1. 搜索框组件
   1. 预期效果



图1-1 搜索框效果图



图1-2 普通按钮效果图（颜色和文本可改变）

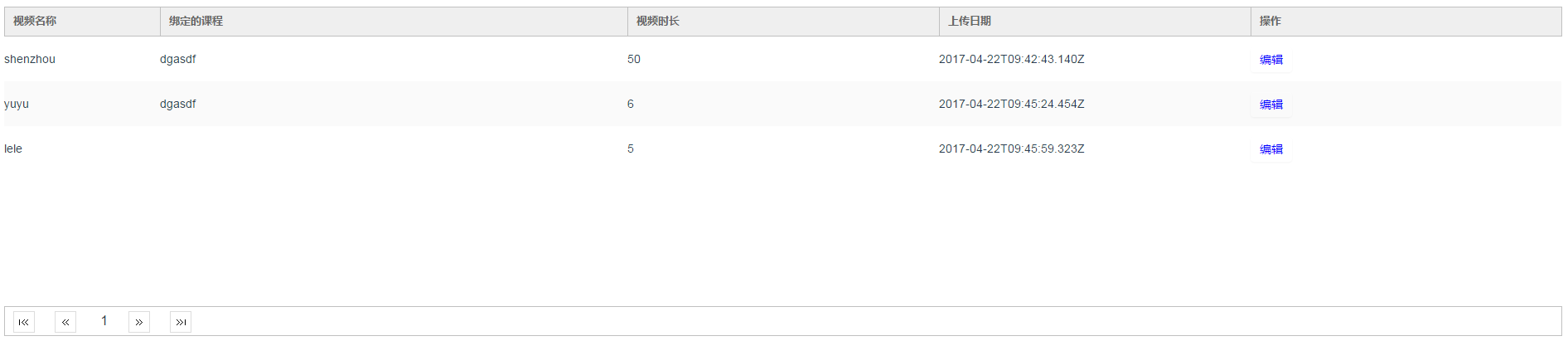


图1-3 表格效果图（表头，操作按钮绑定方法随给定数据而改变）

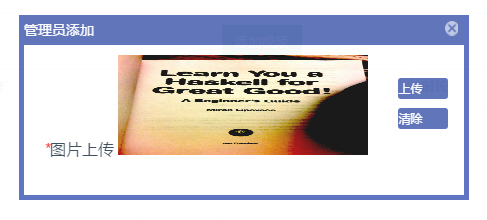


图 1-4 弹窗效果图（可变宽高，可变内容）

* 1. 掌握技能

表1-1 任务说明

|  |  |  |
| --- | --- | --- |
| 任务名称 | 子任务 | 任务内容 |
| Vue-router,vuex的使用 | 1 | Vue-router,vuex的引入和使用 |
| 背景组件的开发 | 2 | 组件的html,css,js |
| 搜索框组件 | 3 | 组件的html,css,js |
| 普通按钮组件 | 4 | 组件的html,css,js |
| 表格组件 | 5 | 组件的html,css,js |
| 弹窗组件 | 6 | 组件的html,css,js |

* 1. 任务分解
     1. vue-router,vuex的引入
        1. 在当前项目目录下执行以下命令：

npm install vue-router  
npm install vuex

如安装失败请检查源和node.js，npm版本

* + - 1. vue-router的使用：

//main.js

import Vue from 'vue';

//路由

import VueRouter from "vue-router";

Vue.use(VueRouter);

// 定义组件, 也可以像教程之前教的方法从别的文件引入

import login from './component/login\_component.vue';

import second from './component/second.vue';

// 创建一个路由器实例

// 并且配置路由规则

const router = new VueRouter({

mode: 'history',

base: \_\_dirname+"/",

routes: [

{

path: '/',

component: login

},

{

path: '/second',

component: second

}

]

});

// 现在我们可以启动应用了！

// 路由器会创建一个 App 实例，并且挂载到选择符 #app 匹配的元素上。

const app = new Vue({

router:router

}).$mount('#app')

