流程设计源代码

<?xml version="1.0" encoding="UTF-8"?>

<projectDescription>

<name>src</name>

<comment></comment>

<projects>

</projects>

<buildSpec> <name>com.aptana.ide.core.unifiedBuilder</name> <arguments>

</arguments>

</buildCommand>

</buildSpec>

<natures>

<nature>com.aptana.projects.webnature</nature>

</natures>

</projectDescription>

< template >

<

div style = "height: 100%;" >

<

a - layout class = "container" >

<

a - layout - sider

width = "300"

theme = "light"

class = "select-area" >

<

a - row >

<

a - checkable - tag v - model = "tag.checked0"

@change = "toggleNodeShow0"

class = "tag" > 工具 < /a-checkable-tag> <

div align = "center" >

<

a - list: grid = "{ gutter: 8, column: 1 }"

v -

if = "tag.toolShow" >

<

a - list - item >

<

a - button - group >

<

a - button

v -

for = "(tool, index) in field.tools": key = "index": icon = "tool.icon": type =

"currentTool.type == tool.type ? 'primary': 'default'"

@click = "selectTool(tool.type)" >

<

/a-button> <

/a-button-group> <

/a-list-item> <

/a-list> <

/div> <

/a-row> <

a - row >

<

a - checkable - tag v - model = "tag.checked1"

@change = "toggleNodeShow1"

class = "tag" > 基础节点 < /a-checkable-tag> <

div align = "center" >

<

a - list: grid = "{ gutter: 8, column: 2 }"

v -

if = "tag.commonNodeShow" >

<

a - list - item v -

for = "(commonNode, index) in field.commonNodes": key = "index" >

<

div class = "node-item": type = "commonNode.type"

belongto = "commonNodes" >

<

a - icon: component = "commonNode.icon" / > {

{

commonNode.nodeName

}

} <

/div> <

/a-list-item> <

/a-list> <

/div> <

/a-row> <

a - row >

<

a - checkable - tag v - model = "tag.checked2"

@change = "toggleNodeShow2"

class = "tag" > 高级节点 < /a-checkable-tag> <

div align = "center" >

<

a - list: grid = "{ gutter: 8, column: 2 }"

v -

if = "tag.highNodeShow" >

<

a - list - item v -

for = "(highNode, index) in field.highNodes": key = "index" >

<

div class = "node-item": type = "highNode.type"

belongto = "highNodes" >

<

a - icon: component = "highNode.icon" / > {

{

highNode.nodeName

}

} <

/div> <

/a-list-item> <

/a-list> <

/div> <

/a-row> <

a - row >

<

a - checkable - tag v - model = "tag.checked3"

@change = "toggleNodeShow3"

class = "tag" > 泳道节点 < /a-checkable-tag> <

div align = "center" >

<

a - list: grid = "{ gutter: 8, column: 2 }"

v -

if = "tag.laneNodeShow" >

<

a - list - item v -

for = "(laneNode, index) in field.laneNodes": key = "index" >

<

div class = "node-item": type = "laneNode.type"

belongto = "laneNodes" >

<

a - icon: component = "laneNode.icon" / > {

{

laneNode.nodeName

}

} <

/div> <

/a-list-item> <

/a-list> <

/div> <

/a-row> <

/a-layout-sider> <

a - layout >

<

a - layout - header class = "header-option" >

<

a - tooltip title = "生成流程图片"

placement = "bottom" >

<

a - button @click = "exportFlowPicture"

class = "header-option-button"

size = "small"

icon = "picture" > < /a-button> <

/a-tooltip> <

a - popconfirm title = "确认要重新绘制吗？"

placement = "bottom"

okText = "确认"

cancelText = "取消"

@confirm = "clear" >

<

a - tooltip title = "重新绘制"

placement = "bottom" >

<

a - button class = "header-option-button"

size = "small"

icon = "delete" > < /a-button> <

/a-tooltip> <

/a-popconfirm> <

a - tooltip: title = "flowData.config.showGridText"

placement = "bottom" >

<

a - button

@click = "toggleShowGrid"

class = "header-option-button"

size = "small": icon = "flowData.config.showGridIcon" >

<

/a-button> <

/a-tooltip> <

a - tooltip title = "设置"

placement = "bottom" >

<

a - button @click = "setting"

class = "header-option-button"

size = "small"

icon = "setting" > < /a-button> <

/a-tooltip> <

a - popconfirm

title = "请选择帮助项："

placement = "bottom"

okType = "default"

okText = "快捷键大全"

cancelText = "使用文档"

@confirm = "shortcutHelper"

