19 评论输入框组件开发

更新时间：2019-08-30 10:07:46

> 谁和我一样用功，谁就会和我一样成功。
>
> ——莫扎特

朋友圈首页 UI - 评论输入框组件

移动 web 输入框相关的交互可以说是一个老生常谈的用户交互了，这也是组织移动 webAPP 在某些场景下无法代替 Native APP 的关键原因，由于键盘相关的交互，浏览器提供给 JavaScript 的 API 太少了，所以很难实现向 Native APP 的用户体验，但是就算如此，我们也要尽力而为，接下来，我们就开始攻坚。

本章节完整源代码地址，大家可以事先浏览一下：

Github-inputBar

Github-wecircle

Github-postItem

先看一下我们最终的效果:

iOS 效果：

Android 效果：

移动端键盘高度问题：

在移动 web 中使用 `<input>` 框时，与键盘相关的交互一直只一个比较头疼的问题，因为 javascript 并没有相关的接口可以操作或者获取到键盘相关的数据，唯一可以用的就是借助 `<input>` 输入框当获取到焦点时，会弹出一个键盘，并抛出一个 `onfocus` 事件，而且弹出键盘的表现在 iOS 和 android 系统中都有不同的表现：

在 iOS 中：

1. 假如输入框在屏幕的下半部分，当键盘弹出时，页面会被 “顶” 上去，这个是 webview 的默认行为，webview 会向上滚动至输入框可见的位置。
2. 而顶上去的距离则取决于你的输入框距离屏幕底部的距离，越靠下则被顶上去越多，越靠上则越少。
3. 假如输入框在屏幕上半部分，则不会被 “顶” 上去。