//index.html文件更改

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>vuefcxy</title>

<style>

body {

margin:0;

padding:10px;

background:#F5F5F5;

}

</style>

</head>

<body>

<div id="app">

<div>

<!-- 4、<router-link>默认会被渲染成一个 `<a>` 标签 ,to指令跳转到指定路径 -->

<router-link to="/first"></router-link>

<router-link to="/second"></router-link>

</div>

<!--&lt;!&ndash; 5、在页面上使用<router-view></router-view>标签，用于渲染匹配的组件 &ndash;&gt;-->

<router-view></router-view>

</div>

<script src="/dist/build.js">

</script>

</body>

</html>

* + - 1. vuex的使用：

项目src目录下新建vuex文件夹，vuex内新建store.js文件，内容如下：

import Vue from 'vue';

import Vuex from 'vuex'

// 告诉 vue “使用” vuex

Vue.use(Vuex);

const state = {

// 应用启动时，配置项目中使用到的url的前缀

url:{

BASEURL:'http://127.0.0.1:3000'//服务器IP和端口

}

}

const mutations = {

// mutation 的第一个参数是当前的 state

// 你可以在函数里修改 state

// 修改BASEURL

UPDATAEBASEURL:function(state,value){

state.url.BASEURL=value;

}

}

// 整合初始状态和变更函数，我们就得到了我们所需的 store

// 至此，这个 store 就可以连接到我们的应用中

export default new Vuex.Store({

state,

mutations

})

//main.js更改

import Vue from 'vue';

//vuex

import Vuex from 'vuex';

//路由

import VueRouter from "vue-router";

Vue.use(Vuex);

Vue.use(VueRouter);

;

// 定义组件, 也可以像教程之前教的方法从别的文件引入

import login from './component/login\_component.vue';

import second from './component/second.vue';

// 创建一个路由器实例

// 并且配置路由规则

const router = new VueRouter({

mode: 'history',

base: \_\_dirname+"/",

routes: [

{

path: '/',

component: login

},

{

path: '/second',

component: second

}

]

});

// 现在我们可以启动应用了！

// 路由器会创建一个 App 实例，并且挂载到选择符 #app 匹配的元素上。

const app = new Vue({

router:router,

store

}).$mount('#app')

* + 1. 背景canvas组件

在当前项目下的/src/component目录下创建bgCanvas.vue文件，内容如下：

<template>

<canvas class="bgCanvas" width="100%" height="100%"></canvas>

</template>

<script>

export default {

name: 'login',

data () {

return {

msg: ""

}

},

mounted:function(){

var canvas = document.getElementsByClassName('bgCanvas')[0],

ctx = canvas.getContext('2d')

canvas.width = window.innerWidth;

canvas.height = window.innerHeight;

ctx.lineWidth = .3;

ctx.strokeStyle = (new Color(150)).style;

var dotsNomber = 400;

if(canvas.width < 1400){

dotsNomber = 200;

}

var mousePosition = {

x: 30 \* canvas.width / 100,

y: 30 \* canvas.height / 100

};

var dots = {

nb: dotsNomber,

distance: 100,

d\_radius: 150,

array: []

};

function colorValue(min) {

return Math.floor(Math.random() \* 255 + min);

}

function createColorStyle(r,g,b) {

return 'rgba(' + r + ',' + g + ',' + b + ', 0.8)';

}

function mixComponents(comp1, weight1, comp2, weight2) {

return (comp1 \* weight1 + comp2 \* weight2) / (weight1 + weight2);

}

function averageColorStyles(dot1, dot2) {

var color1 = dot1.color,

color2 = dot2.color;

var r = mixComponents(color1.r, dot1.radius, color2.r, dot2.radius),

g = mixComponents(color1.g, dot1.radius, color2.g, dot2.radius),

b = mixComponents(color1.b, dot1.radius, color2.b, dot2.radius);

return createColorStyle(Math.floor(r), Math.floor(g), Math.floor(b));