@cancel = "usingDoc" >

<

a - icon slot = "icon"

type = "question-circle-o"

style = "color: red" / >

<

/a-popconfirm> <

a - tooltip title = "退出"

placement = "bottom" >

<

a - button @click = "exit"

class = "header-option-button"

size = "small"

icon = "logout" > < /a-button> <

/a-tooltip> <

/a-layout-header> <

a - layout - content class = "content" >

<

flow - area

ref = "flowArea": browserType = "browserType": flowData = "flowData": select.sync = "currentSelect": selectGroup.sync =

"currentSelectGroup": plumb = "plumb": currentTool = "currentTool"

@findNodeConfig = "findNodeConfig"

@selectTool = "selectTool"

@getShortcut = "getShortcut"

@saveFlow = "saveFlow" >

<

/flow-area> <

vue - context - menu: contextMenuData = "linkContextMenuData"

@deleteLink = "deleteLink" >

<

/vue-context-menu> <

/a-layout-content> <

a - layout - footer class = "foot" >

<

span > < /span> <

/a-layout-footer> <

/a-layout> <

a - layout - sider

width = "350"

theme = "light"

class = "attr-area"

@mousedown.stop = "loseShortcut" >

<

flow - attr: plumb = "plumb": flowData = "flowData": select.sync = "currentSelect" > < /flow-attr> <

/a-layout-sider> <

/a-layout> <

a - modal: title = "'流程设计图\_' + flowData.attr.id + '.png'"

centered

width = "90%": closable = "flowPicture.closable": maskClosable = "flowPicture.maskClosable": visible =

"flowPicture.modalVisible"

okText = "下载到本地"

cancelText = "取消"

@ok = "downLoadFlowPicture"

@cancel = "cancelDownLoadFlowPicture" >

<

div align = "center" >

<

img: src = "flowPicture.url" / >

<

/div> <

/a-modal> <

setting - modal ref = "settingModal" > < /setting-modal> <

shortcut - modal ref = "shortcutModal" > < /shortcut-modal> <

test - modal ref = "testModal"

@loadFlow = "loadFlow" > < /test-modal> <

/div> <

/template>

<

script >

import jsplumb from 'jsplumb'

import {

tools,

commonNodes,

highNodes,

laneNodes

} from './config/basic-node-config.js'

import {

flowConfig

} from './config/args-config.js'

import {

startSvg,

endSvg,

commonSvg,

freedomSvg,

gatewaySvg,

eventSvg,

childFlowSvg,

xLaneSvg,

yLaneSvg,

lanePoolSvg

} from './config/basic-icon-config.js'

import $ from 'jquery'

import 'jquery-ui/ui/widgets/draggable'

import 'jquery-ui/ui/widgets/droppable'

import 'jquery-ui/ui/widgets/resizable'

import html2canvas from 'html2canvas'

import canvg from 'canvg'

import {

ZFSN

} from './util/ZFSN.js'

import FlowArea from './modules/FlowArea'

import FlowAttr from './modules/FlowAttr'

import SettingModal from './modules/SettingModal'

import ShortcutModal from './modules/ShortcutModal'

import UsingDocModal from './modules/UsingDocModal'

import TestModal from './modules/TestModal'

export default {

name: 'vfd',

components: {

jsplumb,

flowConfig,

html2canvas,

canvg,

FlowArea,

FlowAttr,

SettingModal,

ShortcutModal,

UsingDocModal,

TestModal,

StartIcon: {

template: startSvg

},

EndIcon: {

template: endSvg

},

CommonIcon: {

template: commonSvg

},

FreedomIcon: {

template: freedomSvg

},

GatewayIcon: {

template: gatewaySvg

},

EventIcon: {

template: eventSvg

},

ChildFlowIcon: {

template: childFlowSvg

},

XLaneIcon: {

template: xLaneSvg

},

YLaneIcon: {

template: yLaneSvg

},

LanePoolIcon: {

template: lanePoolSvg

}

},

mounted() {

const that = this;

that.dealCompatibility();

that.initNodeSelectArea();

that.initJsPlumb();

that.listenShortcut();

that.initFlow();

that.listenPage();

setTimeout(function() {

ZFSN.consoleLog([

'欢迎使用VFD流程设计器!',

'当前版本：' + that.info.version,

'Powered by ' + that.info.author,

]);

that.$notification.open({

placement: 'bottomRight',

message: '欢迎使用VFD流程设计器!',

description: < p >

当前版本： 1.0 .5 < br / >

Powered by ZSFN < br / >

软件： VFD流程设计器 <

/p>,

icon: < a - icon type = "smile"

style = "color: #108ee9" / >

});

}, 1000);

},

data() {

return {

info: {

version: '1.0.5',

author: 'ZSFN',

gitee: 'https://gitee.com/yjblogs/VFD'

},

tag: {

checked0: true,

checked1: true,

checked2: true,

checked3: true,

toolShow: true,

commonNodeShow: true,

highNodeShow: true,

laneNodeShow: true

},

browserType: 3,

plumb: {},

field: {

tools: tools,

commonNodes: commonNodes,

highNodes: highNodes,

laneNodes: laneNodes

},

flowData: {

nodeList: [],

linkList: [],

attr: {

id: ''

