内容搜索开发

heima-leadnews-app

前端工程

VERSION - 1.0.0

|  |  |  |
| --- | --- | --- |
| 文档版本说明 | | |
| 文件状态  [ ] 草稿  [ √ ] 正式发布  [ ] 正在修改 | 文件标识： |  |
| 当前版本 |  |
| 作者 | miukoo |
| 完成日期 | 2019-06-19 |

目录

[目录 3](#_Toc19089425)

[1. 开发 5](#_Toc19089426)

[1.1 搜索输入组件开发 5](#_Toc19089428)

[1.1.1 文件创建 5](#_Toc19089429)

[1.1.2 实现VIEW 5](#_Toc19089430)

[1.1.3 实现Model 6](#_Toc19089431)

[1.1.4 实现VM 6](#_Toc19089432)

[1.1.5 实现Style 8](#_Toc19089433)

[1.2 历史项组件 9](#_Toc19089434)

[1.2.1 文件创建 9](#_Toc19089435)

[1.2.2 实现VIEW 9](#_Toc19089436)

[1.2.3 实现Model 9](#_Toc19089437)

[1.2.4 实现VM 10](#_Toc19089438)

[1.2.5 实现Style 11](#_Toc19089439)

[1.3 通用标题组件 12](#_Toc19089440)

[1.3.1 文件创建 12](#_Toc19089441)

[1.3.2 实现View 12](#_Toc19089442)

[1.3.3 Model定义 13](#_Toc19089443)

[1.3.4 实现Style 13](#_Toc19089444)

[1.4 图文列表项组件 15](#_Toc19089445)

[1.4.1 文件创建 15](#_Toc19089446)

[1.4.2 实现VIEW 15](#_Toc19089447)

[1.4.3 实现Model 15](#_Toc19089448)

[1.4.4 实现VM 16](#_Toc19089449)

[1.4.5 实现Style 17](#_Toc19089450)

[1.5 联想词列表项组件 19](#_Toc19089451)

[1.5.1 文件创建 19](#_Toc19089452)

[1.5.2 实现VIEW 19](#_Toc19089453)

[1.5.3 实现Model 20](#_Toc19089454)

[1.5.4 实现VM 20](#_Toc19089455)

[1.5.5 实现Style 21](#_Toc19089456)

[1.6 结果搜索输入组件 22](#_Toc19089457)

[1.6.1 创建文件 22](#_Toc19089458)

[1.6.2 实现VIEW 22](#_Toc19089459)

[1.6.3 实现Model 23](#_Toc19089460)

[1.6.4 实现VM 24](#_Toc19089461)

[1.6.5 实现Style 26](#_Toc19089462)

# 开发



## 搜索输入组件开发

搜索输入组件由于没直接可用的UI，在这里需要自行封装。其组件包含返回按钮、输入框；点击返回是返回上一页，点击‘搜索’或回车键跳到搜索结果页。

### 文件创建

* compoents/bars/文件夹下创建search\_top.vue文件，用于封装组件

### 实现VIEW

输入框重用之前封装的Search组件，并监听输入的提交、输入等事件，实现的代码如下：

<template>  
 <div class="bar\_bg">  
 <text class="icon" @click="back">&#xf104;</text>  
 <Search type="search"  
 autofocus="autofocus"  
 @onSubmit="onSubmit"  
 @onInput="onInput"  
 @onBlur="onBlur"  
 @onChange="onChange"  
 :icon="icon"  
 placeholder="请输入搜索关键字..."  
 rightWidth="0" />  
 </div>  
</template>

### 实现Model

Model有两个内部属性，用于控制组件的行为，如下：

data(){  
 return {  
 icon:'\uF002',  
 autofocus:true  
 }  
}

* Icon指定图标
* autofocus：指定输入框是否自动获得焦点

### 实现VM

在这里Vm主要完成的功能及代码如下：

* 在back方法中，直接调用之前封装的back方法，完成返回上一页功能
* 在onSubmit方法中，路由到搜索结果页面，并传递搜索的关键字
* onInput、onChange、onBlur实现事件的监听和广播给上层