在 android 中：

则不会发生页面被顶上去的情况，键盘弹出后，会覆盖挡住一定的页面元素，页面的可视区域会变小。

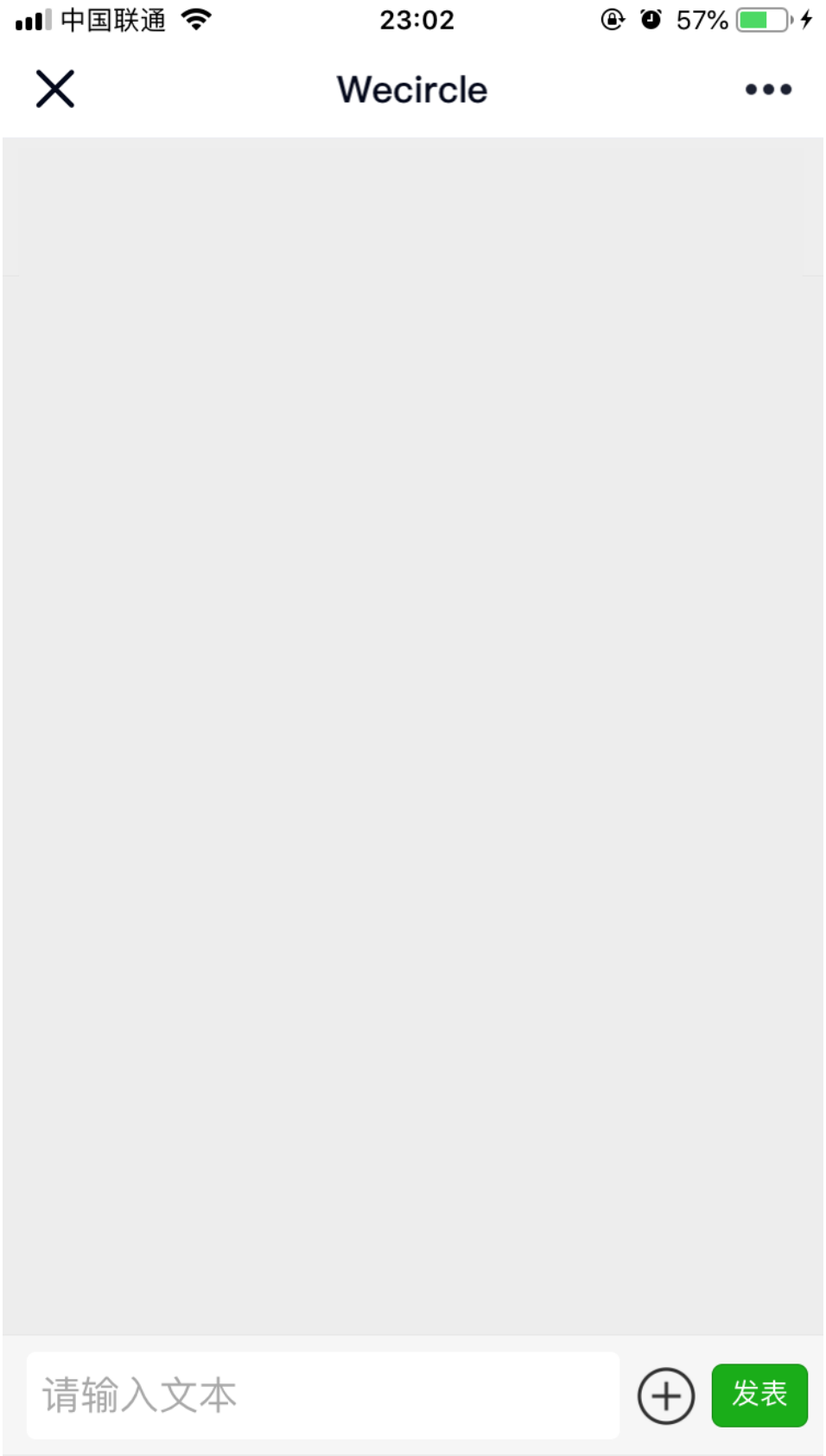
获取键盘高度：

虽然说页面会被顶上去，但是却可以借助于这种特性，判断可视区域的变化，我们可以利用 `window.innerHeight` 在键盘弹出前后的差值来获取到键盘的高度。正如下面这段代码：

```
created () {
  //在获取焦点前获取到`window.innerHeight`即`this.windowHeightOrgin`
  this.windowHeightOrgin = window.innerHeight
},
...
onfocus () {
  //input的onfocus则代表键盘弹出了
  setTimeout(() => {
    //在键盘弹出后利用延时，再次获取`window.innerHeight`，差值则表示键盘高度
    window.keyboardHeight = this.windowHeightOrgin - window.innerHeight

  }, 200)
},
```

我们在某个 vue 组件，`created` 也就是初始化时获取一下 `window.innerHeight`，然后在用户输入操作时，监听听到 `onfocus` 事件，然后获取 `window.innerHeight` 来计算插值来得到高度。注意这种情况获取高度要求输入框必须在屏幕的最底部，并且页面没有基于 body 的滚动条，如下图：



输入框 UI 开发：

在了解了上面的知识点之后，就可以进入开发了，在前端项目的 components 文件夹下新建 inputBar 文件夹，同时新建组件 `index.vue`：

```
<template>
 <div class="wrap scale-1px-top">
  <div class="input-content scale-1px">
   <div class="input-wrap"  @click="clickfocus">
    <input ref="input" class="weui-input input-inner" @focus="onfocus"  type="text" placeholder="请输入文本">
   </div>
   <div class="plus-btn" @click="showPlus" v-show="!option.noPlus"></div>
   <div class="create-btn weui-btn weui-btn_mini weui-btn_primary" @click="publish">发表</div>
  </div>
  <div class="opera-panel" v-show="!option.noPlus">
   <div class="opera-item">
    <div class="item-icon" @click="upload"></div>
    <p class="item-text">照片</p>
    <div style="display:none;" id="uploader">
     <input ref="uploader" id="uploaderInput" class="weui-uploader__input" type="file" accept="image/*" />
    </div>
   </div>
  </div>
 </div>
</template>
```

上面的代码中，我们主要实现了输入框的 UI：

1. `input.weui-input input-inner` 是输入框的 class，另外 `.plus-btn` 这个图片 icon 是选择图片上传功能的入口，我们会在后面的聊天页面里用到，这里也先把逻辑写上。
2. `div.opera-panel` 这个元素是图片上传功能的面板，同时预埋一个隐藏的 `div#uploader` 用作图片上传。
3. inputBar 组件并不设定具体在页面中的位置，它的位置由外层调用者来决定。

然后，我们开始给 inputBar 绑定各种事件，来监听获取键盘高度，代码如下：

```
/*
* 触发上传操作
*/
upload () {
  this.$refs.uploader.click()
},
/*
* 发表文字内容回调
*/
publish () {
  this.$emit('publish', {
    value: this.$refs.input.value,
    data: this.currentData
  })

  this.$refs.input.value = ''
},

/*
* 让输入框失去焦点，实际上这段代码就会收起键盘
*/
blurInput () {
  this.$refs.input.blur()
},
/*
* 让输入框获取焦点，实际上这段代码就会呼起键盘
*/
focusInput (currentData) {
  this.currentData = currentData
  this.$refs.input.focus()
},

onfocus () {
  //这段代码用来获取键盘高度，所以必须满足键盘在页面底部noplus代表没有底部的操作面板
  if (!this.option.noPlus) {

    setTimeout(() => {
      //键盘在页面底部时在获取
      if (!this.panelShow) {
        //键盘呼起前剪去键盘呼起后
        let kh = window.windowHeightOrgin - window.innerHeight;
        if (kh > 0) {

          //由于一些webview上下底部有导航栏，所以我们需要剪去这部分高度
          window.keyboardHeight = kh - (window.screen.height - window.windowHeightOrgin)
        }

      }
      //通知父组件隐藏掉图片操作面板
      this.$emit('hideBottomOnPanel')

    }, 200)
  }

},
```

