<!--

1) //this.contents[index]=this.$options.data().contents[index];

Object.assign(this.contents[index],this.$options.data().contents[index]);

//use assign if you assign directly than getters and setters will not reflect in object

2) v-model - is two way binding so if you use it, change will be reflected in object.

3) v-value - is one way yo need to mannually do the things

4) :key - use this otheriwise dom will be refreshed very time on typing or changing somthing

-->

<!DOCTYPE html>

<html>

<head>

<link href="css\bootstrap.min.css" rel="stylesheet" type="text/css">

<style>

.dirty {

border-color: #red;

//background: #F00;

border: solid red;

}

.dirty:focus {

outline-color: #F00;

}

.error {

border-color: red;

background: #FDD;

}

.error:focus {

outline-color: #F99;

}

input:focus {

border-color: red;

}

select:focus {

border-color: red;

}

textarea:focus {

border-color: red;

}

.labelClass {

margin: 0;

display: inline-flex;

;

}

.dragClass {

outline: solid #f00;

background: #FFCC99;

}

.moveClass {

background-image: url('move.gif');

cursor: move;

height: 40px;

!important;

background-repeat: no-repeat;

background-position: center;

width: 30px;

}

/\* pagging\*/

a {

color: #999;

}

.current {

color: red;

}

ul {

padding: 0;

list-style-type: none;

}

li {

display: inline;

margin: 5px 5px;

}

a.first::after {

content: '...'

}

a.last::before {

content: '...'

}

</style>

<script src="axios.js"></script>

<script src="vue.js"></script>

</head>

<body>

<div id="app" class="table-responsive">

<form method='post' id='frm'>

<table border='1' width='50%' align="center" id="itemTable" class="table table-striped table-hover table-sm">

<tbody>

<tr>

<td :colspan="headers.length"><button @click='addTableRow()' type='button' >Add Last Row</button>

<button @click='saveAll()' type='button'>Save All</button>

<button @click='reselAll()' type='button'>Reset All</button></td>

</tr>

<tr >

<th v-for="(content, index) in headers" >{{content.name}}</th>

</tr>

<tr class="filters" >

<td></td>

<td></td>

<td v-for="(colval, colindex) in columns"><input v-if="colval.filtertype==1" :key="colval.filterid" v-model.lazy.trim="colval.filterVal" placeholder="Search" v-on:keyup.enter="addFilter(colval,$event)" class="form-control" >

<select @change="addFilter(colval,$event)" v-else = "colval.filtertype==2" v-model.lazy.trim="colval.filterVal" class="form-control">

<option v-for="(statusval,statusIndex) in status" v-bind:value="statusIndex">{{statusval}}</option>

</select></td>

<td></td>

</tr>

<tr v-for="(item, index) in filtered " :key='index' v-on:dblclick='edit(item.id,index)' draggable="true" @mouseup="mouseUp(index,item.id)" @mouseover="mouseOver(index,item.id)" @mouseout="mouseOut(index,item.id)" v-bind:style="(dragStyleObj!=null && dragStyleObj[index]!=null)?dragStyleObj[index]:'' " scope="row">

<td v-on:click="cancelEv(index,item.id)" class="text-center"><label >{{parseInt(index)+1}}</label></td>

<td class='moveClass' @mousedown="mouseDown(index,item.id)"></td>

<td v-for="(cols, colkey) in columns"><label v-if = "!editingObj[item.id] && cols.inputType==3"> {{status[item[colkey]]}} </label>

<label v-if = "!editingObj[item.id] && cols.inputType!=3"> {{item[colkey]}} </label>

<input v-if = "editingObj[item.id] && cols.inputType==1"

:ref='index' :key="index" :id="index+'\_'+colkey" @blur= "editTodo(index,colkey,$event);" v-model.lazy.trim="item[colkey]" v-bind:class="{dirty:(validateobj!=null && validateobj[index] && validateobj[index].includes(colkey)==true)?true:false}" v-todo-focus="item[colkey] == editing" @keyup.esc="cancelEv(item[colkey])">

<textarea v-if = "editingObj[item.id] && cols.inputType==2" :ref='index'