<script>  
 import Search from '@/compoents/inputs/search';  
 export default {  
 name: "search\_top\_bar",  
 components: {Search},  
 data(){  
 return {  
 icon:'\uF002',  
 autofocus:true  
 }  
 },  
 methods:{  
 back : function(){  
 this.$router.back();  
 },  
 onSubmit : function(val){  
 this.$router.push({name:'search\_result',params:{'keyword':val}})  
 },  
 onInput:function(val){  
 this.$emit("onInput",val)  
 },  
 onChange:function(val){  
 this.$emit("onChange",val)  
 },  
 onBlur:function(val){  
 this.$emit("onBlur",val)  
 }  
 }  
 }  
</script>

### 实现Style

<style lang="less" scoped>  
 @import '../../styles/common';  
 .bar\_bg{  
 width: @screen-width;  
 flex-direction: row;  
 background-color: @bg-white;  
 border-style: solid;  
 height: @top-height;  
 padding-top: 5px;  
 padding-right: 35px;  
 align-items: center;  
 }  
 .icon{  
 color: #ababab;  
 font-family: fontawesome;  
 font-size: 40px;  
 padding:0px 30px 0px 20px;  
 }  
</style>

## 历史项组件

### 文件创建

* 在compoents\cells文件夹下新建seach\_0.vue文件，存放组件代码

### 实现VIEW

视图包含图标、文字、操作按钮三部分，其中操作按钮需要居右，后续点击文字可以跳转到搜索结果页，点击删除图标可以删除搜索记录，因此需要定义点击事件实现的代码如下：

<template>  
 <div class="cell-body">  
 <text class="icon">{{icon}}</text>  
 <text class="title" @click="onClickText">{{title}}</text>  
 <text class="icon" @click="onDeleteHistory">&#xf00d;</text>  
 </div>  
</template>

### 实现Model

在本案例中组件只实现了基本扩展：

* title：组件显示的文字
* icon：图标代码

props:{  
 title:{  
 type:String,  
 default:'显示字'  
 },  
 icon:{  
 type:String,  
 default:'\uf017'  
 }  
}

### 实现VM

VM中实现对点击事件的广播。

methods : {  
 onDeleteHistory : function(){  
 this.$emit("onDeleteHistory",this.id)  
 },  
 onClickText :function(){  
 this.$emit("onClickText",this.title)  
 }  
}

### 实现Style

<style lang="less" scoped>  
 @import '../../styles/common';  
 .cell-body{  
 flex-direction: row;  
 font-size: 36px;  
 align-items: center;  
 border-bottom-color: #ebebeb;  
 border-bottom-width: 1px;  
 border-bottom-style: solid;  
 padding: 17px 20px;  
 background-color: #ffffff;  
 }  
 .title{  
 flex: 1;  
 color: #222222;  
 }  
 .icon{  
 font-size: 40px;  
 color: #ebebeb;  
 width: 45px;  
 margin-right: 10px;  
 }  
</style>

## 通用标题组件

该组件与历史项组件类同，但布局存在一些类似边距等方面的细节，所以没有重用历史项组件，而是在这里新定义。

### 文件创建

* 创建src/compoents/titles/title.vue文件，用于实现组件

### 实现View

视图包含图标、文字、操作按钮三部分，其中操作按钮需要居右，实现的代码如下：

<template>  
 <div class="t-body">  
 <text class="icon" style="color: red">{{icon}}</text>  
 <text class="title">{{title}}</text>  
 <text class="icon">&#xf105;</text>  
 </div>  
</template>

### Model定义

在本案例中组件只实现了基本扩展，还可以扩展图标颜色控制等功能这里不实现：

* title：组件显示的文字
* icon：图标代码

props:{  
 title:{  
 type:String,  
 default:'显示字'  
 },  
 icon:{  
 type:String,  
 default:'\uf017'  
 }  
}

