组件公共模块设计 1.3.0

组件概要

组件化的目的

组件化的目的，在于快速的构建前端页面，一定程度的减轻前端人员开发页面的压力。在前端功能越来越复杂的今天，组件化是一条很好的应对挑战的道路。

技术栈

使用Jquery 和模板引擎来构建自定义标签，解析页面时，通过解析自定义标签来用模板生成html片段。组件间的通信使用订阅-发布机制。

总体分层

按照自定义标签解析的进度，将组件化的过程分为了三层。

1. 模板生成自定义标签

这是最高的一层，即配置页面时也使用模板配置，通过配置时的参数，通过模板引擎生成了以自定义标签构成的页面，这时，进入第二层。

1. 解析自定义标签

此层为第二层，第二层的主要工作，是将所有的自定义标签，解析为浏览器最基础的html标签。在解析的过程中，会读取自定义标签上的配置和事件绑定与事件订阅，并完成组件的事件绑定与订阅。

北京天源迪科

1. 最基础的

最后的一层，是第二层解析后的结果，页面元素全是浏览器最基础的html元素。

组件结构

将组件分为了公共模块和组件模块。其中，公共模块封装了整个组件公用的API。组件模块分为基础组件和业务组件

1. 公共模块

校验模块，序列化模块，提示语模块，公共ajax模块，对话框模块，超时处理模块，错误抛出模块，事件模块，模板引擎模块，组件解析模块，主题切换模块，生成组件唯一ID的模块，订阅-发布中心，宏命令模块。

公共模块概要

简介

公共模块中，组件解析模块，事件绑定模块和模板引擎模块是最为关键的三个公共模块，是整个组件架构的主体部分。此处将他们成为三将星。

组件解析模块

三将星的老大，是组件架构第二层中，解析自定义标签的入口。此模块提供了组件注册和组件解析的API，只有在此模块注册的组件且组件的初始化函数名为initTag的，才能被此模块解析。

事件绑定模块

三将星的老二，是组件架构第二层中，为组件添加绑定事件和订阅事件的模块，此模块中包含两个缓存列表和对外的三个API。提供了事件绑定与事件订阅，自定义事件绑定的回调函数和事件订阅的回调函数。

模板引擎模块

三将星的老三，为组件架构第二层中的组件解析提供模板功能，此模块提供了三个API，分别为生成模板的API，添加模板生成函数的API与用模板渲染数据的API。

北京天源迪科

订阅-发布中心

订阅发布中心提供了组件间通信的机制，向外暴露了两个API，用于发布事件的API和用于订阅事件的API，主要供内部调用。

校验模块

校验模块，利用一些规则来判断用户的输入是否正确，本身包含一个存放校验规则的缓存列表，并提供了可供外部调用的API。

序列化模块

表单序列化的模块，用于将表单内容序列化为特定的数据结构

ajax模块

公共的ajax模块，用于与后台服务器异步或同步的交互数据，支持动态的传入回调函数。

提示语模块

提示语模块，用来存放若查询为空等情况下的提示语，向外提供一个API。

Dialog模块

弹框模块，用于弹出操作提示的弹框。

错误抛出模块

用来抛出异常日志的模块。

超时处理模块

用于处理请求超时的模块

生成组件唯一ID模块

用于生成组件的唯一ID，此ID会在没有指定的情况下自动生成。

主题切换模块

主题切换的模块，用来切换组件的主题风格。

宏命令模块

用于执行一系列命令的模块，来处理一些公共的过程。

公共模块详细设计

组件解析模块 ParsingHelper

组件解析模块，用来解析自定义标签的模块，只有注册过的组件才能够被解析，通过对dom树的深度遍历来解析自定义标签，解析标签时，会调用组件中的initTag方法

北京天源迪科

| 组件解析模块API | |
| --- | --- |
| API | 功能 |
| ParsingHelper.registerComponent (componentName, fn) | 在解析模块中注册组件 |
| ParsingHelper.getTag ( ) | 获得已注册的组件标签，返回的是数组 |
| ParsingHelper.parsingTag( ) | 从头开始解析标签，不建议外部调用 |
| ParsingHelper.recursivlyParse($tag) | 从$tag开始解析tagNames标签 |

ParsingHelper.registerComponent (componentName, fn)

子自定义标签组建时，需要在解析模块中注册自定义的组件，否则，无法正常解析。componentName为自定义标签名，必须以cu开头，且全为小写。fn 为自定义标签的构造函数。