@blur= "editTodo(index,colkey,$event);" :key='index' :id="index+'\_'+colkey" v-model.lazy.trim="item[colkey]" v-bind:class="{dirty:(validateobj!=null && validateobj[index] && validateobj[index].includes(colkey)==true)?true:false}">

</textarea>

<select @change="editTodo(index,colkey,$event);" :key='index' :id="index+'\_'+colkey" v-if = "editingObj[item.id] && cols.inputType==3" v-model.trim="item.status" :ref='index'

v-bind:class="{dirty:(validateobj!=null && validateobj[index] && validateobj[index].includes(colkey)==true)?true:false}">

<option v-for="(statusval,statusIndex) in status" v-bind:value="statusIndex">{{statusval}}</option>

</select>

<br />

<span v-show="validateobj!=null && validateobj[index] && validateobj[index].includes(colkey)==true" class="invalid"> This field is required. </span></td>

<td><button @click="addAbove(index,item.id)" type='button' :disabled="(editingObj[item.id])?true:false"> add </button>

<span v-if = "!editingObj[item.id]" >

<button @click="edit(item.id,index)" type='button'>Edit</button>

</span> <span v-else="editingObj[item.id]">

<button @click="save()" type='button' >Save</button>

</span><!--:disabled="(validateobj!=null && validateobj[index])?true:false" -->

<span v-if="editingObj[item.id]">

<button @click="cancelEv(index,item.id)" type='button'>Cancel</button>

</span>

<button @click='deleteTableRow(index,item.id)' type='button'>Delete</button></td>

</tr>

<tr>

<td v-if='noData==true' :colspan="headers.length" align="center">No Data</td>

</tr>

<tr>

<td :colspan="headers.length"><button @click='addTableRow()' type='button'>Add Last Row</button>

<button @click='save()' type='button'>Save All</button>

<button @click='reselAll()' type='button'>Reset All</button>

<div style='float:right'>

<ul>

<li v-for="pageNumber in totalPages" v-if="Math.abs(pageNumber - currentPage) < 3 || pageNumber == totalPages - 1 || pageNumber == 0"> <a href="#" @click="setPage(pageNumber)" :class="{current: currentPage === pageNumber, last: (pageNumber == totalPages - 1 && Math.abs(pageNumber - currentPage) > 3), first:(pageNumber == 1 && Math.abs(pageNumber - currentPage) > 3)}">{{ pageNumber }}</a> </li>

</ul>

</div></td>

<td></td>

</tr>

</tbody>

</table>

<br>

</form>

{{contents}}

{{copyObject[0]}} </div>

<script src="jquery.js"></script>

<script src="jquery-ui.js"></script>

<script>

var data=[];

function initialState(params)

{

data=[];

axios(params).then(function(response){

// that.result= response.data;

// that.headers=response.data.headers;

// that.contents=response.data.contents;

// that.columns=response.data.columns;

// that.copyObject=Object.assign([],Object.values(response.data.contents));

data=response.data;

});

console.log(initialData,'respo');

/\*return{

headers:[{name:'Sr No.'},{name:''},{name:'company\_name'},{name:'city'},{name:'turn\_over'},{name:'status'},{name:'action'}],

contents:{

'\_200':{ company\_name: 'Amazon Web Services', city: 'New York', turn\_over: 400,status:2,rec\_id:1,id:200},

'\_100':{ company\_name: 'ABC Infotech', city: 'New Jersey', turn\_over: 100,status:1,rec\_id:3,id:100},

'\_500':{ company\_name: 'Digital Ocean', city: 'Washington', turn\_over: 200,status:3,rec\_id:4,id:500},

'\_22': { company\_name: 'Digital Ocean1', city: 'Washington1', turn\_over: 2020,status:4,rec\_id:2,id:22}

},

columns:{company\_name:{validate:[''],filterid:'company\_name',filtertype:1,inputType:1,filterVal:''},

city:{filterid:'city',filtertype:1,inputType:2,filterVal:''},

turn\_over:{filterid:'turn\_over',filtertype:1,inputType:1,filterVal:'',validate:['']},

status:{filterid:'status',filtertype:2,inputType:3,filterVal:0,validate:['0']}

},

status:{0:'Select Status',1:"No Documents",2:"Documents Linked",3:"Documents Approved",4:"Partially Approved",5:"Not Linked"},

activeFilters: {},

isFilter:false,noData:false,

resultCount:8,

localData:[],editing:{},validateobj:[],editingObj:{}

,copyObject:[],dragStyleObj:{},downIndex:-2,originalobj:[],downID:'',recIds:[],currentPage:0,itemsPerPage:2, axiosFn(){