}

function Color(min) {

min = min || 0;

this.r = colorValue(min);

this.g = colorValue(min);

this.b = colorValue(min);

this.style = createColorStyle(this.r, this.g, this.b);

}

function Dot(){

this.x = Math.random() \* canvas.width;

this.y = Math.random() \* canvas.height;

this.vx = -.5 + Math.random();

this.vy = -.5 + Math.random();

this.radius = Math.random() \* 2;

this.color = new Color();

//console.log(this);

}

Dot.prototype = {

draw: function(){

ctx.beginPath();

ctx.fillStyle = this.color.style;

ctx.arc(this.x, this.y, this.radius, 0, Math.PI \* 2, false);

ctx.fill();

}

};

function createDots(){

for(var i = 0; i < dots.nb; i++){

dots.array.push(new Dot());

}

}

function moveDots() {

for(var i = 0; i < dots.nb; i++){

var dot = dots.array[i];

if(dot.y < 0 || dot.y > canvas.height){

dot.vx = dot.vx;

dot.vy = - dot.vy;

}

else if(dot.x < 0 || dot.x > canvas.width){

dot.vx = - dot.vx;

dot.vy = dot.vy;

}

dot.x += dot.vx;

dot.y += dot.vy;

}

}

function connectDots() {

for(var i = 0; i < dots.nb; i++){

for(var j = 0; j < dots.nb; j++){

var i\_dot = dots.array[i];

var j\_dot = dots.array[j];

if((i\_dot.x - j\_dot.x) < dots.distance && (i\_dot.y - j\_dot.y) < dots.distance && (i\_dot.x - j\_dot.x) > - dots.distance && (i\_dot.y - j\_dot.y) > - dots.distance){

if((i\_dot.x - mousePosition.x) < dots.d\_radius && (i\_dot.y - mousePosition.y) < dots.d\_radius && (i\_dot.x - mousePosition.x) > - dots.d\_radius && (i\_dot.y - mousePosition.y) > - dots.d\_radius){

ctx.beginPath();

ctx.strokeStyle = averageColorStyles(i\_dot, j\_dot);

ctx.moveTo(i\_dot.x, i\_dot.y);

ctx.lineTo(j\_dot.x, j\_dot.y);

ctx.stroke();

ctx.closePath();

}

}

}

}

}

function drawDots() {

for(var i = 0; i < dots.nb; i++){

var dot = dots.array[i];

dot.draw();

}

}

function animateDots() {

ctx.clearRect(0, 0, canvas.width, canvas.height);

moveDots();

connectDots();

drawDots();

requestAnimationFrame(animateDots);

}

canvas.onmousemove=function(e){

mousePosition.x = e.pageX;

mousePosition.y = e.pageY;

}

canvas.onmouseleave= function(e){

mousePosition.x = canvas.width / 3;

mousePosition.y = canvas.height / 3;

}

createDots();

requestAnimationFrame(animateDots);

},

computed: {

},

methods:{

},

components: {}

};

</script>

<style>

</style>

* + 1. 搜索框组件
       1. 在当前项目下的/src/component目录下创建input\_default.vue文件，内容如下：

<template>

<input v-bind:style="input\_default" v-bind:value="defaultObj.defaultStr.value" v-on:focus="fs" v-on:keyup="inp" v-on:blur="bl">

</template>

<script type="text/javascript">

import store from '../vuex/store.js';

var arr = [];

export default {

data () {

return {

input\_default:{}

}

},

props:['width','height','defaultObj'],

mounted:function(){

this.input\_default={

'display':'inline-block',

'width':this.width?typeof this.width==="number"?this.width+"px":this.width?this.width:"100%":0,

'height':this.height?this.height:"30px",

'border':'1px solid #C0C0C0'

}

},

methods:{

fs: function(event) {

if(this.defaultObj&&this.defaultObj.fs){

try{

this.defaultObj.fs(event);