},

config: {

showGrid: true,

showGridText: '隐藏网格',

showGridIcon: 'eye'

},

status: flowConfig.flowStatus.CREATE,

remarks: []

},

currentTool: {

type: 'drag',

icon: 'drag',

name: '拖拽'

},

currentSelect: {},

currentSelectGroup: [],

activeShortcut: true,

linkContextMenuData: flowConfig.contextMenu.link,

flowPicture: {

url: '',

modalVisible: false,

closable: false,

maskClosable: false

}

}

},

methods: {

toggleNodeShow0(flag) {

if (!flag) {

this.tag.toolShow = false;

} else {

this.tag.toolShow = true;

}

},

toggleNodeShow1(flag) {

if (!flag) {

this.tag.commonNodeShow = false;

} else {

this.tag.commonNodeShow = true;

}

},

toggleNodeShow2(flag) {

if (!flag) {

this.tag.highNodeShow = false;

} else {

this.tag.highNodeShow = true;

}

},

toggleNodeShow3(flag) {

if (!flag) {

this.tag.laneNodeShow = false;

} else {

this.tag.laneNodeShow = true;

}

},

getBrowserType() {

let userAgent = navigator.userAgent;

let isOpera = userAgent.indexOf("Opera") > -1;

if (isOpera) {

return 1;

};

if (userAgent.indexOf("Firefox") > -1) {

return 2;

}

if (userAgent.indexOf("Chrome") > -1) {

return 3;

}

if (userAgent.indexOf("Safari") > -1) {

return 4;

}

if (userAgent.indexOf("compatible") > -1 && userAgent.indexOf("MSIE") > -1 && !isOpera) {

alert('IE浏览器支持性较差，推荐使用Firefox或Chrome');

return 5;

}

if (userAgent.indexOf("Trident") > -1) {

alert('Edge浏览器支持性较差，推荐使用Firefox或Chrome');

return 6;

};

},

dealCompatibility() {

const that = this;

that.browserType = that.getBrowserType();

if (that.browserType == 2) {

flowConfig.shortcut.scaleContainer = {

code: 16,

codeName: 'SHIFT(chrome下为ALT)',

shortcutName: '画布缩放',

};

}

},

initJsPlumb() {

const that = this;

that.plumb = jsPlumb.getInstance(flowConfig.jsPlumbInsConfig);

that.plumb.bind('beforeDrop', function(info) {

let sourceId = info.sourceId;

let targetId = info.targetId;

if (sourceId == targetId) return false;

let filter = that.flowData.linkList.filter(link => (link.sourceId == sourceId && link.targetId ==

targetId));

if (filter.length > 0) {

that.$message.error('同方向的两节点连线只能有一条！');

return false;

}

return true;

});

that.plumb.bind('connection', function(conn, e) {

let connObj = conn.connection.canvas;

let o = {},

id, label;

if (that.flowData.status == flowConfig.flowStatus.CREATE || that.flowData.status == flowConfig.flowStatus

.MODIFY) {

id = 'link-' + ZFSN.getId();

label = '';

} else if (that.flowData.status == flowConfig.flowStatus.LOADING) {

let l = that.flowData.linkList[that.flowData.linkList.length - 1];

id = l.id;

label = l.label;

}

connObj.id = id;

o.type = 'link';

o.id = id;

o.sourceId = conn.sourceId;

o.targetId = conn.targetId;

o.label = label;

o.cls = {

linkType: flowConfig.jsPlumbInsConfig.Connector[0],

linkColor: flowConfig.jsPlumbInsConfig.PaintStyle.stroke,

linkThickness: flowConfig.jsPlumbInsConfig.PaintStyle.strokeWidth

};

$('#' + id).bind('contextmenu', function(e) {

that.showLinkContextMenu(e);

that.currentSelect = that.flowData.linkList.filter(l => l.id == id)[0];

});

$('#' + id).bind('click', function(e) {

let event = window.event || e;

event.stopPropagation();

that.currentSelect = that.flowData.linkList.filter(l => l.id == id)[0];

});

if (that.flowData.status != flowConfig.flowStatus.LOADING) that.flowData.linkList.push(o);

});

that.plumb.importDefaults({

ConnectionsDetachable: flowConfig.jsPlumbConfig.conn.isDetachable

});

ZFSN.consoleLog(['实例化JsPlumb成功...']);

},

initNodeSelectArea() {

$(document).ready(function() {

$(".node-item").draggable({

opacity: flowConfig.defaultStyle.dragOpacity,

helper: 'clone',

cursorAt: {

top: 16,

left: 60

},

containment: "window",

revert: "invalid"