ParsingHelper.getTag ( )

获得所有在解析模块中注册的标签名，会以数组的形式返回。

ParsingHelper.parsingTag( )

从body开始解析标签，在初始化页面的时候调用，因此，不建议在初始化以外的地方使用此API。

ParsingHelper.recursivlyParse($tag)

以$tag标签为树的根部，深度遍历Dom树来解析标签。解析时，会将标签解析的第几次解析绑在$tag.data(“tagindex”)上，

事件绑定模块 BindEvent

事件绑定模块是核心模块之一，目的是为了实现组件事件绑定好组件间通信的灵活性，为了便于维护，将模块分为了三个小模块，分别为事件绑定的 BindEvent ，存放事件回调函数的EventHandler 和存放订阅事件后的回调函数 EventListenr。绑定事件，需要给组件配置绑定事件的属性，如：

<cubtn bindcl=“serialize,even1” listen='{"even1":"listen1","event2":"linten2"}' >按钮</cubtn>

| 事件配置 | | | |
| --- | --- | --- | --- |
| 属性名 | 属性值 | 属性类型 | 属性意义 |
| bind\* 如 bindcl | serialize,even1 | string | 绑定事件 |
| listen | {"even1":"listen1","event2":"linten2"} | object | 订阅事件 |

bind\*

组件事件绑定的属性，如bindcl=“serialize,even1” 就是为组件中， 含有属性hasbind的DOM绑定单击事件，且，单击事件的回调函数为serialize，且会将even1消息名称传递给serialize回调函数。当事件触发时，会通过BindEvent .bindEvent( )来调用回调函数列表中的函数。

| bind属性对应表 | |
| --- | --- |
| 名称 | 事件名 |
| bindcl | click |
| binddb | dbclick |
| bindfs | focus |
| bindfsi | focusin |
| bindfso | focusout |
| bindbl | blur |
| bindcg | change |
| bindku | keyup |
| bindkd | keydown |
| bindkp | keypress |

listen=‘{"even1":"listen1","event2":"linten2"}'

北京天源迪科

北京天源迪科

为组件订阅消息，一个组件可以订阅很多消息，订阅消息时，会调用BindEvent .setListener( ) 来订阅消息。如示例中，会监听even1消息，当发送了even1消息后，会执行listen1函数。

用bind\*绑定事件的回调函数中 this指向触发此事件的Dom元素

用listen 来订阅事件的回调函数中， this指向订阅此消息的组件。

| BindEvent API | |
| --- | --- |
| API | 功能 |
| BindEvent .bindEvent ($dom) | 为组件绑定事件， |
| BindEvent .setListener ($dom) | 为组件订阅事件 |
| BindEvent .parseOptions($tag, $dom) | 解析标签时，解析配置 |

BindEvent .bindEvent ($dom)

为组件绑定事件的API，接受一个参数，，$dom为需要绑定事件的Dom元素。绑定事件时，会读取Dom上的bindlist 配置来绑定事件。

BindEvent .setListener ($dom)

为组件订阅事件，接受一个参数， $dom为需要订阅事件的Dom元素。订阅事件时，会读取$dom上的listen配置来订阅事件。

BindEvent .parseOptions($tag, $dom)

解析标签时，将绑定事件的配置 与订阅事件的配置 转义为bindlist和listen，并放置在正确的位置，正确的位置为：1. hasbind=“true”的位置 2. 如未找到hasbind=“true”，则绑定在最外层

| EventHandler API | |
| --- | --- |
| API | 功能 |
| EventHandler .callHandler (eventName [, triggerName]) | 调用列表中的函数 |
| EventHandler .addHandler (eventName, function(triggerName){}) | 为列表中添加函数 |

EventHandler 模块的目的是为了方便的维护事件的回调函数，所以，此模块包含了两个API，即添加回调函数的API和执行回调函数的API。

EventHandler .callHandler (eventName [, triggerName])

执行回调函数的API，需要传递回调函数名，执行此API时，会按照eventName在回调函数列表中查找回调函数，找到后执行此函数。triggerName为需要发布的消息名称。此参数会传递给列表中的回调函数。

EventHandler .addHandler (eventName, function(triggerName){})

为列表中添加函数，需要传递两个参数，即回调函数名和回调函数，回调函数需要有triggerName形参。当回调函数被执行时，会传递消息名triggerName 给回调函数。

| EvEntListener API | |
| --- | --- |
| API | 功能 |
| EventListenr .listenEvent (eventName, handlerName) | 为组件订阅事件，此API中会调用Event.listen |
| EventListenr .addListenr (handlerName, fn) | 给列表添加函数 |