}catch(e){

if(e==="跳出函数"){

event.target.blur();

return false

}

}

}else{

return false;

}

if(this.defaultObj&&this.defaultObj.defaultStr&&event.target.value===this.defaultObj.defaultStr.value){

event.target.value="";

}

},

inp:function(){

if(this.defaultObj&&this.defaultObj.inp){

this.defaultObj.inp();

}else{

return;

}

},

bl:function(event){

if(this.defaultObj&&this.defaultObj.bl){

this.defaultObj.bl(event);

}else{

return;

}

}

}

}

</script>

<style>

</style>

* + 1. 普通按钮组件
       1. 在当前项目下的/src/component目录下创建btn.vue文件，内容如下：

<template>

<button class="btn\_class" v-bind:style="props\_class" v-text="btn\_text" v-on:click="btn\_click()"></button>

</template>

<script type="text/javascript">

import store from '../vuex/store.js';

var arr = [];

export default {

store:store,

name: 'input\_default',

props:['props\_btn\_data\_obj'],

data () {

return {

props\_class:{},

btn\_text:""

}

},

mounted:function(){

this.props\_class={

'background':this.props\_btn\_data\_obj.background

};

this.btn\_text=this.props\_btn\_data\_obj.btn\_text;

},

methods:{

btn\_click: function() {

if(this.props\_btn\_data\_obj&&this.props\_btn\_data\_obj.btn\_click){

this.props\_btn\_data\_obj.btn\_click();

}else{

return false;

}

}

}

}

</script>

<style>

.btn\_class{

width:80px;

height:32px;

margin:0;

padding:0;

border:none;

cursor:pointer;

color:#fff;

margin-right:10px;

}

</style>

* + 1. 表格组件
       1. 在当前项目下的/src/component目录下创建tableList.vue文件，内容如下

<template>

<div class="box">

<table class="tableList\_table table-hover table-bordered">

<thead>

<tr>

<th v-for="data in th\_lists" v-bind:width="data.width" v-text="data.value"></th>

</tr>

</thead>

<tbody>

<tr v-for="(data,index) in lists" v-bind:class="judge\_oddOrevent(index)" v-bind:id="data.clickClass" v-on:click="classChang($event,data,index)">

<td v-for="(da,ind) in th\_lists" v-if="ind<th\_lists.length-1" v-html="data[da.trueName]"></td>

<td>

<button class="btn btn-default" @click="putawayData(data)" v-if="data.putaway" v-text="data.putawayText">上架</button>

<button class="btn btn-default" @click="update($event,data)" v-if="data.edit">编辑</button>

<button class="btn btn-default" @click="deleteRow($event,data)" v-if="data.deleteButton">删除</button>

<button class="btn btn-default" style="width:60px" @click="stickData(data)" v-if="data.stick" v-text="data.stickText">置顶</button>

<button class="btn btn-default" @click="blocking(data)" v-if="data.blocking" v-text="data.status==='正常'?'冻结':'解冻'"></button>

<button class="btn btn-default" @click="addVideo(data)" v-if="data.add" >添加</button>

</td>

</tr>

<tr>&nbsp;</tr>

<tr>&nbsp;</tr>

<tr>&nbsp;</tr>

</tbody>

<tfoot v-show="tfootShow">

<tr>

<td v-bind:colspan="th\_lists.length" class="paging\_class">

<div >

<ul class="">

<li>

<a @click="onFirstClick()">

<img src="../assets/first.png">

</a>

</li>

<li>

<a @click="onPrevClick()">

<img src="../assets/prev.png">

</a>

</li>

<li style="border:none;" >

{{counts.value}}

</li>

<li>

<a @click="onNextClick()">

<img src="../assets/next.png">

</a>

</li>

<li>

<a @click="onLastClick()">

<img src="../assets/last.png">

</a>

</li>

</ul>

</div>

</td>

</tr>

</tfoot>

</table>

</div>

</template>

<script>

import paging from './paging.vue'

