文章详情开发

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 2](#_Toc18938768)

[1 前端开发 3](#_Toc18938769)

[1.1 导航组件开发 3](#_Toc18938770)

[1.1.1 文件创建 3](#_Toc18938771)

[1.1.2 路由调整 3](#_Toc18938772)

[1.1.3 实现导航VIEW 6](#_Toc18938773)

[1.1.4 实现Model 7](#_Toc18938774)

[1.1.5 实现VM 7](#_Toc18938775)

[1.2 评论输入组件 10](#_Toc18938776)

[1.2.1 文件创建 10](#_Toc18938777)

[1.2.2 实现VIEW 10](#_Toc18938778)

[1.2.3 实现Model 11](#_Toc18938779)

[1.2.4 实现VM 14](#_Toc18938780)

[1.2.5 实现Style 18](#_Toc18938781)

[1.3 详情功能栏组件 19](#_Toc18938782)

[1.3.1 文件创建 19](#_Toc18938783)

[1.3.2 实现View 19](#_Toc18938784)

[1.3.3 Model定义 20](#_Toc18938785)

[1.3.4 VM开发 20](#_Toc18938786)

[1.3.5 实现Style 21](#_Toc18938787)

[1.4 图文按钮组件 22](#_Toc18938788)

[1.4.1 文件创建 22](#_Toc18938789)

[1.4.2 实现VIEW 22](#_Toc18938790)

[1.4.3 实现Model 23](#_Toc18938791)

[1.4.4 实现VM 24](#_Toc18938792)

[1.4.5 实现Style 26](#_Toc18938793)

# 前端开发

## 导航组件开发

导航一般存在后退的情况，为什么兼容各个终端，不能直接使用router的history API，需要自己实现，所以需要对其进行封装。

为提高开发效率，项目使用weex-ui快速开发，选用的导航样式参考：<https://alibaba.github.io/weex-ui/#/cn/packages/wxc-tab-bar/>

### 文件创建

* compoents/bars/文件夹下创建article\_top\_bar.vue文件，用于封装导航栏组件
* utils文件夹下创建navigator.js文件，用于记录路由器的历史记录

### 路由调整

#### 路由封装

路由使用数组进行封装，用于最近的10个请求URL，并提供back方法获取上一请求地址，其封装文件为utils/navigator.js，其中temp参数用于放置回退环，实现如下：

function Navigator(){  
 this.navigator = []  
 this.count = 0  
 this.max = 9  
 this.temp={}  
}  
Navigator.prototype= {  
 push : function(to,form,next) {  
 if (this.temp['path'] != to['path']) {  
 this.navigator[this.count] = form  
 this.count++;  
 if (this.count > this.max) this.count = 0;  
 }else{  
 this.temp={}  
 }  
 next()  
 },  
 back : function(){  
 this.count=this.count-1;  
 if(this.count<0)this.count=this.max;  
 this.temp=this.navigator[this.count]  
 return this.temp  
 }  
}  
  
export default new Navigator();

#### 路由拦截

在src/router.js中增加以下改动，以满足全局自动记录路由的功能：

* 注册之前封装的路由Navigator
* 注册一个back方法，用于全局便捷调用返回上一个页面
* 实现router的beforeResolve方法，拦截路由前置处理

实现代码如下：

/\*global Vue\*/  
import Router from 'vue-router'  
import routes from '@/routers/index'  
import navigator from '@/utils/navigator'  
  
Vue.use(Router)  
const router = new Router({  
 routes:routes  
});  
// 注册导航器属性  
router.$navigator = navigator  
// 注册返回方法  
router.back = function(){  
 let to = this.$navigator.back()  
 if(to){  
 this.push(to)  
 }  
}  
// 路由之前记录路由处理  
router.beforeResolve((to, from, next) => {  
 router.$navigator.push(to, from, next)  
})  
export {router}