}

}\*/

}

/\*

var mainVue=Vue.extend({

data () {

return initialState();

}

});\*/

new Vue({

el: '#app',

data(){

return {

result:[],

headers:[],contents:[],columns:[],\_beforeEditingCache :[],

noData:false,

status:{0:'Select Status',1:"No Documents",2:"Documents Linked",3:"Documents Approved",4:"Partially Approved",5:"Not Linked"},

activeFilters: {},

isFilter:false,

resultCount:8,

localData:[],editing:{},validateobj:[],editingObj:{}

,copyObject:[],dragStyleObj:{},downIndex:-2,originalobj:[],downID:'',recIds:[],currentPage:0,itemsPerPage:2,temp:[]

}

},

mounted(){

console.log('mounted 2nd method called',this.headers)

}

,watch:{

deep:true,

copyObject(o,p){

//console.log('watch',o,p);

}

},created(){

this.getPost({ data:{p:1,act:'grid'}});

//initialState({ data:{p:1,act:'grid'}});

console.log('created','1st method called',data)

// this.copyObject=Object.assign([],Object.values(this.contents));

// this.originalobj=initialState().contents;

// for(var p in this.copyObject)

// {

// this.recIds.push(this.copyObject[p].rec\_id);

// }

// this.recIds.sort();

// console.log('createdee ',this.copyObject,this.originalobj)

}

,computed:{

totalPages: function() {

console.log(Math.ceil(this.resultCount / this.itemsPerPage),'totalPages');

return Math.ceil(this.resultCount / this.itemsPerPage)

},

filtered(){

console.log('computed');

var filtered=this.copyObject;

// filtered=this.sortData(filtered);

console.log('filtered',filtered,this.isFilter,this.activeFilters);

this.noData=false;

if(this.isFilter)

{

for(filterkey in this.activeFilters)

{

var filterValue=this.activeFilters[filterkey];

filtered= filtered.filter(function(record){

console.log(new RegExp(filterValue, 'i').test(record[filterkey]),filterValue,record[filterkey]);

return new RegExp(filterValue, 'i').test(record[filterkey])

});

}

}

if(filtered && filtered.length==0)

{

this.noData=true;

}

//filtered=this.paginate(filtered);

console.log('paginate filter');

// this.activeFilters={};

return filtered;

}

}

,

filters:{

}

,

methods:{

getPost(params){

var that=this;

params=$.extend({ method:'post', url:'data.php',data:params,responseType:'json'},params);

axios(params).then(function(response){

that.result= response.data;

that.headers=response.data.headers;

var localCont =response.data.contents;

//that.\_beforeEditingCache = Object.assign({}, localCont);

that.contents = localCont;

that.columns=response.data.columns;

that.copyObject=Object.assign([],Object.values(response.data.contents));

//const that.temp=response.data.contents;

});

},

setPage: function(pageNumber) {

// console.log('setPage',pageNumber);

this.currentPage = pageNumber;

},

paginate: function(list) {

//this.totalPages=Math.ceil(this.resultCount / this.itemsPerPage);

//this.totalPages=[10];

if (this.currentPage >= this.totalPages) {

this.currentPage = this.totalPages-1;

}

//if(this.totalPages>0 && this.currentPage==0)

{

//this.currentPage=1;

}

var index = this.currentPage \* this.itemsPerPage

console.log('pagginate',index,this.currentPage,this.resultCount,this.itemsPerPage);

return list.slice(index, this.itemsPerPage);

// return list.slice(index, index + this.itemsPerPage)

}//,

//totalPages(){

//console.log(Math.ceil(this.resultCount / this.itemsPerPage),'totalPages');

//return Math.ceil(this.resultCount / this.itemsPerPage)

//}

,

deleteTableRow: function (index,id) {

this.counter--;

var that=this;

this.copyObject.filter(function(record,k){

console.log('deleteTableRow',record,k);

if(record.id==id)

{

return that.$delete(that.copyObject,k);

