16 单条内容组件开发

更新时间：2019-08-28 15:34:42

> 路漫漫其修远兮，吾将上下而求索。
>
> ——屈原

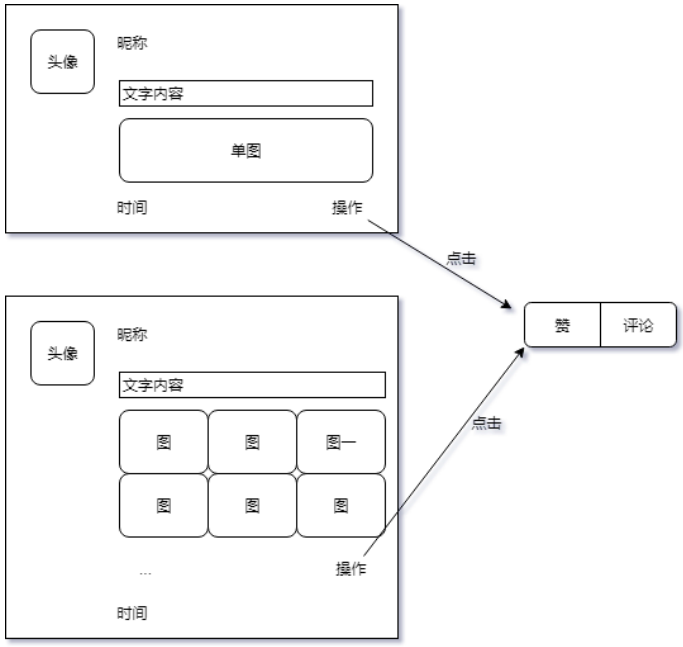
朋友圈单条内容组件指的是每一条的朋友圈内容，我们把每一条朋友圈封装成一个公共组件，这部分内容的逻辑是相对来说比较多的，也是整个朋友圈首页比较重要的一部分，下面就进入开发吧。

本章节完整源代码地址，大家可以预先浏览一下：

Github

组件逻辑图

首先，我们需要看一下这个组件都由哪些元素组成，并且有哪些用户交互，可以看下图：



用过朋友圈的人应该都不陌生，基本的交互和微信朋友圈一样，只是省去了一部分，下面就开始写组件代码。

开发 **postItem** 组件

在前端项目的 components 文件夹下新建 postItem 文件夹，同时新建 index.vue：
下面这段代码主要实现的是单条内容组件的 UI 中左侧头像的模块：

```
<div :class="(data.user.params &&  data.user.params.vip == 1 ) ? 'avatar-wrap vip' : 'avatar-wrap'">
  <img class="avatar" @click="goPersonPage(data.user._id)" :src="data.avatar">
</div>
```

这里我们新增了一个 vip 的逻辑，如果后端返回了 vip 的标识，就会在头像的左下角显示出标识。对应的 css 代码如下：

```
.avatar-wrap {
  width: 44px;
  height: 44px;
  border-radius: 4px;
  background-color: #e9e9e9;
  position: relative;
}
.avatar-wrap.vip {
  border: 2px solid #fcd20c;
}
.avatar {
  width: 100%;
  height: 100%;
  display: block;
  border-radius: 4px;
}
.avatar-wrap.vip::after {
  content: " ";
  display: block;
  width: 16px;
  height: 16px;
  position: absolute;
  background-image: url("./img/vip1.png");
  background-size: cover;
  position: absolute;
  right: -5px;
  bottom: -3px;
}
```

在这里使用伪类 after 来作为 vip 的元素，需要设置成绝对定位 `position: absolute;`。

接着，我们需要给头像绑定 click 事件，`goPersonPage()` 方法表示点击的回调，代码逻辑如下：

```
goPersonPage (_id) {
  var id = _id
  // 如果是当前用户点击当前的本人的头像 暂不跳转
  if (id === this.$store.state.currentUser._id) {
  } else {
    this.$router.push({
      path: 'personpage',

      query: {
        id: id
      }
    })
  }
},
```

我们给使用一个 `.item` 来包裹整个单条内容模块，设置 `displa:flex;`，这样左边就是头像部分，右边就是内容部分。

```
.item {
  display: flex;
  padding-left: 10px;
  padding-top: 15px;
  position: relative;
  padding-bottom: 17px;
}
```