### 实现Style

<style lang="less" scoped>  
 @import '../../styles/common';  
 .t-body{  
 flex-direction: row;  
 font-size: 36px;  
 align-items: center;  
 border-bottom-color: #ebebeb;  
 border-bottom-width: 1px;  
 border-bottom-style: solid;  
 padding: 18px 20px;  
 background-color: #ffffff;  
 margin-top: 15px;  
 }  
 .title{  
 flex: 1;  
 color: #222222;  
 font-weight: bold;  
 }  
 .icon{  
 color: #ebebeb;  
 font-size: 40px;  
 margin-right: 10px;  
 width: 38px;  
 }  
</style>

## 图文列表项组件

图文列表项组件与以上定义的组件也类同，但在图标处理上存在细节差异，在本案例中新自定义组件来达到目的。

### 文件创建

* 创建src/compoents/cells/search\_1.vue文件，用于实现组件

### 实现VIEW

View主要包含图标和文字

<template>  
 <div class="cell-body">  
 <text class="title">{{title}}</text>  
 <text class="icon" :style="{'backgroundColor':getColor()}" v-if="tip!=''">{{tip}}</text>  
 </div>  
</template>

### 实现Model

Model定义了3个参数：

* title：参数用于传入文字
* tip：参数用于传入图标文字
* type：参数用于控制图标文字

props:{  
 title:{  
 type:String,  
 default:'显示字'  
 },  
 tip:{  
 type:String,  
 default:''  
 },  
 type:{  
 type:String,  
 default:''  
 }  
}

### 实现VM

Vm中主要生成文字图标的颜色，和文字图标的字，以及实现点击事件监听与广播，实现代码如下：

methods:{  
 getColor : function(){  
 if(this.tip=='热'||this.type=='0'){return '#ff1111';}  
 if(this.tip=='荐'||this.type=='1'){return '#8fdf19';}  
 if(this.tip=='新'||this.type=='2'){return '#db5f19';}  
 if(this.tip=='火'||this.type=='3'){return '#af1239';}  
 if(this.tip=='精'||this.type=='4'){return '#68da89';}  
 if(this.tip=='亮'||this.type=='5'){return '#8f78f5';}  
 return '#dbdbdb'  
 },  
 getText : function(){  
 if(this.tip=='热'||this.type=='0'){return '热';}  
 if(this.tip=='荐'||this.type=='1'){return '荐';}  
 if(this.tip=='新'||this.type=='2'){return '新';}  
 if(this.tip=='火'||this.type=='3'){return '火';}  
 if(this.tip=='精'||this.type=='4'){return '精';}  
 if(this.tip=='亮'||this.type=='5'){return '亮';}  
 return ''  
 },  
 onClick : function(){  
 this.$emit("onClick",this.title)  
 }  
}

### 实现Style

<style lang="less" scoped>  
 @import '../../styles/common';  
 .cell-body{  
 flex: 1;  
 flex-direction: row;  
 font-size: 36px;  
 align-items: center;  
 border-bottom-color: #ebebeb;  
 border-bottom-width: 1px;  
 border-bottom-style: solid;  
 padding: 17px 20px;  
 background-color: #ffffff;  
 }  
 .title{  
 flex: 1;  
 color: #222222;  
 }  
 .icon{  
 font-size: 24px;  
 color: #ffffff;  
 background-color: red;  
 border-radius: 10px;  
 padding: 2px 8px;  
 }  
</style>

## 联想词列表项组件

联想词组件与历史项组件布局类同，但存在大小和定位的差异，未快捷实现功能，这里新建文件实现。

### 文件创建

* 创建src/compoents/cells/search\_2.vue文件，用于实现组件

### 实现VIEW

联想词有一个输入标红的特性，这里我们可以把联想词拆成字符数组，每个字符对应一个text元素，如果元素字符与输入字符相同，则进行标红，因此VIEW显示的文字需要循环输出：