### 实现导航VIEW

复制示例代码，需要替换左侧图标，以及绑定标题文字，其实现代码如下：

<template>  
 <wxc-minibar :title="getText"  
 :background-color="backgroundColor"  
 text-color="#FFFFFF"  
 :useDefaultReturn="useDefaultReturn"  
 @wxcMinibarLeftButtonClicked="minibarLeftButtonClick"  
 @wxcMinibarRightButtonClicked="minibarRightButtonClick">  
 <text class="icon" slot="left" style="text-align: left">&#xf104;</text>  
 </wxc-minibar>  
</template>

### 实现Model

导航的Model只有一个参数和一个属性，如下：

props:{  
 text:{  
 type:String,  
 default:'文章标题'  
 }  
},  
data:()=>{  
 return {  
 backgroundColor:'#009ff0'  
 }  
},

* title是一个参数，由父组件传入，指定中间位置显示的文字内容
* backgroundColor是一个属性，用于控制导航栏背景色

### 实现VM

在这里Vm主要完成的功能及代码如下：

* 在mounted后，初始化backgroundColor属性为全局变量值，以便统一控制APP主色调
* 提供getText方法，以保障标题的长度不超过12个字符，影响导航栏美观性
* minibarLeftButtonClick方法中实现返回按钮功能

<script>  
 import { WxcMinibar } from 'weex-ui';  
  
 export default {  
 name: "article\_top\_bar",  
 components: { WxcMinibar},  
 props:{  
 text:{  
 type:String,  
 default:'文章标题'  
 }  
 },  
 data:()=>{  
 return {  
 backgroundColor:'#009ff0',  
 useDefaultReturn:false  
 }  
 },  
 mounted(){  
 this.backgroundColor = this.$config.style.main\_bg;  
 },  
 computed:{  
 getText:function(){  
 let temp = this.text;  
 if(this.text.length>12){  
 temp = this.text.substring(0,11)+'...';  
 }  
 return temp;  
 }  
 },  
 methods: {  
 minibarLeftButtonClick () {  
 this.$router.back()  
 },  
 minibarRightButtonClick () {  
 }  
 }  
 }  
</script>

## 评论输入组件

### 文件创建

* 在compoents\inputs文件夹下新建search.vue文件，存放评论搜索组件代码

### 实现VIEW

由于weex-ui提供的搜索组件不能调整宽度，不够灵活，因此在这里自我定义评论输入组件，其视图代码如下，注意使用autocomplete属性控制浏览器输入记录，实现代码如下：

<template>  
 <div class="body" :style="getBodyStyle">  
 <text class="icon" :style="getIconStyle">{{icon}}</text>  
 <input @blur="onBlur"  
 @focus="onFocus"  
 @input="onInput"  
 @return="onSubmit"  
 @change="onChange"  
 :return-key-type="returnType"  
 :autofocus="autofocus"  
 autocomplete="off"  
 :type="type"  
 :placeholder="placeholder"  
 class="input"  
 :value="value"  
 :style="getInputStyle"  
 />  
 <text class="skin">&nbsp;</text>  
 </div>  
</template>

### 实现Model

为使组件更为通用，需定义以下参数：

* flex：定义组件的位置占比
* height：定义组件高度
* leftWidth：组件左边距
* rightWidth：组件右边距
* returnType：键盘返回键类型，默认search
* icon：组件显示的图标
* type：组件输入区域的输入类型，比如：text、password等
* color：文字颜色
* borderWidth：边框宽度
* backgroundColor：组件背景颜色
* placeholder：输入提示文字
* value：初始化的内容
* inpValue：用户输入的值

props:{  
 flex:{  
 type:Number,  
 default:1  
 },  
 height:{  
 type:Number,  
 default:70  
 },  
 autofocus:{  
 type:Boolean,  
 default:false  
 },  
 leftWidth:{  
 type:String,  
 default:"5"  
 },  
 icon:{  
 type:String,  
 default:'\uf044'  
 },  
 rightWidth:{  
 type:String,  
 default:"5"  
 },  
 type:{  
 type:String,  
 default:'text'  
 },  
 returnType:{  
 type:String,  
 default:'search'  
 },  
 color:{  
 type:String,  
 default:'#ababab'  
 },  
 borderWidth:{  
 type:Number,  
 default:1  
 },  
 backgroundColor:{  
 type:String,  
 default:'#ffffff'  
 },  
 placeholder:{  
 type:String,  
 default:'请输入...'  
 },  
 value:{  
 type:String,  
 default:''  
 }  
},  
data(){  
 return {  
 inpValue:''  
 }  
}