EventListenr 的目的，是为了方便的管理订阅事件列表，因此，提供了两个API，订阅事件的API和添加定会回调函数的API。

EventListenr .listenEvent (eventName, handlerName)

北京天源迪科

为组件订阅事件，需提供两个参数，调用此API时，会调用Event.listen()将eventName和回调函数托付给订阅发布中心。

EventListenr .addListenr (handlerName, fn)

为订阅事件列表添加名为handlerName的回调函数。

订阅-发布中心 Event

订阅-发布中心 是用来进行组件间的通信，包含两个API，发布消息的Event.trigger() 与订阅消息的Event.listen()

| API | 功能 |
| --- | --- |
| Event.trigger(triggerName,params) | 发布triggerName消息，并将params传递给订阅者的回调函数。 |
| Event.listen(eventName, fn) | 订阅名为eventName的消息 |

模板引擎模块 TemplateHelper

模板引擎模块，此模块默认使用mustache模板插件，并且可以很方便的更换模板插件。此模块中含有模板缓存列表和模板生成函数的列表，其中，模板生成函数列表中会固化一些模板生成函数，并支持动态的添加模板生成函数。

| 模板引擎模块API | |
| --- | --- |
| API | 功能 |
| TemplateHelper.render ( id, data [, EngineName]) | 标识为id的EngineName模板通过data数据来生成html，EngineName 为空时，默认为mustache模板 |
| TemplateHelper.parseOptions(type, id, options [, EngineName] ) | 通过options来生成标识为id的EngineName模板。EngineName 为空时，默认为mustache模板 |
| TemplateList.addTemFactory (type , fn [ ,EngineName ]) | 为EngineName 添加类型为type的模板构造函数，EngineName 为空时，默认为mustache模板 |

TemplateHelper.render ( id, data [,EngineName ])

通过id到模板缓存列表中查找模板，找到后通过data数据来渲染出html片段，EngineName未传时，默认为mustache模板

TemplateHelper.parseOptions(type, id, options [, EngineName])

用类型为type的模板生成函数，通过options配置，来生成标识为id的模板，生成后的模板会存入模板的缓存列表。EngineName未传时，默认为mustache模板

TemplateList. addTemFactory (type , fn [, EngineName])

添加类型为type的模板生成函数fn

校验模块 CheckHelper

校验模块，用来判断用户的输入是否正确，若输入错误，则弹出错误提示。校验模块一般用在表单中的输入项中，此框架目前只支持对input和textarea的校验。为了使校验正常进行，在拼装页面时需添加跟校验有关的配置项，如：

<cuinput regExp=”required-必输项,lowerLetter-小写字母” tipstyle="after-blue"></cuinput>

| 属性配置表 | | | | |
| --- | --- | --- | --- | --- |
| 属性名 | 属性值 | 属性类型 | 属性意义 | 开启校验时，是否为必输项 |
| regexp | required-必输项,lowerLetter-小写字母 | string | 校验规则名-校验提示，如有多个校验规则，用“,”隔开 | 是 |
| tipstyle | after-blue | string | 提示信息的样式 | 否 |

regExp ： 校验规则，当有多个校验规则时，需用“,”号隔开，配置好此属性后，校验模块在进行校验时，会依次查找校验函数列表，正则表达式列表。若还未找到，则会将属性值转化为正则表达式来进行校验。

tipStyle： 校验提示的样式，在启用校验时，是可有可无的属性，若此属性未配置，则样式采用默认的class名：invalid-div ，若配置了属性，则会将invalid-div替换为此配置项的值。

| 校验模块APi | |
| --- | --- |
| API | 功能 |
| CheckHelper.bindCheck ( $dom ) | 为组件绑定校验 |
| CheckHelper.checkForm ( $form ) | 进行表单校验，若未校验通过，则返回false |
| CheckHelper.setTheme ( themeName ) | 设置校验模块提示的主题样式 |
| CheckHelper.hideAllTip ( $form ) | 取消所有的校验提示，重置表单时使用 |

CheckHelper.bindCheck ( $dom )

为组件绑定校验的APi，接受一个参数，第一个参数为绑定校验的组件元素，

CheckHelper.checkForm ( $form )

对整个表单进行校验，且最后一个未校验通过的元素会获得焦点，当校验未通过时，返回false，此APi一般用于表单序列化之前的校验。

CheckHelper.setTheme ( themeName )

