作业 4——前端图片处理说明文档

191250024　丁炳智　软件学院

一、文件结构说明

此次作业基于前三次作业，并为解决跨域问题而基于第三次作业的框架。涉及的文件主要包括：

1) HTML：/public/pages/photo-process.html 及/public/pages/tailor.html

2) JavaScript：/public/js/photo_process.js 及/public/js/util/*

3) CSS：/public/style/Process.css

二、作业说明

0、进入主页：启动 App.js 后，在浏览器中输入 http://127.0.0.1:3000 即可访问索引页，注册账号并成功登录后，即可进入个人主页并看到瀑布流图片展示。

1、基础滤镜功能

用户通过点击瀑布流中的任意一张图片，即可进入该图片的基础滤镜处理页面。基础滤镜包括“复古”、“锐化”、“老照片”、“负片”、“明艳”、“鲜红”，基础滤镜的功能基于 LenaJS 实现。

在进入二级页面后，页面默认展示“复古”滤镜效果，代码中通过将原图路径传递到 Lena 函数体中，并指定使用的滤镜，即可获得转化后的图片（画布形式）。在编码过程中出现了画布内联样式不能更改的问题，使用!important 即可解决。此外，认为滤镜不可叠加，因此再次选择其他滤镜时，将会基于原图像进行处理，代码主要逻辑见下。

```
function addSimpleFilter(imgSrc, filterName) {
  let filteredImageCanvas = document.createElement( tagName: 'canvas')
  filteredImageCanvas.setAttribute( qualifiedName: 'id', value: 'lena-canvas')
  let filter = LenaJS[filterName]
  LenaJS.filterImage(filteredImageCanvas, filter, imgSrc)
  return filteredImageCanvas
}

function handleChangeSimpleFilter(clickBtn, realImage) {
  // 将之前按钮恢复默认色
  $('.picked').removeClass( value: 'picked')

  // 滤镜操作
  // 选取了哪种滤镜
  let chooseText = clickBtn.text()
  let filterChosen = ''
  if (chooseText.indexOf( value: '复古') !== -1) {
    filterChosen = 'sepia'
  } else if (chooseText.indexOf( value: '锐化') !== -1) {
    filterChosen = 'gaussian'
  } else if (chooseText.indexOf( value: '老照片') !== -1) {
    filterChosen = 'grayscale'
  } else if (chooseText.indexOf( value: '负片') !== -1) {
    filterChosen = 'invert'
  } else if (chooseText.indexOf( value: '明艳') !== -1) {
    filterChosen = 'saturation'
  } else if (chooseText.indexOf( value: '鲜红') !== -1) {
    filterChosen = 'red'
  }
```

在基础滤镜页面中，用户可以通过选择点击其他滤镜按钮进行滤镜更换，并在右侧图片中查看滤镜效果，也可右键另存为进行下载，示意图如下。

（基础滤镜：“复古”示意图）

（基础滤镜：“明艳”示意图）

2、图像裁切等功能

若用户希望对图像进行裁切、旋转、缩放等处理工作，可以在基础滤镜页面中点击“图像裁切→”按钮进入操作图像页面。此功能基于 CropperJS 库实现。

在编码过程中，遇到了$(…).cropper is not a function 的报错问题，经反复更改后，发现引入 jquery-cropper.js 即可解决，但仍要注意 html 文件中<script>标签的前后顺序。代码中，通过监听左侧功能栏按钮点击情况，对不同操作作出不同反应，代码逻辑如下。

```
$(function () {
  const photoPath = decodeURI(location.search.substr(11))
  const originalImage = document.getElementById('image')
  originalImage.setAttribute('src', photoPath)

  let image = $('#image')
  image.cropper({aspectRatio: NaN...})

  $('.ratio-btn').click(function () {...})

  $('.rotate-btn').click(function () {...})

  $('.scale-btn').click(function () {...})
})
```

代码详情请参看/public/pages/tailor.html 中<script>标签内容。

此页面中，包括不同比例裁切图像功能、图像旋转功能、图像镜像翻转功能及图像缩放功能，用户可通过页面左侧的功能栏进行操作，并在下方小图中看到操作后的效果，示意图如下。