然后就是右侧的内容部分，首先是从上到下依次是　nickname，文字模块和图片模块然后是评论点赞模块；其中图片模块逻辑分为单图逻辑和多图逻辑。右侧模块的 html 代码如下：

```
<div class="content-info">
  <p class="nickname one-line">{{data.nickname}}</p>
  <div class="post-content three-line">{{data.content}}</div>
  <div class="img-content" v-if="data.piclist.length > 1">
    <div
      class="img-wrap"
      :style="imgWrapStyle(item)"
      v-for="(item,index) in data.piclist"
      :key="index"
      @click="showImage(index)"
    ></div>
  </div>
  <div class="img-content-one" v-if="data.piclist.length === 1">
    <img :style="imgOneStyle(data.piclist[0])" class="img-wrap-one" :src="getImageUrl(data.piclist[0].url)" @click="showImage()">
  </div>
  <div class="time">{{data.time}}</div>
  <div class="opera-box" @click="showPanel($event)">
    <div class="box-panel-wrap">
      <transition name="slide">
        <div class="box-panel" v-show="showOpera">
          <div class="like-box" @click="dealWithLike">
            <div class="like-icon"></div>
            <div class="like-text" v-show="!data.isLike">赞</div>
            <div class="like-text" v-show="data.isLike">取消</div>
          </div>
          <div class="divider"></div>
          <div class="comment-box" @click="addComment($event)">
            <div class="comment-icon"></div>
            <div class="comment-text">评论</div>
          </div>
        </div>
      </transition>
    </div>
  </div>
  </div>
</div>
```

文字正文 3 行显示

上面的代码中，我们给 `.post-content` 设置了 `three-line` 这个 class，这里所要实现的逻辑就是当文字高度超出 3 行时，要已省略号结尾展示，这里我们需要利用 `-webkit-line-clamp: 3;` 这个属性。

在前端项目的 src 文件夹下的 assets 文件夹新建 `common.css` 用来存放一些多个页面通用的一些样式，在这里写上 `.three-line` 对应的样式，代码如下：

```
.three-line {
    overflow: hidden;
    display: -webkit-box;
    -webkit-line-clamp: 3;
    text-overflow: ellipsis;
    -webkit-box-orient: vertical;
}
```

1. `text-overflow: ellipsis;` 表示文字的省略号结尾。
2. `-webkit-line-clamp: 3;` 表示超出 3 行就开始截取，后面的数字代表行数。
3. `display: -webkit-box;` 表示需要 flex 布局，这种样式才能生效。

单图和多图的样式

`.img-wrap` 是图片的外层容器，用来设置内部的图片样式，内部的图片将分为单图和多图两种样式，通过 `data.piclist.length === 1` 和 `data.piclist.length > 1` 可以判断出当前的数据是展示单图还是多图。

`img-content-one` 和 `img-content` 分别对应两种朋友圈类型的图片展示，在多图中，我们利用 background-image 来将一个图片居中：

```
.img-wrap {
  width: 80px;
  height: 80px;
  margin-right: 4px;
  margin-bottom: 4px;
  background-size: cover;
  background-position: center center;
  background-repeat: no-repeat;
  background-color: #e9e9e9;
}
```

上面的代码利用了我们之前讲解过的 `background-size: cover;` 这个属性，在单图中，我们指定一个最大宽度或者最大高度一条边来进行展示，`<img>` 标签的特性就是给定 1 条边，另外一条会自适应大小：

```
  /*
  * 单图样式
  */
  imgOneStyle () {
    return item => {
      let height = null;
      let width = null;
      //如果图片是长图则给定最大的长度
      if (item.size.height > item.size.width) {
        height = Math.min(200,item.size.height);
        //根据比例设置宽度
        width = height*item.size.width/item.size.height;

      } else {//如果图片是宽图则给定固定的宽度
        width = Math.min(200,item.size.width);
        //根据比例设置高度
        height = width*item.size.height/item.size.width;

      }
      //转换成vw单位
      return {
        height:this.pxtovw(height),
        width:this.pxtovw(width)
      }
    }
  }
```

