**管道操作符**，它的上一个输出作为下一个输入。

定义过滤器的基本语法如下：

```
template.defaults.imports.filterName = function(value){/*return处理的结果*/}
```

5.4 art-template标准语法

6. 标准语法 – 过滤器

```
<div>注册时间：{{regTime | dateFormat}}</div>
```

定义一个格式化时间的过滤器 dateFormat 如下：

```
template.defaults.imports.dateFormat = function(date) {
    var y = date.getFullYear()
    var m = date.getMonth() + 1
    var d = date.getDate()

    return y + '-' + m + '-' + d // 注意，过滤器最后一定要 return 一个值
}
```

5. art-template模板引擎

5.6 案例 – 新闻列表

5.6 案例 – 新闻列表

1. 实现步骤

① 获取新闻数据

② 定义 template 模板

③ 编译模板

④ 定义时间过滤器

⑤ 定义补零函数

目录 Contents

6. 模板引擎的实现原理

6.1 正则与字符串操作

1. 基本语法

exec() 函数用于检索字符串中的正则表达式的匹配。

如果字符串中有匹配的值，则返回该匹配值，否则返回 null。

```
RegExpObject.exec(string)
```

示例代码如下：

```
var str = 'hello'
var pattern = /o/
// 输出的结果["o", index: 4, input: "hello", groups: undefined]
console.log(pattern.exec(str))
```

6.1 正则与字符串操作

2. 分组

正则表达式中 ( ) 包起来的内容表示一个分组，可以通过分组来**提取自己想要的内容**，示例代码如下：

```
var str = '<div>我是{{name}}</div>'
var pattern = /{{([a-zA-Z]+)}}/

var patternResult = pattern.exec(str)
console.log(patternResult)
// 得到 name 相关的分组信息
// ["{{name}}", "name", index: 7, input: "<div>我是{{name}}</div>", groups: undefined]
```

6. 模板引擎的实现原理

6.1 正则与字符串操作

3. 字符串的replace函数

replace() 函数用于在字符串中用一些字符**替换**另一些字符，语法格式如下：

```
var result = '123456'.replace('123', 'abc') // 得到的 result 的值为字符串 'abc456'
```

示例代码如下：

```
var str = '<div>我是{{name}}</div>'
var pattern = /{{([a-zA-Z]+)}}/

var patternResult = pattern.exec(str)
str = str.replace(patternResult[0], patternResult[1]) // replace 函数返回值为替换后的新字符串
// 输出的内容是：<div>我是name</div>
console.log(str)
```

6. 模板引擎的实现原理

6.1 正则与字符串操作

4. 多次replace

```
var str = '<div>{{name}}今年{{ age }}岁了</div>'
var pattern = /{{\s*([a-zA-Z]+)\s*}}/

var patternResult = pattern.exec(str)
str = str.replace(patternResult[0], patternResult[1])
console.log(str) // 输出 <div>name今年{{ age }}岁了</div>

patternResult = pattern.exec(str)
str = str.replace(patternResult[0], patternResult[1])
console.log(str) // 输出 <div>name今年age岁了</div>

patternResult = pattern.exec(str)
console.log(patternResult) // 输出 null
```

6. 模板引擎的实现原理

6.1 正则与字符串操作

5. 使用while循环replace

```
var str = '<div>{{name}}今年{{ age }}岁了</div>'
var pattern = /{{\s*([a-zA-Z]+)\s*}}/

var patternResult = null
while(patternResult = pattern.exec(str)) {
    str = str.replace(patternResult[0], patternResult[1])
}
console.log(str) // 输出 <div>name今年age岁了</div>
```

6. 模板引擎的实现原理

6.1 正则与字符串操作

6. replace替换为真值

```
var data = { name: '张三', age: 20 }
var str = '<div>{{name}}今年{{ age }}岁了</div>'
var pattern = /{{\s*([a-zA-Z]+)\s*}}/

var patternResult = null
while ((patternResult = pattern.exec(str))) {
    str = str.replace(patternResult[0], data[patternResult[1]])
}
console.log(str)
```

6. 模板引擎的实现原理

6.2 实现简易的模板引擎

1. 实现步骤

① 定义模板结构

② 预调用模板引擎

③ 封装 template 函数

④ 导入并使用自定义的模板引擎

6. 模板引擎的实现原理

6.2 实现简易的模板引擎

2. 定义模板结构

```
<!-- 定义模板结构 -->
<script type="text/html" id="tpl-user">
    <div>姓名：{{name}}</div>
    <div>年龄：{{ age }}</div>
    <div>性别：{{  gender}}</div>
    <div>住址：{{address  }}</div>
</script>
```

6. 模板引擎的实现原理

6.2 实现简易的模板引擎

3. 预调用模板引擎

```
<script>
    // 定义数据
    var data = { name: 'zs', age: 28, gender: '男', address: '北京顺义马坡' }

    // 调用模板函数
    var htmlStr = template('tpl-user', data)

    // 渲染HTML结构
    document.getElementById('user-box').innerHTML = htmlStr
</script>
```

6. 模板引擎的实现原理

6.2 实现简易的模板引擎

4. 封装template函数

```
function template(id, data) {
  var str = document.getElementById(id).innerHTML
  var pattern = /{{\s*([a-zA-Z]+)\s*}}/

  var pattResult = null

  while ((pattResult = pattern.exec(str))) {
    str = str.replace(pattResult[0], data[pattResult[1]])
  }

  return str
}
```

6. 模板引擎的实现原理

6.2 实现简易的模板引擎

5. 导入并使用自定义的模板引擎

```
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>自定义模板引擎</title>
    <!-- 导入自定义的模板引擎 -->
    <script src="./js/template.js"></script>
</head>
```

传智播客旗下高端IT教育品牌