});

ZFSN.consoleLog(['初始化节点选择列表成功...']);

});

},

listenShortcut() {

const that = this;

document.onkeydown = function(e) {

let event = window.event || e;

if (!that.activeShortcut) return;

let key = event.keyCode;

switch (key) {

case flowConfig.shortcut.multiple.code:

that.$refs.flowArea.rectangleMultiple.flag = true;

break;

case flowConfig.shortcut.dragContainer.code:

that.$refs.flowArea.container.dragFlag = true;

break;

case flowConfig.shortcut.scaleContainer.code:

that.$refs.flowArea.container.scaleFlag = true;

break;

case flowConfig.shortcut.dragTool.code:

that.selectTool('drag');

break;

case flowConfig.shortcut.connTool.code:

that.selectTool('connection');

break;

case flowConfig.shortcut.zoomInTool.code:

that.selectTool('zoom-in');

break;

case flowConfig.shortcut.zoomOutTool.code:

that.selectTool('zoom-out');

break;

case 37:

that.moveNode('left');

break;

case 38:

that.moveNode('up');

break;

case 39:

that.moveNode('right');

break;

case 40:

that.moveNode('down');

break;

}

if (event.ctrlKey) {

if (event.altKey) {

switch (key) {

case flowConfig.shortcut.settingModal.code:

that.setting();

break;

case flowConfig.shortcut.testModal.code:

that.openTest();

break;

}

}

}

}

document.onkeyup = function(e) {

let event = window.event || e;

let key = event.keyCode;

if (key == flowConfig.shortcut.dragContainer.code) {

that.$refs.flowArea.container.dragFlag = false;

} else if (key == flowConfig.shortcut.scaleContainer.code) {

event.preventDefault();

that.$refs.flowArea.container.scaleFlag = false;

} else if (key == flowConfig.shortcut.multiple.code) {

that.$refs.flowArea.rectangleMultiple.flag = false;

}

}

ZFSN.consoleLog(['初始化快捷键成功...']);

},

listenPage() {

window.onbeforeunload = function(e) {

e = e || window.event;

if (e) {

e.returnValue = '关闭提示';

}

return '关闭提示';

};

},

initFlow() {

const that = this;

if (that.flowData.status == flowConfig.flowStatus.CREATE) {

that.flowData.attr.id = 'flow-' + ZFSN.getId();

} else {

that.loadFlow();

}

ZFSN.consoleLog(['初始化流程图成功...']);

},

loadFlow(json) {

const that = this;

that.clear();

let loadData = JSON.parse(json);

that.flowData.attr = loadData.attr;

that.flowData.config = loadData.config;

that.flowData.status = flowConfig.flowStatus.LOADING;

that.plumb.batch(function() {

let nodeList = loadData.nodeList;

nodeList.forEach(function(node, index) {

that.flowData.nodeList.push(node);

});

let linkList = loadData.linkList;

that.$nextTick(() => {

linkList.forEach(function(link, index) {

that.flowData.linkList.push(link);

let conn = that.plumb.connect({

source: link.sourceId,

target: link.targetId,

anchor: flowConfig.jsPlumbConfig.anchor.default,

connector: [

link.cls.linkType,

{

gap: 5,

cornerRadius: 8,

alwaysRespectStubs: true

}

],

paintStyle: {

stroke: link.cls.linkColor,

strokeWidth: link.cls.linkThickness

}

});

if (link.label != '') {

conn.setLabel({

label: link.label,

cssClass: 'linkLabel'

});

}

});

that.currentSelect = {};

that.currentSelectGroup = [];

that.flowData.status = flowConfig.flowStatus.MODIFY;

});

}, true);

let canvasSize = that.computeCanvasSize();

that.$refs.flowArea.container.pos = {

top: -canvasSize.minY + 100,

left: -canvasSize.minX + 100

}

},

findNodeConfig(belongto, type, callback) {

let node = null;

switch (belongto) {

case 'commonNodes':

node = commonNodes.filter(n => n.type == type);

break;

case 'highNodes':

node = highNodes.filter(n => n.type == type);

break;

case 'laneNodes':

node = laneNodes.filter(n => n.type == type);

break;

}

if (node && node.length >= 0) node = node[0];

callback(node);

},

selectTool(type) {

let tool = tools.filter(t => t.type == type);

if (tool && tool.length >= 0) this.currentTool = tool[0];

switch (type) {

case 'drag':

this.changeToDrag();

break;

case 'connection':

this.changeToConnection();

break;

case 'zoom-in':

this.changeToZoomIn();

break;

case 'zoom-out':

this.changeToZoomOut();

break;

}

},

changeToDrag() {

const that = this;

that.flowData.nodeList.forEach(function(node, index) {

let f = that.plumb.toggleDraggable(node.id);

if (!f) {

that.plumb.toggleDraggable(node.id);