<template>  
 <div class="cell-body">  
 <text class="icon">{{icon}}</text>  
 <div @click="onClick" v-for="item in getText()">  
 <text class="text" :style="{color:getColor(item)}">{{item}}</text>  
 </div>  
 <text class="skip"> </text>  
 </div>  
</template>

### 实现Model

Model定义了3个参数：

* keyword：显示的关键字
* search：搜索输入的关键字
* icon：搜索图标

props:{  
 keyword:{  
 type:String,  
 default:""  
 },  
 search:{  
 type:String,  
 default:""  
 },icon:{  
 type:String,  
 default:'\uf002'  
 }  
}

### 实现VM

VM中各方法的用途：

* getText：把显示的关键字转出字符数组
* getColor：查找输入关键字中是否存在显示的字符，如果是则标红色
* onClick：广播点击事件

methods:{  
 getText : function(){  
 return this.keyword.split('')  
 },  
 getColor : function(chr){  
 return this.search.indexOf(chr)!=-1?'red':'#222222'  
 },  
 onClick : function(){  
 this.$emit("onClick",this.keyword)  
 }  
}

### 实现Style

<style lang="less" scoped>  
 @import '../../styles/common';  
 .cell-body{  
 flex-direction: row;  
 font-size: 36px;  
 align-items: center;  
 border-bottom-color: #ebebeb;  
 border-bottom-width: 1px;  
 border-bottom-style: solid;  
 padding: 17px 20px;  
 background-color: #ffffff;  
 }  
 .skip{  
 flex: 1;  
 }  
 .icon{  
 width: 60px;  
 font-size: 28px;  
 text-align: center;  
 color: #ababab;  
 }  
</style>

## 结果搜索输入组件

### 创建文件

* 创建src/pages/article/search\_result\_top.vue文件，用于实现功能

### 实现VIEW

View包括返回、输入框、取消按钮：

<template>  
 <div class="bar\_bg">  
 <text class="icon" @click="back">&#xf104;</text>  
 <Search  
 :icon="icon"  
 @onSubmit="onSubmit"  
 :value="value"  
 :placeholder="placeholder"  
 right-width="30"  
 left-width="30" />  
 <text class="icon" @click="home">&#xf00d;</text>  
 </div>  
</template>

### 实现Model

组件Model包括2个参数和一个属性：

* value：用于初始化输入框的值
* placeholder：用于初始化空值提示
* icon用于设置输入框的图标

props : {  
 value : {  
 type:String,  
 default:'12'  
 },  
 placeholder:{  
 type:String,  
 default:'请输入...'  
 }  
},  
data(){  
 return {  
 icon:'\uF002'  
 }  
}

### 实现VM

Vm主要实现相关按钮的功能：

* back方法实现返回上一页功能
* home方法实现取消搜索返回主页的功能
* onSubmit方法实现搜索事件的触发

<script>  
 import Search from '@/compoents/inputs/search';  
 export default {  
 name: "search\_top\_bar",  
 components: {Search},  
 props : {  
 value : {  
 type:String,  
 default:'12'  
 },  
 placeholder:{  
 type:String,  
 default:'请输入...'  
 }  
 },  
 data(){  
 return {  
 icon:'\uF002'  
 }  
 },  
 methods:{  
 back : function(){  
 this.$router.back()  
 },  
 home : function(){  
 this.$router.push('/home');  
 },  
 onSubmit : function(val){  
 this.$emit('onSubmit', val);  
 }  
 }  
 }  
</script>

### 实现Style

<style lang="less" scoped>  
 @import '../../styles/common';  
 .bar\_bg{  
 width: @screen-width;  
 flex-direction: row;  
 background-color: @bg-white;  
 border-style: solid;  
 height: @top-height;  
 padding: 10px 15px 5px 20px;  
 align-items: center;  
 background-color: #ffffff;  
 }  
 .icon{  
 color: #dbdbdb;  
 font-size: 42px;  
 }  
</style>