注：图片的缩放功能可直接在图片处理区域使用鼠标滚轮进行缩放。

具体图像操作功能展示如下：

（左：图像裁剪；右：图像缩放）

（左：图像逆时针旋转 45°；右：图像水平翻转）

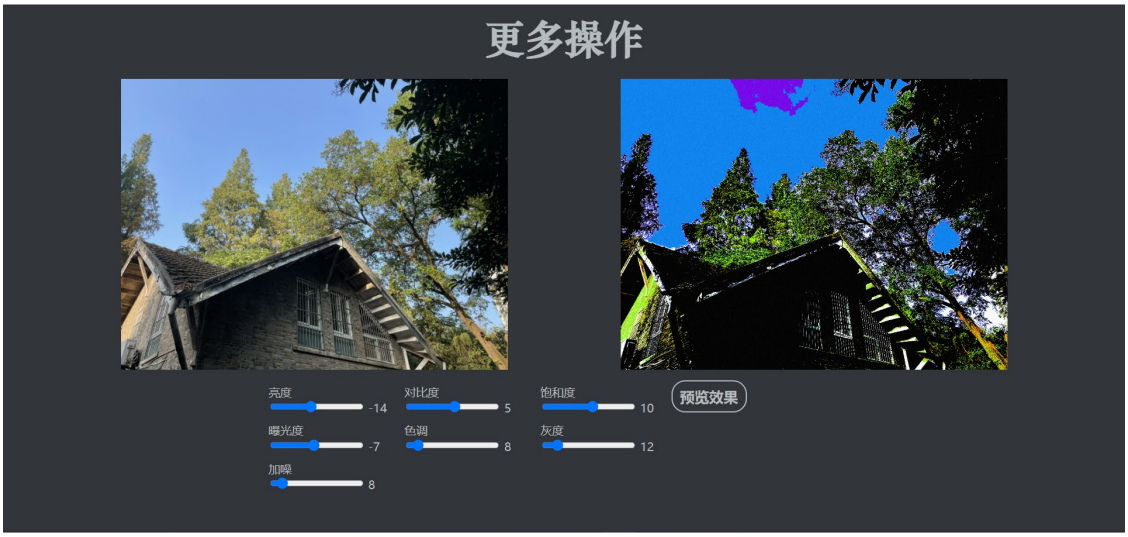
3、用户自定义参数功能

若用户有更多图片处理的需求，并希望自己定义相关的参数（如亮度、对比度、饱和度、曝光度、色调、灰度、噪声值等），则可以在基础滤镜页面点击“自定参数→”按钮进入“更多操作”页面，并根据需求拖拽参数范围条调整参数值。

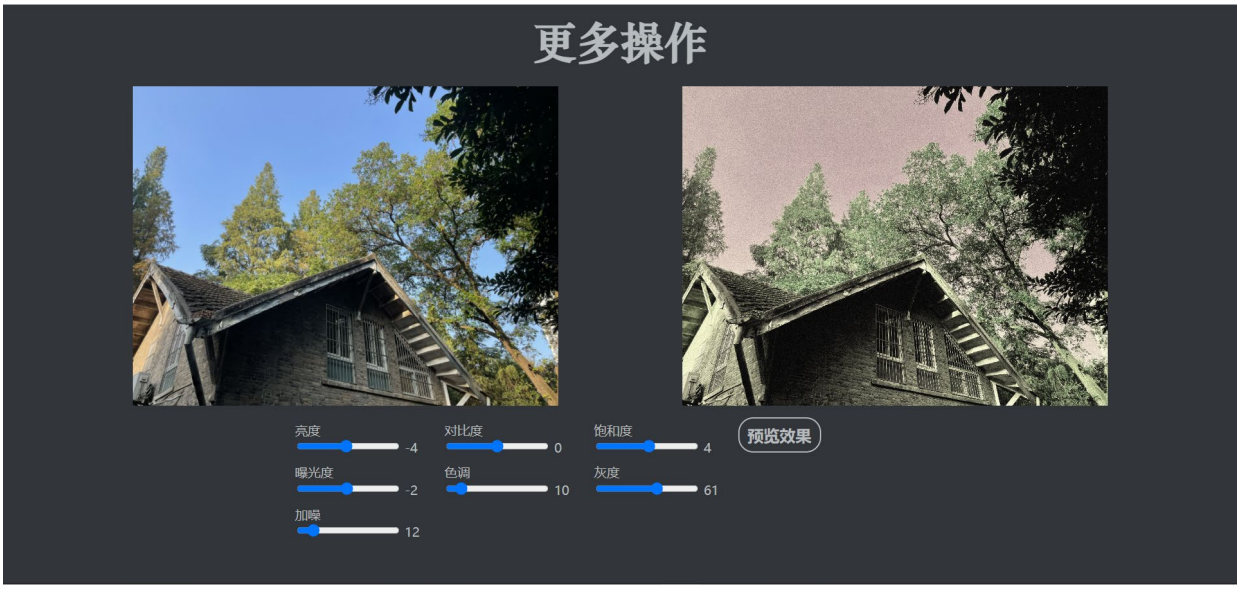
代码实现时，基于<input type=range>元素的取值及元素 id 进行图片处理并渲染，在一次循环中完成对所有图片参数的处理，代码主要逻辑见下。

```
function handleFilterSetting(photoPath) {
  $('#check-btn').text( value: '渲染中')

  Caman('#caman-canvas', photoPath, function () {
    this.revert( updateContext: false)
    let ranges = $('input[type=range]')
    for (let i = 0; i < ranges.length; i++) {
      let setParam = ranges.eq(i).attr( name: 'id')
      let filterVal = ranges.eq(i).val()

      if (setParam === 'brightness') this.brightness(filterVal)
      else if (setParam === 'contrast') this.contrast(filterVal)
      else if (setParam === 'saturation') this.saturation(filterVal)
      else if (setParam === 'exposure') this.exposure(filterVal)
      else if (setParam === 'hue') this.hue(filterVal)
      else if (setParam === 'sepia') this.sepia(filterVal)
      else if (setParam === 'noise') this.noise(filterVal)
    }

    this.render()
  })
}
```

所有参数值调整好后，点击“预览效果”按钮即可在页面右侧看到处理后的图像，示意图如下。



（自定参数示意图 1）



（自定参数示意图 2）

本页面功能基于 `CamanJS` 库进行实现。实质上是对原图的每个像素点进行通道操作，将转化后的图片绘制到页面右侧的画布上。用户通过右键另存为即可下载处理好的图片。