}

if (node.type != 'x-lane' && node.type != 'y-lane') {

that.plumb.unmakeSource(node.id);

that.plumb.unmakeTarget(node.id);

}

});

},

changeToConnection() {

const that = this;

that.flowData.nodeList.forEach(function(node, index) {

let f = that.plumb.toggleDraggable(node.id);

if (f) {

that.plumb.toggleDraggable(node.id);

}

if (node.type != 'x-lane' && node.type != 'y-lane') {

that.plumb.makeSource(node.id, flowConfig.jsPlumbConfig.makeSourceConfig);

that.plumb.makeTarget(node.id, flowConfig.jsPlumbConfig.makeTargetConfig);

}

});

that.currentSelect = {};

that.currentSelectGroup = [];

},

changeToZoomIn() {

console.log('切换到放大工具');

},

changeToZoomOut() {

console.log('切换到缩小工具');

},

checkFlow() {

const that = this;

let nodeList = that.flowData.nodeList;

if (nodeList.length <= 0) {

this.$message.error('流程图中无任何节点！');

return false;

}

return true;

},

saveFlow() {

const that = this;

let flowObj = Object.assign({}, that.flowData);

if (!that.checkFlow()) return;

flowObj.status = flowConfig.flowStatus.SAVE;

let d = JSON.stringify(flowObj);

console.log(d);

this.$message.success('保存流程成功！请查看控制台。');

return d;

},

exportFlowPicture() {

const that = this;

if (!that.checkFlow()) return;

let $Container = that.$refs.flowArea.$el.children[0],

svgElems = $($Container).find('svg[id^="link-"]'),

removeArr = [];

svgElems.each(function(index, svgElem) {

let linkCanvas = document.createElement('canvas');

let canvasId = 'linkCanvas-' + ZFSN.getId();

linkCanvas.id = canvasId;

removeArr.push(canvasId);

let svgContent = svgElem.outerHTML.trim();

canvg(linkCanvas, svgContent);

if (svgElem.style.position) {

linkCanvas.style.position += svgElem.style.position;

linkCanvas.style.left += svgElem.style.left;

linkCanvas.style.top += svgElem.style.top;

}

$($Container).append(linkCanvas);

});

let canvasSize = that.computeCanvasSize();

let pbd = flowConfig.defaultStyle.photoBlankDistance;

let offsetPbd = ZFSN.div(pbd, 2);

html2canvas($Container, {

width: canvasSize.width + pbd,

height: canvasSize.height + pbd,

scrollX: -canvasSize.minX + offsetPbd,

scrollY: -canvasSize.minY + offsetPbd,

logging: false,

onclone: function(args) {

removeArr.forEach(function(id, index) {

$('#' + id).remove();

});

}

}).then(canvas => {

let dataURL = canvas.toDataURL('image/png');

that.flowPicture.url = dataURL;

that.flowPicture.modalVisible = true;

});

},

downLoadFlowPicture() {

const that = this;

let alink = document.createElement('a');

let alinkId = 'alink-' + ZFSN.getId();

alink.id = alinkId;

alink.href = that.flowPicture.url;

alink.download = '流程设计图\_' + that.flowData.attr.id + '.png';

alink.click();

},

cancelDownLoadFlowPicture() {

this.flowPicture.url = '';

this.flowPicture.modalVisible = false;

},

computeCanvasSize() {

const that = this;

let nodeList = Object.assign([], that.flowData.nodeList),

minX = nodeList[0].x,

minY = nodeList[0].y,

maxX = nodeList[0].x + nodeList[0].width,

maxY = nodeList[0].y + nodeList[0].height;

nodeList.forEach(function(node, index) {

minX = Math.min(minX, node.x);

minY = Math.min(minY, node.y);

maxX = Math.max(maxX, node.x + node.width);

maxY = Math.max(maxY, node.y + node.height);

});

let canvasWidth = maxX - minX;

let canvasHeight = maxY - minY;

return {

width: canvasWidth,

height: canvasHeight,

minX: minX,

minY: minY,

maxX: maxX,

maxY: maxY

};

},

clear() {

const that = this;

that.flowData.nodeList.forEach(function(node, index) {

that.plumb.remove(node.id);

});

that.currentSelect = {};

that.currentSelectGroup = [];

that.flowData.nodeList = [];

that.flowData.linkList = [];

that.flowData.remarks = [];

},

toggleShowGrid() {

let flag = this.flowData.config.showGrid;

if (flag) {

this.flowData.config.showGrid = false;

this.flowData.config.showGridText = '显示网格';

this.flowData.config.showGridIcon = 'eye-invisible';

} else {

this.flowData.config.showGrid = true;

this.flowData.config.showGridText = '隐藏网格';

this.flowData.config.showGridIcon = 'eye';