export default {

data () {

return {

lenArr: [5, 50, 100], // 每页显示长度设置

pageLen: 5, // 可显示的分页数

url: '/bootpage/', // 请求路径

param: {}, // 传递参数

//调用组件时需传的数据

tr\_click\_class:"tableList\_table\_tbody\_tr\_click",//控制列表行的点击样式

th\_lists:this.tableListData.th\_lists,

lists: this.tableListData.lists, // 表格加工后数据

tfootShow:this.tableListData.btnList.tfootShow,//是否显示table的foot

counts: this.tableListData.count,// 表格原始数据

}

},

props:['tableListData','isComponent'],

created:function(){//组件创建后初始化数据

//this.getData();

},

mounted:function(){

this.$forceUpdate();

},

components: {

paging

},

methods: {

getData:function(){//获取数据

this.$http({

url: "http://localhost:3000/"+this.database+"/show",

method: 'GET',

data: {}

})

.then(function (response) {

if (!response.data.data.length) {

alert("服务器异常");

}

this.lists=response.data.data;

})

},

addData:function(data){//添加数据

},

update:function(e,data){//编辑数据

//如果提供了事件对象，则这是一个非IE浏览器

if ( e && e.stopPropagation ){

//因此它支持W3C的stopPropagation()方法

e.stopPropagation();

}else{

//否则，我们需要使用IE的方式来取消事件冒泡

window.event.cancelBubble = true;

}

if(this.tableListData.btnList.update){

this.tableListData.btnList.update(data);

}

},

deleteRow:function(e,data){//删除数据

//如果提供了事件对象，则这是一个非IE浏览器

if ( e && e.stopPropagation ){

//因此它支持W3C的stopPropagation()方法

e.stopPropagation();

}else{

//否则，我们需要使用IE的方式来取消事件冒泡

window.event.cancelBubble = true;

}

if(this.tableListData.btnList.deleteRow){

this.tableListData.btnList.deleteRow(data);

}

},

blocking:function(data){//冻结处理

if(this.tableListData.btnList.blocking){

this.tableListData.btnList.blocking(data);

}

},

addVideo:function(data){

if(this.tableListData.btnList.addVideo){

this.tableListData.btnList.addVideo(data);

}

},

putawayData:function(data){//处理上下架

if(this.tableListData.btnList.putawayData){

this.tableListData.btnList.putawayData(data);

}

},

stickData:function(data){//置顶与否

if(this.tableListData.btnList.stickData){

this.tableListData.btnList.stickData(data);

}

},

judge\_oddOrevent:function(num){//隔行样式

if(num%2===1){

return "tableList\_tr\_black";

}

},

classChang:function(e,data,num){//鼠标hover行样式

data.clickClass="tableList\_table\_tbody\_tr\_click";

for(var i = 0;i<this.lists.length;i++){

if(i!==num){this.lists[i].clickClass="";}

}

if(this.tableListData.btnList.clickRow){

this.tableListData.btnList.clickRow(e,data,num);

}

},

onNextClick:function(){

this.tableListData.btnList.NextClick();

},

onPrevClick:function(){

this.tableListData.btnList.PrevClick();

},

onLastClick:function(){

this.tableListData.btnList.LastClick();

},

onFirstClick:function(){

this.tableListData.btnList.FirstClick();

}

},

events: {

// 分页组件传回的表格数据

'data' (data) {

this.tableList = data

}

}

}

</script>

<style>

.tableList\_table {

width:100%;

height:400px;

background-color:#FFFFFF;

border-collapse:collapse;

}

.tableList\_table thead tr{

color:#666666;

background-color:#EFEFEF;

border:1px solid #C0C0C0;

height:34px;

font-size:13px;

}

.tableList\_table tbody tr{

height:34px;

font-size:14px;

}

.tableList\_table tbody tr td{

height:34px;

font-size:14px;

}

.tableList\_table thead tr{

border-right:none;

}

.tableList\_table thead tr th{

padding-left:10px;

border-right:1px solid #C0C0C0;