### 实现VM

组件VM的处理逻辑分为样式拼装和事件监听

需要计算组件各元素的样式，在computed方法中提供对应的实现：

* getInputStyle：计算input输入文本的高、字体大小、行高
* getBodyStyle：计算组件的左右边距，边框颜色和宽度、高度，还有背景颜色
* getIconStyle：计算图标的颜色、高度、宽度、大小、行高

该组件为通用组件，用于首页点击跳转到搜索页，以及搜素页输入搜索字后发起搜索功能，但由于兼容性的问题，input对点击事件兼容不是很好，因此需用onFocus事件替代：

* onBlur：失去焦点延迟10ms广播onBlur失去，并传参value
* onFocus：input获得焦点的时广播onFocus事件
* onSubmit：PC按回车，APP按键盘搜索时触发，广播onSubmit事件，并传参输入值
* onInput：记录用户输入的值
* onChange：内容改变事件

<script>  
 export default {  
 props:{},  
 data(){  
 return {  
 inpValue:''  
 }  
 },  
 computed:{  
 getInputStyle:function(){  
 return {  
 height: (this.height - 10) + "px",  
 lineHeight: (this.height - 10) + "px",  
 fontSize: (this.height / 70) \* 32 + "px"  
 }  
 },  
 getBodyStyle:function(){  
 return {  
 marginLeft:this.leftWidth+'px',  
 marginRight:this.rightWidth+'px',  
 borderColor:this.color,  
 borderWidth:this.borderWidth+"px",  
 backgroundColor: this.backgroundColor,  
 'flex':this.flex,  
 'height':this.height+'px'  
 }  
 },  
 getIconStyle:function(){  
 return {  
 color:this.color,  
 width:(this.height-10)+"px",  
 height:(this.height-10)+"px",  
 lineHeight:(this.height-10)+"px",  
 fontSize:(this.height/70)\*32+"px"  
 }  
 }  
 },  
 methods:{  
 onBlur() {  
 const self = this;  
 setTimeout(() => {  
 self.$emit('onBlur', { value: self.inpValue });  
 }, 10);  
 },  
 onFocus : function(){  
 this.$emit('onFocus', {});  
 },  
 onChange : function(){  
 this.$emit('onChange', this.inpValue);  
 },  
 onSubmit : function(){  
 if(this.inpValue!=''){  
 this.$emit('onSubmit', this.inpValue);  
 }  
 },  
 onInput : function(e){  
 this.inpValue = e.value;  
 this.$emit('onInput', this.inpValue);  
 }  
}

}  
</script>

### 实现Style

<style lang="less" scoped>  
 @import '../../styles/common';  
 .body{  
 flex-direction: row;  
 justify-content: space-between;  
 align-items: center;  
 border-width:1px;  
 border-color: @border-color;  
 border-style: solid;  
 border-radius: @top-height - 30;  
 }  
 .icon{  
 font-size: 28px;  
 text-align: center;  
 }  
 .input{  
 flex:1;  
 border: none;  
 }  
 .skin{  
 width: 20px;  
 }  
</style>

## 详情功能栏组件

### 文件创建

* 创建src/compoents/bars/article\_bottom\_bar.vue文件，用于实现组件

### 实现View

View主要引入评论输入组件，以及增加其它操作图标，实现代码如下：

<template>  
 <div class="bar\_bg">  
 <Search rightWidth=25 />  
 <text class="icon">&#xf075;</text>  
 <text @click="clickCollection" class="icon" :style="{color:collection?'red':'#bdbdbd'}">&#xf005;</text>  
 <text @click="clickForward" class="icon" :style="{color:forward?'red':'#bdbdbd'}">&#xf14d;</text>  
 </div>  
</template>