}

},

setting() {

this.$refs.settingModal.open();

},

shortcutHelper() {

this.$refs.shortcutModal.open();

},

usingDoc() {

window.open(this.info.gitee);

},

exit() {

alert('退出流程设计器...');

},

showLinkContextMenu(e) {

let event = window.event || e;

event.preventDefault();

event.stopPropagation();

$('.vue-contextmenuName-flow-menu').css('display', 'none');

$('.vue-contextmenuName-node-menu').css('display', 'none');

let x = event.clientX;

let y = event.clientY;

this.linkContextMenuData.axis = {

x,

y

};

},

deleteLink() {

const that = this;

let sourceId = that.currentSelect.sourceId;

let targetId = that.currentSelect.targetId;

that.plumb.deleteConnection(that.plumb.getConnections({

source: sourceId,

target: targetId

})[0]);

let linkList = that.flowData.linkList;

linkList.splice(linkList.findIndex(link => (link.sourceId == sourceId || link.targetId == targetId)), 1);

that.currentSelect = {};

},

loseShortcut() {

this.activeShortcut = false;

},

getShortcut() {

this.activeShortcut = true;

},

openTest() {

const that = this;

let flowObj = Object.assign({}, that.flowData);

that.$refs.testModal.flowData = flowObj;

that.$refs.testModal.testVisible = true;

},

moveNode(type) {

const that = this;

let m = flowConfig.defaultStyle.movePx,

isX = true;

switch (type) {

case 'left':

m = -m;

break;

case 'up':

m = -m;

isX = false;

break;

case 'right':

break;

case 'down':

isX = false;

}

if (that.currentSelectGroup.length > 0) {

that.currentSelectGroup.forEach(function(node, index) {

if (isX) {

node.x += m;

} else {

node.y += m;

}

});

that.plumb.repaintEverything();

} else if (that.currentSelect.id) {

if (isX) {

that.currentSelect.x += m;

} else {

that.currentSelect.y += m;

}

that.plumb.repaintEverything();

}

}

}

} <

/script>

<

style lang = "scss" >

@import './style/flow-designer.scss' <

/style>

import {

flowConfig

} from '../config/args-config.js'

export let ZFSN = {

seqNo: 1,

consoleLog: function(strArr) {

let log = '';

for (let i = 0, len = strArr.length; i < len; i++) {

log += strArr[i] + '\n';

}

},

getId: function() {

let idType = flowConfig.idType;

if (typeof idType == 'string') {

if (idType == 'uuid') {

return this.getUUID();

} else if (idType == 'time\_stamp') {

return this.getTimeStamp();

}

} else if (idType instanceof Array) {

if (idType[0] == 'time\_stamp\_and\_sequence') {

return this.getSequence(idType[1]);

} else if (idType[0] == 'time\_stamp\_and\_sequence') {

return this.getTimeStampAndSequence(idType[1]);

} else if (idType[0] == 'custom') {

return idType[1]();

}

}

},

getUUID: function() {

let s = [];

let hexDigits = "0123456789abcdef";

for (let i = 0; i < 36; i++) {

s[i] = hexDigits.substr(Math.floor(Math.random() \* 0x10), 1);

}

s[14] = "4";

s[19] = hexDigits.substr((s[19] & 0x3) | 0x8, 1);

s[8] = s[13] = s[18] = s[23] = "-";

let uuid = s.join("");

return uuid.replace(/-/g, '');

},

getTimeStamp: function() {

return new Date().getTime();

},

getSequence: function(seqNoLength) {

let zeroStr = new Array(seqNoLength).fill('0').join('');

return (zeroStr + (this.seqNo++)).slice(-seqNoLength);

},

getTimeStampAndSequence: function(seqNoLength) {

return this.getTimeStamp() + this.getSequence(seqNoLength);

},

add: function(a, b) {

let c, d, e;

try {

c = a.toString().split(".")[1].length;

} catch (f) {

c = 0;

}

try {

d = b.toString().split(".")[1].length;

} catch (f) {

d = 0;

}

return e = Math.pow(10, Math.max(c, d)), (this.mul(a, e) + this.mul(b, e)) / e;

},

sub: function(a, b) {

let c, d, e;

try {

c = a.toString().split(".")[1].length;

} catch (f) {

c = 0;

}

try {

d = b.toString().split(".")[1].length;

} catch (f) {

d = 0;

}

return e = Math.pow(10, Math.max(c, d)), (this.mul(a, e) - this.mul(b, e)) / e;

},

mul: function(a, b) {

let c = 0,

d = a.toString(),

e = b.toString();

try {

c += d.split(".")[1].length;

} catch (f) {}

try {

c += e.split(".")[1].length;

} catch (f) {}

return Number(d.replace(".", "")) \* Number(e.replace(".", "")) / Math.pow(10, c);