我们在 `computed` 使用 `return` 一个 `function` 可以达到传参的技巧：`<img :style="imgOneStyle(data.piclist[0])" ... />`。

使用阿里云不同尺寸的图片

在开发图片这个模块时，我们从后端收到的图片 url 数据是原始的 url ，就是说是原图的 url，而一般情况下在列表页面会优先使用缩略图，来加快图片显示的速度，我们新建一个 filter ，来处理返回的原始 url ，采用不同尺寸的 url ：

```
/*
* 格式化url
*/
 filters: {

  /*
  * 格式化url
  */
  urlFilter(val){
   let _url = val+'?x-oss-process=image/resize,l_400';//500,400,300,200,100...文档地址：https://help.aliyun.com/document_detail/44688.html
   return _url
  }
 },
```

在 url 后面加 ?x-oss-process=image/resize,l_400';//500,400,300,200,100... 表示使用不同的尺寸，这个是阿里云图片服务提供的功能，简单方便。

1px 的边框样式

根据之前的 UI 图，每个单条内容组件有一个 1px 的边框，在移动端使用 1px 的边框可不是简单的设置 border:1px 就可以的，我们需要写一个 1px 的边框样式，原理就是将 1px 高度的 div 利用 transform 的 scaleY 来进行缩放。

在前端项目的 src 文件夹下的 assets 文件夹新建 common.css 用来存放一些多个页面通用的一些样式，由于 .scale-1px 是一个通用的样式，我们将这部分代码写在 common.css 中。

```
.scale-1px::after {
    content: '';
    position: absolute;
    height: 1px;
    width: 100%;
    bottom: 0;
    left: 0;
    -webkit-transform: scaleY(0.5);
    transform: scaleY(0.5);
    -webkit-transform-origin: 0 0;
    transform-origin: 0 0;
    background-color: #e5e5e5;
}
```

点赞和评论弹出框样式

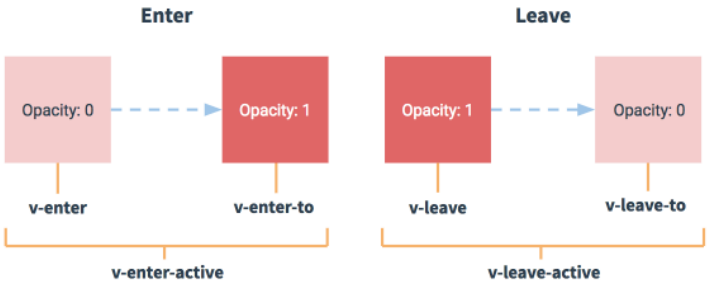
在单条内容组件中，点赞和评论也是一个重要的模块，下面这段代码用来编写对应的 UI，我们把这段代码添加在上面我们写右侧模块的 html 中，这里用到了 vue 的 transition：

```
<div class="opera-box" @click="showPanel($event)">
 <transition name="slide">
  <div class="box-panel" v-show="showOpera">
   <div class="like-box">
    <div class="like-icon"></div>
    <div class="like-text" @click="addLike" v-show="!data.isLike">赞</div>
    <div class="like-text" @click="removeLike" v-show="data.isLike">取消</div>
   </div>
   <div class="divider"></div>
   <div class="comment-box" @click="addComment($event)">
    <div class="comment-icon"></div>
    <div class="comment-text">评论</div>
   </div>
  </div>
 </transition>
</div>
```

需要注意的是，开始时，这个模块的 UI 是隐藏的，只有在点击 `.opera-box` 来触发 `showPanel()` 回调来显示点赞评论面板。

`transition` 是 vue 里使用动画的内置组件，我们只需要给 `transition` 设置不同的 class 就可以配置动画不同阶段的参数:

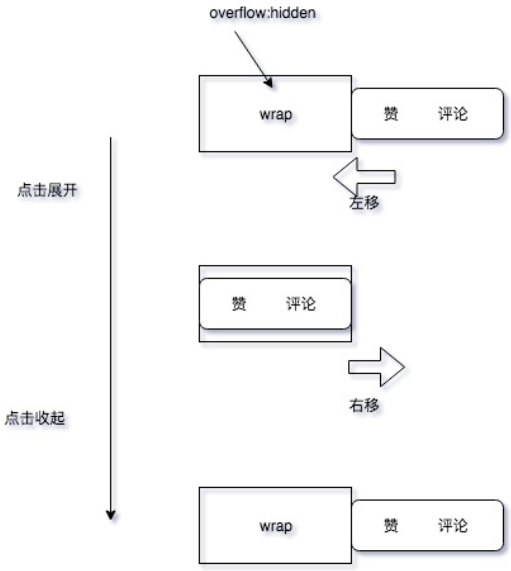
- **v-enter**：定义进入过渡的开始状态。在元素被插入之前生效，在元素被插入之后的下一帧移除。
- **v-enter-active**：定义进入过渡生效时的状态。在整个进入过渡的阶段中应用，在元素被插入之前生效，在过渡 / 动画完成之后移除。这个类可以被用来定义进入过渡的过程时间，延迟和曲线函数。
- **v-enter-to**: 2.1.8 版及以上 定义进入过渡的结束状态。在元素被插入之后下一帧生效 (与此同时 v-enter 被移除)，在过渡 / 动画完成之后移除。
- **v-leave**: 定义离开过渡的开始状态。在离开过渡被触发时立刻生效，下一帧被移除。
- **v-leave-active**：定义离开过渡生效时的状态。在整个离开过渡的阶段中应用，在离开过渡被触发时立刻生效，在过渡 / 动画完成之后移除。这个类可以被用来定义离开过渡的过程时间，延迟和曲线函数。
- **v-leave-to**: 2.1.8 版及以上 定义离开过渡的结束状态。在离开过渡被触发之后下一帧生效 (与此同时 v-leave 被删除)，在过渡 / 动画完成之后移除。



2.1.8 版指的是 vue 的版本，我们将 v 替换成 `transition` 的 name=`slide` 即可：

```
.slide-enter-active {
  width: 180px;
  transition: width 300ms;
}
.slide-leave-active {
  width: 0;
  transition: width 300ms;
}

.slide-enter,
.slide-leave-to {
  width: 0;
}
```



通过改变 transform:translate3d 属性，来实现点赞评论面板的位移动画。

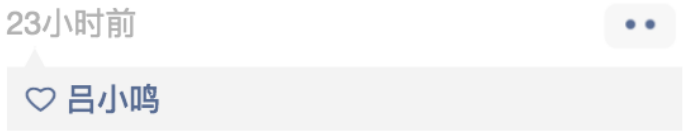
点赞和评论内容显示

有了点赞和评论的弹出框，就要有每条朋友圈的内容展示部分了，我们在 .opera-box 这个 class 的后面添加上这部分代码：

```
<div class="comment-list" v-show="data.likes.length || data.comments.length">
  <div class="like-content" v-show="data.likes.length">
    <div class="like-heart-icon"></div>
    <span
      v-for="(item,index) in data.likes"
      class="like-nickname"
      @click="goPersonPage(item.user._id,index)"
      :key="item.user._id"
    >
      {{item.user.nickname | likeFilter(index,data.likes.length)}}
    </span>
  </div>
  <div
    class="comment-item scale-1px-top"
    v-for="(item,index) in data.comments"
    v-show="data.comments.length"
    :key="index"
  >
    <div
      @click="goPersonPage(item.user._id)"
      class="comment-nickname one-line"
    >{{item.user.nickname}}:</div>
    <div>{{item.content}}</div>
  </div>
</div>
```

上面代码的知识点在于：

使用伪类模拟一个三角形，来形成气泡的样式，如下图：



这部分原理我们会放在后面聊天信息内容 UI 开发中讲解，比较典型。

对于点赞部分的 UI，每一个点过赞的名称要从左往右排列，并且每个名称都可以点击调整，要能换行，这里我们采用 `span` 来包裹每一个名称，同时父元素采用 `word-break: break-all` 来强制换行，对应的 css 代码如下：

```
.like-nickname {
  color: rgb(87, 107, 149);
  font-weight: 500;
}
.like-content {

  padding-top: 3px;
  padding-bottom: 3px;
  padding-left: 8px;
  padding-right: 8px;
  text-align: left;
  word-break: break-all;
}
.like-heart-icon {
  width: 14px;
  height: 14px;
  background-image: url("./img/sq-like.png");
  background-size: cover;
  align-self: center;
  margin-right: 4px;
  margin-top: 4px;
  float: left;
}
```