为检验提示语替换样式文件，themeName默认为.css文件，因此可以不用添加文件后缀，目前，只支持替换为同一样式主题下的文件。

CheckHelper.hideAllTip ( $form )

隐藏所有的错误提示，会搜索$form下所有的input和textarea来清除错误提示，一般用在重置表单，输入项上。

序列化模块 Serialize

表单序列化的模块，用于将form表单内以一定的格式序列化为一个对象，序列化时，会以组件的name属性来区分具体组件，如：

<cuinput data-options=‘{…,”name”:”input1”,…}’></cuinput>

北京天源迪科

| 属性配置 | | | | |
| --- | --- | --- | --- | --- |
| 属性名 | 属性值 | 值类型 | 属性意义 | 是否为必输项 |
| name | input1 | string | 表单序列化的标识 | 是 |

| 序列化模块API | |
| --- | --- |
| API | 功能 |
| Serialize.serializeObject ( $form) | 对表单进行序列化，序列化的格式为{name:value} |

Serialize.serializeObject ( $form)

对给定的表单进行序列化，且$form必须为form表单，序列化后的对象中，键为组件的name属性，值为组件的value值。在序列化时，没value值或没name属性的不会被序列化，且相同name属性的，value值会被拼成一个数组。因此，name值最好保证唯一。

ajax模块 CommonAjax

公共的ajax模块，进行ajax请求的模块，可设置成功或失败后的回调函数，还可设置一些其他的属性。

| ajax模块API | |
| --- | --- |
| API | 功能 |
| CommonAjax.ajax (data) | 发送ajax请求 |

| data属性配置 | | | |
| --- | --- | --- | --- |
| 属性名 | 属性值 | 属性类型 | 属性意义 |
| type | 默认为post | string | ajax请求类型，默认为post |
| url | 若不配置，则使用默认配置 | string | ajax请求路径，有默认地址 |
| params | {…name:value…} | object | ajax请求携带的参数 |
| cache | 默认为 false | boolean | 是否开启缓存 |
| dataType | 默认为 json | string | ajax数据类型 |
| successF | function(returnData){…} | Function | ajax成功后的回调函数 |
| errorF | function(XMLHttpRequest, textStatus, errorThrown){…} | Function | ajax失败后的回调函数 |
| respCode | 默认为 0000 | string | 后台是否处理成功的标识 |

主题切换模块 LoadStyle

此模块用来加载组件的样式，因此样式文件名需要与自定义标签名相同

| API | 功能 |
| --- | --- |
| LoadStyle.loadTheme(tagNames [,isUpdate]) | 加载传入的自定义标签的主题 |
| LoadStyle.setTheme (themeName) | 设置主题 |

LoadStyle.loadTheme(tagNames [,isUpdate])

此APi会从styles中与tagNames同名的样式文件，未传isUpdate时，只会加载一次样式文件，isUpdate为true时，会覆盖掉原先加载的样式文件。

LoadStyle.setTheme (themeName)

此API为 设置主题，主题名是styles文件夹中的目录名，主题文件夹下存放的就是具体的样式文件。

宏命令模块 Macro

Macro 的功能是执行一系列的操作

| API | 功能 |
| --- | --- |
| Macro .macroCommand($tag, $dom, options) | 执行一系列的标签解析操作 |

Macro .macroCommand($tag, $dom, options)

此API时为了执行一些公共的操作，里面调用的是其他公共模块的API，此模块需要3个参数，$tag自定义标签，$dom标签解析后的DOm元素 与组件配置。此API会执行绑定cid，绑定校验，事件绑定与发布，为组件外层设置内联样式。

生成组件唯一cid模块 CidFactory

生成cid的模块，目的是为了为组件生成页面级别的唯一标示cid，当页面配置cid时，以cid为主，当页面未配置cid时，根据自定义标签的解析顺序来生成cid。

| API | 功能 |
| --- | --- |
| CidFactory . makingCid (tagName, tagIndex, cid) | 为组件生成唯一标识并返回。 |

CidFactory . makingCid (tagName, tagIndex, cid)

此API用来为组件生成唯一标识且接收三个参数，tagName 为标签名 ，tagIndex为标签解析位置 cid为页面上配置的cid 。当cid存在，且cid不等于标签名，且cid唯一时，会返回cid，否则，会返回 tagName+tagIndex组成的字符串。

错误抛出模块 err

提示语模块

Dialog模块

超时处理模块