11 发布页 UI 界面开发

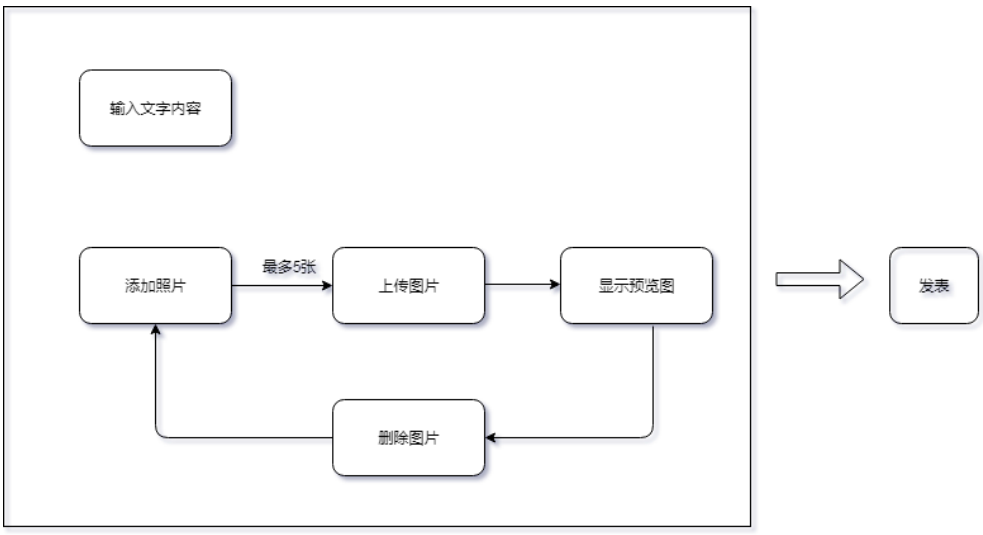
更新时间：2019-09-04 13:58:44

> 学习从来无捷径，循序渐进登高峰。
>
> —— 高永祚

发表功能是一个项目的核心之一，一个功能完备，用户体验优良的发表页面可以提升用户发表的欲望，直接影响到整个项目内容的质量和多样性，本章节将要讲解发表页面的相关UI开发，会用到 weui 和HTML5 `<input>` 的相关技术，准备好让我们进入开发。

发表页面逻辑图：

首先来看看页面的逻辑流程图：



UI效果：

然后是最终要实现的UI效果图：

发表页面逻辑开发

整个发表页面的UI是相对来说比较简单的，页面上的UI元素很少，下面我们一起来实现一下。

新建发布页面

在前端项目的 views 文件夹下新建 publish 文件夹，同时新建 index.vue，编写template：

```
<template>
 <div class="container">
  <div class="header">
   <div class="cancel" @click="cancel">取消</div>
   <div class="create weui-btn weui-btn_mini weui-btn_primary" @click="publish">发表</div>
  </div>
  <div class="input-content">
   <textarea
    class="weui-textarea"
    @input="oninput"
    placeholder="这一刻的想法.."
    v-model="content"
    maxlength="70"
   ></textarea>
   <div class="weui-textarea-counter">
    <span>{{this.textCount}}</span>/70
   </div>
  </div>
  <ul class="img-content">
   <div id="uploaderPub">
    <div class="weui-uploader">
     <div class="weui-uploader__bd">
      <ul class="weui-uploader__files" id="uploaderFiles" @click="showImg($event)"></ul>
      <div class="weui-uploader__input-box">
       <input
        id="uploaderInputPub"
        class="weui-uploader__input"
        type="file"
        accept="image/*"
        multiple="multiple"
       >
      </div>
     </div>
     <div class="weui-uploader__hd">
      <div class="weui-uploader__info">
       <span id="uploadCount">{{uploadCount}}</span>/5
      </div>
     </div>
    </div>
   </div>
  </ul>
 </div>
</template>
```

上面的 UI 基本按照 weui 的方式创建，class 和标签可以直接复制即可，主要说明一下 input 上传文件的使用：

在手机端HTML5页面上，使用 `<input type="file">` 可以实现文件上传效果。

accept：限制上传文件的类型，`image/png,image/gif` 表示只能上传图片类型，并且扩展名是png或gif，`image/*` 表示任何图片类型的文件，当然accept属性也支持 `.xx`，表示扩展名为标识的限制，例如 `accept=".pdf,.doc"`。

multiple：设置是否支持选多个文件，选择支持后，files将会得到一个数组。同时在呼起相册面板时，可以进行打勾多选。

capture：capture 属性可以调用系统默认相机、摄像和录音功能，同时有其他取值：

capture=“camera” 相机
capture=“camcorder” 摄像机
capture=“microphone” 录音

调用 weui 图片上传组件：

在 index.vue 中新增一个 mounted 方法:

下面这段代码是用来初始化我们的 weui 图片上传uploader组件，需要注意的是放在 mounted 里是为了保证UI已经渲染完成，这样才能找到相关的dom节点进行初始化。

```
mounted () {
  let self = this
  weui.uploader('#uploaderPub', {//id为dom的id
    url: service.baseURL + 'post/uploadimg',//上传服务的后台接口，返回值需要使用json格式
    auto: true,//选择完图片后立刻上传
    type: 'file',//上传类型, file为文件上传; base64为以base64上传
    fileVal: 'image',//文件上传域的name，这里的配置和后台接收上传图片的字段保持一致
    compress: {//压缩配置
      width: 1300,//图片的最大宽度
      height: 1300,//图片的最大高度
      quality: 0.8//压缩质量, 取值范围 0 ~ 1
    },
    onBeforeQueued: function (files) {
      // `this` 是轮询到的文件, `files` 是所有文件
      if (
        ['image/jpg', 'image/jpeg', 'image/png', 'image/gif'].indexOf(
          this.type
        ) < 0//只允许上传这几个类型的图片
      ) {
        weui.alert('请上传图片')
        return false // 阻止文件添加
      }
      if (this.size > 10 * 1024 * 1024) {
        weui.alert('请上传不超过10M的图片')
        return false
      }
      if (files.length > 5) {
        // 防止一下子选择过多文件
        weui.alert('最多只能上传5张图片，请重新选择')
        return false
      }
      if (self.uploadCount + 1 > 5) {
        weui.alert('最多只能上传5张图片')
        return false
      }
      //已经上传的图片个图
      ++self.uploadCount

      // return true; // 阻止默认行为，不插入预览图的框架
    },
    onBeforeSend: function (data, headers) {
      // console.log(this, data, headers);
      //将token字段放在headers里，API校验
      headers['wec-access-token'] = getCookie('token')

      // return false; // 阻止文件上传
    },
    onProgress: function (procent) {
      // console.log(this, procent);
      // return true; // 阻止默认行为，不使用默认的进度显示

    },
    onSuccess: function (ret) {
      // console.log(this, ret)
      //ret.data是后台上传接口的返回json的数据
      ret.data.id = this.id
      //将选择的图片放在一个self.picList数组里
      self.picList.push(ret.data)
      // return true; // 阻止默认行为，不使用默认的成功态
    },
    onError: function () {

      // console.log(this, err);
      // return true; // 阻止默认行为，不使用默认的失败态
    }
  })
},
```

监听**textarea**输入，动态修改剩余字数：

在`index.vue`中新增如下代码和方法:

```
<textarea
      ...
      @input="oninput"
      ...
    ></textarea>
```

```
oninput () {
  this.textCount = this.content.length
},
```

上面代码用到了`oninput`事件，下面讲解一下相关的知识点：

常用的监听输入事件，在PC上时，我们可能会用 `keyup` 或者 `keydown` 事件来监听，但是到了移动端，主要有下面两个事件来监听用户的输入行为：

oninput: 事件在用户输入时，value改变时触发，是实时的。通过 js 改变 value 时，不会触发。

onchange: 事件在用户输入时，value改变（两次内容有可能还是相等的）且失去焦点时触发。通过 js 改变 value 时，不会触发。

onpropertychange: 事件在用户输入时，value改变时触发，是实时的。通过 js 改变 value 时，会触发。IE专属。

上面这些事件的介绍比较详细，针对我们这个场景，采用oninput事件最合理。下面我们来看下删除选中图片这一功能的逻辑：

在`index.vue`中新增一个`showImg`和`deleteImg`方法:

```
/*
* 点击已经上传的预览图
*/
showImg (e) {
  let target = e.target

  if (target.classList.contains('weui-uploader__file')) {
    //找到点击的那个dom元素
    let url = target
      //通过正则找到绑定在background-image上的图片url
      .getAttribute('style')
      .match(/url\((.*?)\)/)[1]
      .replace(/"/g, '')

    //调用weui的gallery组件查看大图和是否删除的按钮
    let gallery = weui.gallery(url, {
      onDelete: () => {
        //将删除的图片从已经上传的列表里删除
        this.deleteImg(target, gallery)
      }
    })
  }

},
/*
* 删除已上传的图片
*/
deleteImg (target, gallery) {
  //从target上得到点击的那个图片的id序号
  let id = target.getAttribute('data-id')
  //删除前给一个确认提示
  if (confirm('确定删除该图片？')) {
    console.log('删除')
  }
  //从self.picList数组里找到这个图片并删除
  for (var i = 0, len = this.picList.length; i < len; ++i) {
    var file = this.picList[i]
    if (file.id == id) {
      this.picList.splice(i - 1, 1)
      break
    }
  }
  //将图片的dom移除
  target.remove()
  //隐藏gallert图片查看器
  gallery.hide()
  //已上传图片个数减1
  this.uploadCount--
},
```

上面这段代码主要实现了2个功能：

当我们点击已经上传的图片时，通过`showImg()`方法，会出现一个gallery图片查看器组件来查看大图。

当gallery图片查看器出现之后，在查看器的底部有一个删除按钮，点击之后通过`deleteImg()`方法将已选的图片删除。

gallery 是一个 weui 的图片查看组件，当然功能比较单一，支持传一个图片，不过不用担心，后面我们会开发一个功能更强的图片查看器。

className： string 自定义类名。

onDelete： function 点击删除图片时的回调。

效果如下所示：

代码里的background-image和data-id都是由upload组件上传之后生成在页面dom里的，如下图：

OK，到此我们的发表界面的前端逻辑已经完成，后面会接着开发后端逻辑。

小结

本章主要讲解了发表页面的前端UI逻辑开发，业务逻辑比较多一些。
相关技术点:

1. HTML5中使用来实现图片上传，其中 accept ， multiple ， capture 属性的用法。
2. weui的图片上传组件uploader使用方法以及gallery图片查看器使用方法。
3. 在使用weui的uploader组件之后，初始化方法要放在 mounted 里面，这样可以保证UI存在的情况下初始化成功。

本章节完整源代码地址：Github

}

10 JSON Web Token实现用户验证

12 upload 图片上传接口