}

.tableList\_tr\_black {

background-color:#FAFAFA;

}

.tableList\_table tbody tr:hover {

background-color:#ECF4FF;

}

.tableList\_table tbody button {

margin:0;

padding:0;

list-style-type:none;

color:blue;

background:none;

border-top-style: none;

border-right-style: none;

border-bottom-style: none;

border-left-style: none;

width:50px;

height:30px;

line-height:30px;

}

.tableList\_table tbody button:hover{

cursor:pointer;

}

#tableList\_table\_tbody\_tr\_click {

border:1px solid #3892D3;

}

.tableList\_table tfoot {

width:100%;

}

.tableList\_table tfoot tr {

width:100%;

height:30px;

border:1px solid #C0C0C0;

}

.tableList\_table tfoot tr td{

height:30px;

}

.tableList\_table tfoot tr td li{

border:none;

}

.tableList\_table tfoot tr td li img{

vertical-align: bottom;

margin-bottom:3px;

}

.tableList\_table tfoot tr td li img:hover{

cursor:pointer;

}

.paging\_class {

text-align:left;

}

.box {

padding: 0;

}

</style>

* + 1. 弹窗组件
       1. 在当前项目下的/src/component目录下创建tableList.vue文件，内容如下

<template>

<div v-bind:style = "alertClass">

<div v-bind:style ="alertHol">

<div style=" height:30px;background:#4F66B3;margin-bottom:10px">

<span style="display:inline-block;float:left;color:white;line-height:30px;font-size:14px">修改密码</span><img v-on:click="AlertInputArr.btn[1].btn\_click" style="display:inline-block;float:right;margin-top:5px;margin-right:5px" src="../assets/images/x.png">

</div>

<ul>

<li v-for="item in this.AlertInputArr.list" style="display:block;margin-bottom:10px">

<span style="display:inline-block;width:80px;text-align:right">{{item.rowName}}:</span><input\_default width="item.width" height="20px" v-bind:defaultObj="item.defaultObj"></input\_default>

</li>

</ul>

<div style="text-align:center">

<btn v-bind:props\_btn\_data\_obj="AlertInputArr.btn[0]"></btn><btn v-bind:props\_btn\_data\_obj="AlertInputArr.btn[1]"></btn>

</div>

</div>

</div>

</template>

<script type="text/javascript">

import input\_default from "./input\_default.vue";

import btn from "./btn.vue";

export default {

name: 'btn\_add',

data () {

return {

alertClass: {

'width':window.innerWidth+"px",

'height':window.innerHeight+"px",

background:"white",

'z-index':1050,

'position':'absolute',

'top':0,

'left':0,

'opacity':0.9

},

alertHol:{},

alertHolTop:{},

AlertInputArr:this.$safeAlert.state.AlertInputArr

}

},

computed:{

},

mounted:function(){

var width = (function(arr){

for(var i = 1;i<arr.length;i++){

if(arr[i-1].width>arr[i].width){

var temp = arr[i-1].width;

arr[i-1].width = arr[i].width;

arr[i].width = temp;

}

};

return arr[arr.length-1].width;

})(this.AlertInputArr.list)+110;

var height = this.AlertInputArr.list.length\*30+60+30;

this.alertHol = {

'padding-top':0,

'position':'relative',

'background':'#FFFFFF',

'border':'5px solid #4F66BB',

'border-top':'none',

'width':width+'px',

'height':height+"px",

'top':window.innerHeight/2-height/2+'px',

'left':window.innerWidth/2-width/2+'px'

}

this.alertHolTop = {

'height':"30px",

}

},

beforeDestroy:function(){

},

components: {input\_default,btn}

}

</script>

<style>

.alert56HolId {

min-width:210px;

min-height:100px;

position:absolute;

top:0;

left:0;

background:#4F66B3;

opacity:0.8;

}

</style>

* 1. 参考资料

无

* 1. 扩展练习

教师自行补充