},

div: function(a, b) {

let c, d, e = 0,

f = 0;

try {

e = a.toString().split(".")[1].length;

} catch (g) {}

try {

f = b.toString().split(".")[1].length;

} catch (g) {}

return c = Number(a.toString().replace(".", "")), d = Number(b.toString().replace(".", "")), this.mul(c /

d, Math.pow(10, f - e));

}

};

$active-color: #409EFF;

.btn-wrapper-simple {

height: 24px !important;

line-height: 24px !important;

}

.vue-contextmenu-listWrapper {

padding-left: 1px !important;

}

.child-ul-wrapper {

padding-left: 1px !important;

}

.flow-container {

width: 1000%;

height: 1000%;

position: relative;

transition: transform 0.5s ease 0s,transform-origin 0.5s ease 0s;

&.grid {

background-image: -webkit-linear-gradient(90deg, rgba(235, 235, 235, 1) 5%, rgba(0, 0, 0, 0) 5%);

background-image: -moz-linear-gradient(90deg, rgba(235, 235, 235, 1) 5%, rgba(0, 0, 0, 0) 5%);

background-image: -o-linear-gradient(90deg, rgba(235, 235, 235, 1) 5%, rgba(0, 0, 0, 0) 5%);

background-image: -webkit-gradient(linear, 0 100%, 0 0, color-stop(0.05, rgba(235, 235, 235, 1)), color-stop(0.05, rgba(0, 0, 0, 0)));

background-image: linear-gradient(90deg, rgba(235, 235, 235, 1) 5%, rgba(0, 0, 0, 0) 5%),linear-gradient(rgba(235, 235, 235, 1) 5%, rgba(0, 0, 0, 0) 5%);

background-size: 1rem 1rem;

}

&.zoomIn {

cursor: zoom-in;

}

&.zoomOut {

cursor: zoom-out;

}

&.canScale {

cursor: url(../assets/search.png), default;

}

&.canDrag {

cursor: grab;

}

&.canMultiple {

cursor: url(../assets/multip-pointer.png), default;

}

}

.rectangle-multiple {

position: absolute;

border: 1px dashed #31676f;

background-color: #0cceea29;

}

.flow-container-active {

background-color: #e4e4e438;

cursor: crosshair;

}

.auxiliary-line-x {

position: absolute;

border: 0.5px solid $active-color;

width: 100%;

z-index: 9999;

}

.auxiliary-line-y {

position: absolute;

border: 0.5px solid $active-color;

height: 100%;

z-index: 9999;

}

.link-active {

outline: 2px dashed $active-color;

}

.container-scale {

position: absolute;

top: 0;

left: 5px;

}

.mouse-position {

position: absolute;

bottom: 0;

right: 5px;

}

.common-remarks {

width: 100px;

height: 150px;

position: absolute;

background-color: #ffffaa;

}

$primary-color: #409EFF;

.container {

border: 2px solid #e4e7ed;

height: 100%;

moz-user-select: -moz-none;

-moz-user-select: none;

-o-user-select:none;

-khtml-user-select:none;

-webkit-user-select:none;

-ms-user-select:none;

user-select:none;

}

.select-area {

border-right: 1px solid #e4e7ed;

}

.header-option {

background: white;

height: 36px;

line-height: 36px;

border-bottom: 2px solid #e4e7ed;

text-align: right;

}

.header-option-button {

border: 0;

margin-left: 8px;

}

.content {

background: #fafafa;

height: 100%;

border: 2px dashed rgba(170,170,170,0.7);

}

.ant-layout-footer {

padding: 4px 8px;

}

.foot {

height: auto;

text-align: center;

}

.attr-area {

border-left: 1px solid #e4e7ed;

}

.tag {

margin: 6px;

}

.tool-item {

background: #f4f6fc;

height: 32px;

line-height: 32px;

margin: 5px;

padding-left: 8px;

text-align: center;

cursor: pointer;

&:hover{

background: #d2d3d6;

}

&.active {

background: black;

}

}

.node-item {

background: #f4f6fc;

height: 32px;

line-height: 32px;

margin: 5px;

padding-left: 8px;

text-align: left;

cursor: move;

min-width: 120px;

&:hover{

color: $primary-color;

outline: 1px dashed $primary-color;

}

}

.ant-list-grid .ant-list-item {

margin-bottom: 8px;

}

.linkLabel {

background-color: white;

padding: 1px;

border: 1px solid #346789;

border-radius: 5px;

opacity: 0.8;

z-index: 3;

}

$common-node-bg: #f4f6fc;

$common-node-bg-hover: #f4f6fcb0;

$common-node-active: #409EFF;

.common-circle-node {

position: absolute;

height: 50px;

width: 50px;

line-height: 50px;

text-align: center;

border: 1px solid #777;

border-radius: 50%;

background-color: $common-node-bg;

white-space: nowrap;