### Model定义

定义两个变量用于控制收藏、转发按钮的点亮：

* Collection：点亮收藏按钮
* Forward：点亮转发按钮

props:{  
 collection:{  
 type:Boolean,  
 default:false  
 },  
 forward:{  
 type:Boolean,  
 default:false  
 }  
},

### VM开发

VM主要用于实现组件的事件监听与广播：

* clickCollection：监听收藏按钮的点击，并广播事件
* clickForward：监听转发按钮的点击，并广播事件

<script>  
 import Search from '@/compoents/inputs/search';  
 export default {  
 name: "article\_bottom\_bar",  
 components: {Search},  
 props:{ },  
 methods:{  
 clickCollection : function(){  
 this.$emit("clickCollection",{})  
 },  
 clickForward : function(){  
 this.$emit("clickForward",{})  
 }  
 }  
 }  
</script>

### 实现Style

<style lang="less" scoped>  
 @import '../../styles/common';  
 .bar\_bg{  
 width: @screen-width;  
 flex-direction: row;  
 align-items:center;  
 border-width:1px;  
 border-color: @border-color;  
 background-color: @bg-white;  
 border-style: solid;  
 height: @top-height;  
 padding-top: 4px;  
 padding-left: 5px;  
 }  
 .icon{  
 color: @border-color;  
 font-size: 48px;  
 width: 80px;  
 }  
</style>

## 图文按钮组件

### 文件创建

* 创建src/compoents/buttons/button.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="botton" @click="click">  
 <text class="icon" v-if="icon!=''" :style="{color:active?'red':'#997283'}">{{icon}}</text>  
 <text class="text">{{getText()}}</text>  
 </div>  
</template>

### 实现Model

Model定义了4个参数和一个属性：

* text：参数用于传入按钮文字
* icon：参数用于传入按钮图标
* active：参数用于控制按钮是否激活
* activeText：参数用于显示在激活状态下的文字
* color：属性用于控制图标的颜色

props:{  
 text : {  
 type:String,  
 default : ''  
 },  
 icon : {  
 type:String,  
 default: '\uf164'  
 },  
 active : {  
 type:Boolean,  
 default: false  
 },  
 activeText : {  
 type:String,  
 default : ''  
 }  
},  
data(){  
 return {  
 color:''  
 }  
}

### 实现VM

Vm中主要实现按钮状态的文字显示和监听并广播点击事件：

<script>  
 export default {  
 name: "button",  
 props:{ },  
 data(){  
 return {  
 color:''  
 }  
 },  
 methods : {  
 click : function(){  
 this.$emit('onClick', {});  
 },  
 getText : function(){  
 if(this.active){  
 return this.activeText==''?this.text:this.activeText;  
 }else{  
 return this.text;  
 }  
 }  
 }  
 }  
</script>

### 实现Style

<style lang="less" scoped>  
 @import '../../styles/common';  
 .botton{  
 flex: 1;  
 border-width: 1px;  
 border-radius: 30px;  
 border-color: @border-color;  
 line-height: 60px;  
 flex-direction: row;  
 align-items: center;  
 font-size: 24px;  
 vertical-align: center;  
 margin: 0px 10px;  
 }  
 .icon{  
 font-size: 32px;  
 color: @border-color;  
 width: 40px;  
 margin-left: 10px;  
 }  
 .text{  
 color: @border-color;

font-size: 26px;  
 }  
</style>

// 基本样式-颜色  
@mian-color : #009ff0;  
@font-color : #ffffff;  
@border-color : #adadad;  
@bg-white : #ffffff;  
  
@top-height : 90px;  
@screen-width : 750px;  
  
  
.icon{  
 color: #ffffff;  
 font-family: fontawesome;  
}