使用 vuex 更新点赞和评论后的 UI

我们开发完成了点赞和评论对应的 UI 之后，就要有对应的用户点击逻辑，下面我们新定义两个 addLike 方法和 removeLike 方法，同时给 `.like-box` 绑定 `dealWithLike` 方法，然后就主要是调用 `service.post()` 方法来调用后端的接口：

```
dealWithLike () {
  if (this.data.isLike) {
    this.removeLike()
  } else {
    this.addLike()
  }
},
/*
* 点赞
*/
async addLike () {
  // 调用API
  let resp = await service.post('likecomment/addlike', {
    postId: this.data.id
  })
  if (resp.code === 0) {
    //通知store去更新ui
    this.$store.dispatch('addLike', {
      pid: this.data.id,
      user: this.$store.state.currentUser
    })
    this.showOpera = false
  }
},
/*
* 取消点赞
*/
async removeLike () {
  // 调用API
  let resp = await service.post('likecomment/removelike', {
    postId: this.data.id
  })

  if (resp.code === 0) {
    //通知store去更新ui
    this.$store.dispatch('removeLike', {
      pid: this.data.id,
      user: this.$store.state.currentUser
    })
  }
},
```

我们在完成数据调用之后，需要对当前也的相关 UI 进行更新，例如点完赞要让自己的名字出现在当前的那个朋友圈下面，评论完之后要让自己的评论数据出现，这里是无刷新更新，为什么是无刷新？

1. 每次点赞评论完成之后如果在重新发一次请求，浪费资源。
2. 在用户交互上，刷新页面的体验并不好，而且需要定位到之前操作的位置。

所以需要借助 vuex 来实现跨组件的数据通信来达到无刷新更新 UI ，如下逻辑。

在 store.js 里新增 action 和 mutations:

```
actions: {
  ...
  addLike (context, obj) {
    context.commit('addLike', obj)
  },
  ...
},
mutations: {
  ...
  addLike (state, obj) {
    var list = state.wecircleDataList
    //在wecircleDataList里找到当前点赞的那个post，然后修改lists字段
    for (var i = 0; i < list.length; i++) {
      if (list[i].id === obj.pid) {
        list[i].isLike = true
        list[i].likes.push({
          user: obj.user
        })
      }
    }
  },
  ...
}
```

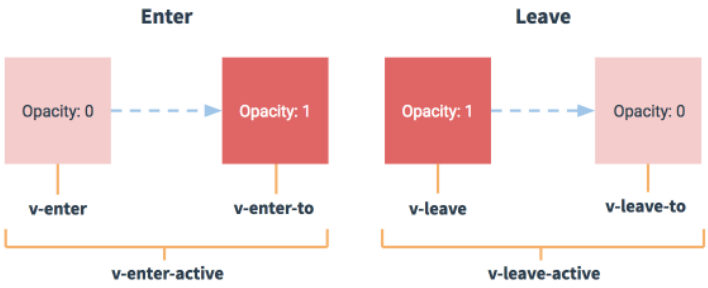
现在大家可以理解为何当时要把列表的数据放进 `store` 来让 vuex 管理了吧，就是为了点赞和评论后能够直接拿到列表的数据的数据，然后进行无刷新更新。评论相关的代码我们会在后面的章节讲解。

小结：

本章节主要讲解了单条内容组件 `postItem` 的前端逻辑开发，同时介绍了在 vue 里如果利用 `transition`，实现一个过渡动画。

相关技术点：

1. 在移动端实现 1px 边框主要利用了 `transform` 的 `scaleY` 来进行缩放，即将 1px 的线缩放之后在手机端看就会比较细，相当于 0.5px。
2. `transition` 是 vue 里使用动画的内置组件，在使用时需要定义不同状态的 class 值来实现动画效果，下面这个图可以比较清晰的展示各个状态：



3. 利用 vuex 来实现跨组件的数据通信来达到无刷新更新 UI，在实现点赞和评论操作时，修改 `store` 里的朋友圈列表数据 `wecircleDataList` 来实现。

本章节完整源代码地址：

Github

}