}

});

//this.copyObject.splice(idx, 1);

},

addTableRow: function () {

this.counter++;

var editId='new\_'+ (this.copyObject.length+1);

var rec\_id=(this.recIds[this.recIds.length]+1);

this.copyObject.push({company\_name:'',city:'',turn\_over:'0',status:1,id:editId,isNew:1,rec\_id:rec\_id});

this.setValues(this.editingObj,editId,true);

console.log(this.copyObject,'addTableRow');

},

setValues(obj,index,value){

this.$set(obj,index,value);

}

,

addAbove:function(index,rowID){

console.log(index,'addAbove');

// this.$set(this.editingObj,rowID,true);

var editId='new\_'+ (this.copyObject.length+1);

var rec\_id=(this.recIds[this.recIds.length]+1);

this.copyObject.splice(index, 0, {company\_name:'',city:'',turn\_over:'0',status:1,id:editId,isNew:1,rec\_id:rec\_id});

this.$set(this.editingObj,editId,true);

// this.$set(this.editing,index,true);

console.log('addAbove',this.copyObject,this.editingObj);

//this.$nextTick(function(){

//this.$refs[index][0].focus();

//console.log('edit',this.$refs.input,this.$refs,this.editing,indx);

//});

},edit:function(id,indx){

var localColl = this.content;

this.\_beforeEditingCache = Object.assign({}, localColl);

this.setValues(this.editingObj, id, true);

//this.$set(this.editing, id, true);

// this.temp[indx]={};

//this.temp[indx]=this.copyObject[indx];

console.log('this.editing');

// this.$nextTick(function(){

//console.log(this.$refs,'rr');

//});

//}

},sortData(obj,sortKey){

sortKey=sortKey || this.currentOrder;

obj.sort((a, b) => {

if (a[sortKey] < b[sortKey]) return -1

if (a[sortKey] > b[sortKey]) return 1

return 0

});

return obj;

}

,

cancelEv(task)

{

this.\_beforeEditingCache = Object.assign({}, task);

this.editing = task;

},

editTask: function(task) {

// Keep track of the original task information

this.\_beforeEditingCache = task;

// Put the task into edit mode (also changes the interface)

this.editing = task;

},

resetCheck(index,id){

console.log('resetCheck',this.contents,this.editingObj,this.temp);

that=this;

this.copyObject.filter(function(record,k){

console.log(record.id,id,index,k,'yyyy')

if(that.editingObj && that.editingObj[record.id] && record.id && record.id==id && !record.isNew)

{

if(that.validateobj[k]){

that.$delete(that.validateobj,k);

} console.log('ttgg ',that.copyObject,k,that.originalobj["\_"+record.id])

that.setValues(that.copyObject,k,that.originalobj["\_"+record.id]);

that.$delete(that.editingObj,id);

return true;

}else{

if(record.isNew==1 && record.id==id && index==k && that.editingObj[record.id]){

that.$delete(that.copyObject,index);

}

console.log(that.copyObject,'resetCheck')

//that.$set(that.copyObject[index],'edit',false);

return true;

}

});

}

,

mouseDown(ind,id){

console.log('mouseDown',ind,this.dragStyleObj);

this.$set(this.dragStyleObj,ind,{

cursor:'move'

});

this.downIndex=ind;

this.downID=id;

}

,

mouseOver(ind,id)

{

if(this.downIndex!=-2 && this.downIndex!=ind )

{

this.$set(this.dragStyleObj,ind,{

outline:'1px solid #f00',

});

}

}

,

mouseOut(ind,id){

if(this.downIndex!=-2 && this.downIndex!=ind )

{

this.$set(this.dragStyleObj,ind,{

outline:'1px solid #000',

});

}

},

mouseUp(ind,id){

console.log(this.downIndex,id,ind,this.downID,'mousup');

if(this.downIndex!=ind && this.downIndex!=-2)

{

var fromId= this.downID,ToID=id,fromOrder=null,ToOrder=null;

// this.copyObject.filter(function(record,k){

// if(that.copyObject[k].id==fromId)

// {

// fromOrder=parseInt(that.copyObject[k].rec\_id);

// that.downIndex=k;

// cnt++;

// }

// if(that.copyObject[k].id==ToID)