这段代码里的方法比较多，总结一下：

1. `upload()` 方法是触发图片上传的回调，当点击上传按钮时，会将 weui 的 uploader 组件呼起。
2. `publish()` 方法是发表内容的回调，会触发一个事件，将内容通知外部组件。
3. `blurInput()` 方法是暴露给外层组件的方法，目的是让外层组件可以将 inputBar 关闭。
4. `focusInput()` 方法是暴露给外层组件的方法，目的是让外层组件可以将 inputBar 呼起，我们会在评论组件的评论按钮的点击回调里面去触发这个方法。

   这里涵盖了一个知识点：

5. 在移动端尤其是 iOS，如果我们想用 javascript 代码（非人工手动交互）的方式呼起一个键盘，就是直接在代码中调用 `<input>` 的 `focus()` 方法，原则上 webview 是不允许的，为什么不允许呢？因为为了避免代码无线循环会导致一直弹出键盘很不友好，但是通过对 webview 进行定制 `KeyboardDisplayRequiresUserAction=true` 可以解决这个问题，而达到效果，但是这需要修改 webview 的代码，所以目前至少在 Safari 上是不行的。
6. 但是系统虽然不能直接让我们通过代码方式呼起键盘，但是可以把呼起键盘的 `focus()` 方法代码调用放在一个 `click` 的回调里面，或者是其他通过用户真实手指交互行为的方法回调里面，中间不能有任何异步操作例如 `setTimeout`，就可以呼起键盘。正如我们代码里将用户点击评论按钮的回调利用起来，在这里呼起键盘。

计算输入框和键盘位置

处理完 `input` 之后，我们如果想要评论输入框出现在我们点击评论的那个位置上，就像微信一样，我们还需要这样处理：

首先回到我们之前开发的单条内容组件中 (前端项目的 components 目录的 postItem 目录下的 index.vue)，在评论按钮点击的时候，添加逻辑，去呼起键盘，代码如下：

```
/*
* 点击评论框回调
*/
addComment (evt) {

  //将事件发生时实际位置标记
  this.data.pageY = evt.pageY

  this.$bus.$emit('showInput', true, this.data)

  this.showOpera = false
},
```

这里我们采用 EventBus (EventBus 在本章节结尾来讲解) 的 `$emit()` 方法抛出一个事件 showInput，然后在回到朋友圈首页的组件 (前端项目的 views 目录下的 wecircle 目录的 index.vue) 中，去接受这个事件，代码如下：

```
//处理评论输入框的显示和隐藏
this.$bus.$on('showInput', (flag, currentData) => {
  this.showInput = flag
  if (flag) {

    this.windowHeightOrgin = window.innerHeight;
    this.$refs.inputBar.focusInput(currentData)

    //在ios里，页面会被顶上来，要单独处理
    if (os.isIOS){

      //设置input在手指点击的那个位置出现，10表示稍向下移动一些
      this.$refs.inputBarWrap.style.top = (currentData.pageY-10) + 'px'


      setTimeout(() => {

        //输入框的位置剪去键盘的高度剪去位置微调系数45 ： 5
        let y = currentData.pageY - window.keyboardHeight-(os.isIpP ? 45 : 5)

        //通过调用window.scroll在将页面顶回去一定距离
        window.scroll(0, y)

      }, 210)
    }

  } else {
    try {
      this.$refs.inputBar.blurInput()
    } catch (e) {}
  }
})
```

然后，还需要将我们之前写的 inputBar 组件，给引入进来，并添加在 template 中，代码如下：

```
...
<div :style="{zIndex:showInput?'999':'-1',opacity:showInput?'1':'0'}" ref="inputBarWrap" class="input-wrap ios" v-if="iosInput">
  <inputBar ref="inputBar" :option="inputBarOption" @publish="publish"/>
</div>
...
```

1. 在 android 上，由于页面不会被顶上来，所以我们将 `input` 放在最底部，紧贴着键盘即可，我们需要单独对 iOS 处理，就是在键盘呼起时，让页面还会到最初的位置。
2. 通过调用 `window.scroll()` 可以让页面顶回去，但是具体顶多少距离需要通过输入框的位置以及键盘的高度通过计算得出，公式：（输入框的位置 - 键盘的高度 - 微调系数（其中 5 是表示对位置微调））。
3. iOS 中，我们让 `input` 采取绝对定位，通过控制 top 值来改变输入框位置 `this.$refs.inputBarWrap.style.top = (currentData.pageY-10) + 'px'`，在 android 中，无需控制，让输入框紧贴底部即可。

iOS 的逻辑复杂一些，其中的逻辑可以看下图的解释:

点击时记录位置 (pageY),
用来定位输入框

iOS 和 android 中，初始化状态输入框都是隐藏的，但是 iOS 的隐藏是通过绝对定位和 z-index 来让其不可见，android 的隐藏是通过绝对定位到屏幕外面来让其不可见，下面是不同的输入框样式：

```
.input-wrap.ios {
  position: absolute;
  left: 0;
  right: 0;
  top: 80px;
  z-index: -1;
}
.input-wrap.android {
  position: fixed;
  left: 0;
  bottom: -60px;
  right: 0;
  z-index: 999;
}
```

之所以没有使用 `display:none` 是因为，如果尝试调用 input 的 `focus()` 方法让其呼起键盘，这个输入框不能隐藏(display:none)。

兼容没有获取键盘高度的情况

由于我们必须在键盘呼起一次才能得到键盘的高度，所以在用户还没有这个操作前我们还得不到键盘高度，所以我们给常见的 iOS 机型设置一些默认的键盘高度默认值：
在前端项目的 utils 文件夹下创建 `os.js`：

```
const u = navigator.userAgent;
const isAndroid = u.indexOf('Android') > -1 || u.indexOf('Linux') > -1;
const isIOS = !!u.match(/\(i[^;]+;( U;)? CPU.+Mac OS X/);

export default {
  isIOS: isIOS,
  isAndroid: isAndroid,
  getKeyBoardHeightDefault:function(){
    if (isIOS) {
      let screen = window.screen;
      //iphone x
      if(screen.height == 812 && screen.width == 375){
          return 377
      }else if(screen.height == 736 && screen.width == 414){//iphone plus
          return 315
      }else if(screen.height == 667 && screen.width == 375){//iphone 678
          return 304
      }else if(screen.height == 568 && screen.width == 320){//iphone 5 se
          return 297
      }else{
          return 304
      }
    } else {
      return 304
    }

  }
}
```

我们将这个逻辑写在工具方法里面，方便我们调用。
注意：这里提醒一些各位同学，在移动端 web 开发中，有很多场景需要对 iOS 和 android 做单独处理，抽离出一个公用方法是一个不错的选择。

跨组件通信方案 EventBus

在我们 input 组件和外界通信时，利用了 EventBus 通信方案，这个是在不使用 vuex 情况下的另外一种通过跨级组件通信方案，在 Vue 中可以使用 EventBus 来作为沟通桥梁的概念，就像是所有组件共用相同的事件中心，可以向该中心注册发送事件或接收事件，所以组件都可以上下平行地通知其他组件。
我们在 main.js 中加入 vue 原生方法：

```
Vue.prototype.$bus = new Vue()
```

之后我们就可以在后面的 vue 组件中直接使用：

```
this.$bus.$emit('AEvent',{ ... pass some params ...}); //派发事件

this.$bus.$on('AEvent',(params) => { //监听事件
    // ...
})

this.$bus.$off('AEvent') //移除事件
```

实际上是利用了一个空的 vue 组件，并且调用了 $emit 和 $on 方法来事件通信，它的工作原理是发布 / 订阅方法，通常称为 Pub/Sub，也就是发布订阅的模式。

小结

本章节主要讲解了移动 web 端键盘和输入框的定位问题，而键盘和输入框是公认的坑比较多的地方，所以代码里可能会看到很多 setTimeout 的地方，这里都是由于键盘呼起和收起并没有提供 javascript 的接口回调，所以用延时来代替，正因为如此，如果各位能够充分理解这里的原理，那么恭喜你进入了中高级的移动 web 工程师。
相关技术点：

1. 利用 <input> 的 focus 事件前后的 window.innerHeight 可以得到手机的键盘高度，当然这个是一个 hack 方法，并没有标准的 API 可以得到这个参数。
2. 在移动端尤其是 iOS，如果我们想用 javascript 代码（非人工手动交互）的方式呼起一个键盘，原则上 webview 时不允许的，但是可以把呼起键盘的 focus() 代码调用放在一个 click 的回调里面，或者是其他通过用户手指交互行为的方法回调里面，中间不能有任何异步操作例如 setTimeout 就可以呼起键盘。
3. 跨组件通信方案 EventBus，实际上是利用了一个空的 vue 组件，并且调用了 $emit 和 $on 方法来事件通信，它的工作原理是发布 / 订阅方法，通常称为 Pub/Sub，也就是发布订阅的模式。

本章节完整源代码地址：
Github-inputBar
Github-wecircle
Github-postItem

}