// {

// ToOrder=parseInt(that.copyObject[k].rec\_id);

// ind=k;

// cnt++;

// }

// if(fromOrder!=null)

// {

// that.$set(that.copyObject[ind],'rec\_id', (fromOrder));

// // this.$set(this.originalobj[ind],'rec\_id',fromOrder);

// }

// if(ToOrder!=null)

// {

// that.$set(that.copyObject[that.downIndex],'rec\_id',(ToOrder));

// //this.$set(this.originalobj[this.downIndex],'rec\_id',ToOrder);

// }

// console.log(cnt,'cnt',that.downIndex,ind,fromOrder,ToOrder);

// });

if(this.copyObject[this.downIndex].id==fromId){

var fromOrder=(this.copyObject[this.downIndex]);

}

if(this.copyObject[ind].id==ToID){

ToOrder=(this.copyObject[ind]);

}

if(fromOrder!=null){

this.$set(this.copyObject,ind, (fromOrder));

// // this.$set(this.originalobj[ind],'rec\_id',fromOrder);

}

if(ToOrder!=null){

this.$set(this.copyObject,this.downIndex,(ToOrder));

//this.$set(this.originalobj[this.downIndex],'rec\_id',ToOrder);

}

console.log('mouseUp',ind,this.downIndex+' this.downIndex',(fromOrder),ToOrder,this.copyObject);

this.dragStyleObj={};

this.downIndex=-2;

this.downID=null;

}

}

,reselAll:function(){

console.log('reselAll');

var that=this;

this.copyObject.filter(function(record,k){

if(that.editingObj[record.id] && that.copyObject[k].id==record.id)

{

// this.$delete( this.validateobj,ii);

that.$delete( that.editingObj,record.id);

if(record.isNew==1)

{

that.$delete( that.copyObject,k);

}else{

Object.assign(that.copyObject[k],that.$options.data().contents["\_"+record.id]);

}

}

});

}

,

addFilter(colval,ev)

{console.log(ev,'$event',ev.target.value);

this.resetfields();

if(colval.filtertype==2 && ev.target.value==0)

{

this.isFilter=false;

}else

{

this.activeFilters[colval.filterid]=ev.target.value;

this.isFilter=(this.activeFilters[colval.filterid])?true:false;

this.$set(this.columns[colval.filterid],'filterVal',ev.target.value);

}

console.log('addFilter',this.activeFilters,this.isFilter,this.columns)

}

,

editTodo: function(index,col,ev) {

console.log('editTodo',ev.target,this.copyObject);

var str=index+'\_'+col;

if(this.columns[col] && this.columns[col].validate)

{ console.log('in666',col,ev.target.value,this.columns[col].validate.includes(ev.target.value))

if(this.columns[col].validate.includes(ev.target.value))

{

if(!this.validateobj[index])

{

this.validateobj[index]=[];

}

this.validateobj[index].push(col);

}else{

console.log(this.validateobj,'before');

if(this.validateobj && this.validateobj[index] && this.validateobj[index].includes(col)==true)

{

// delete this.validateobj[index][str];

var delIndex=this.validateobj[index].indexOf(col);

this.validateobj[index].splice(delIndex, 1);

}

if(this.validateobj!=null && this.validateobj[index] && this.validateobj[index].length==0)

{

this.validateobj.splice(index, 1);

}

}

}

console.log(this.validateobj,'after');

}

,

save : function(obj){

var localData={},expData=[];

for(var i in this.$refs)

{

localData={};

for(var ii in this.$refs[i])

{

var expVal=this.$refs[i][ii].id.split('\_');

localData[expVal[1]]=this.$refs[i][ii].value;

console.log(this.$refs,this.$refs[i][ii].value,'yyy',this.$refs[i][ii].id,localData);

}

expData.push(localData);

}

console.log(expData,'save');

}

,resetfields(){

this.validateobj={};

var that=this;

this.copyObject.filter(function(record,k){

if(that.editingObj && that.editingObj[record.id])

{

if(record.isNew==1){

that.$delete(that.copyObject,k);

}else{

that.$delete(that.editingObj,record.id);

}

}

});

//for(var i in this.$refs)

{

console.log('resetfields');

}

}

}

});

</script>

</body>

</html>