&:hover {

background-color: $common-node-bg-hover;

z-index: 2;

}

&.active {

outline: 2px dashed $common-node-active;

outline-offset: 0px;

}

}

.common-rectangle-node {

position: absolute;

height: 50px;

width: 120px;

line-height: 50px;

text-align: center;

border: 1px solid #777;

border-radius: 5px;

background-color: $common-node-bg;

white-space: nowrap;

&:hover {

background-color: $common-node-bg-hover;

z-index: 2;

}

&.active {

outline: 2px dashed $common-node-active;

outline-offset: 0px;

}

}

.common-diamond-node {

position: absolute;

height: 50px;

width: 50px;

line-height: 50px;

text-align: center;

border: 1px solid #777;

border-radius: 3px;

background-color: $common-node-bg;

transform: rotate(45deg);

white-space: nowrap;

&:before {

position: absolute;

content: '网关';

transform: rotate(-45deg);

top: 0;

left: 0;

right: 0;

bottom: 0;

}

&:hover {

background-color: $common-node-bg-hover;

z-index: 2;

}

&.active {

outline: 2px dashed $common-node-active;

outline-offset: 0px;

}

}

.common-x-lane-node {

position: absolute;

text-align: center;

border: 1px solid #777;

border-radius: 2px;

z-index: -1;

&.active {

outline: 2px dashed $common-node-active;

outline-offset: 0px;

}

.lane-text-div {

width: 18px;

height: 100%;

position: absolute;

display: table;

border-right: 1px solid #777;

background-color: $common-node-bg;

&:hover {

z-index: 2;

}

.lane-text {

word-wrap: break-word;

display: table-cell;

vertical-align: middle;

font-size: 0.8em;

}

}

}

.common-y-lane-node {

position: absolute;

text-align: center;

border: 1px solid #777;

border-radius: 2px;

z-index: -1;

&.active {

outline: 2px dashed $common-node-active;

outline-offset: 0px;

}

.lane-text-div {

width: 100%;

height: 18px;

position: absolute;

display: table;

border-bottom: 1px solid #777;

background-color: $common-node-bg;

&:hover {

z-index: 2;

}

.lane-text {

word-wrap: break-word;

display: table-cell;

font-size: 0.8em;

}

}

}

.node-icon {

position: absolute;

top: 3px;

left: 3px;

}

< template >

<

div style = "width: 100%; height: 100%; overflow: hidden; position: relative;" >

<

div v -

if = "container.auxiliaryLine.isOpen && container.auxiliaryLine.isShowXLine" class = "auxiliary-line-x": style =

"{ top: auxiliaryLinePos.y + 'px' }" > < /div> <

div v -

if = "container.auxiliaryLine.isOpen && container.auxiliaryLine.isShowYLine" class = "auxiliary-line-y": style =

"{ left: auxiliaryLinePos.x + 'px' }" > < /div> <

div id = "flowContainer"

class = "flow-container": class =

"{ grid: flowData.config.showGrid, zoomIn: currentTool.type == 'zoom-in', zoomOut: currentTool.type == 'zoom-out', canScale: container.scaleFlag, canDrag: container.dragFlag, canMultiple: rectangleMultiple.flag }":

style =

"{ top: container.pos.top + 'px', left: container.pos.left + 'px', transform: 'scale(' + container.scale + ')', transformOrigin: container.scaleOrigin.x + 'px ' + container.scaleOrigin.y + 'px' }"

@click.stop = "containerHandler"

@mousedown = "mousedownHandler"

@mousemove = "mousemoveHandler"

@mouseup = "mouseupHandler"

@mousewheel = "scaleContainer"

@DOMMouseScroll = "scaleContainer"

@contextmenu = "showContainerContextMenu" >

<

flow - node

v -

for = "(node, index) in flowData.nodeList": key = "index": node = "node": plumb = "plumb": select.sync =

"currentSelect": selectGroup.sync = "currentSelectGroup": currentTool = "currentTool"

@showNodeContextMenu = "showNodeContextMenu"

@isMultiple = "isMultiple"

@updateNodePos = "updateNodePos"

@alignForLine = "alignForLine"

@hideAlignLine = "hideAlignLine" >

<

/flow-node> <

div class = "rectangle-multiple"

v -

if = "rectangleMultiple.flag && rectangleMultiple.multipling": style =

"{ top: rectangleMultiple.position.top + 'px', left: rectangleMultiple.position.left + 'px', width: rectangleMultiple.width + 'px', height: rectangleMultiple.height + 'px' }" >

<

/div> <

/div> <

div class = "container-scale" >

缩放倍数： {

{

container.scaleShow

}

} %

<

/div> <

div class = "mouse-position" >

x: {

{

mouse.position.x

}

}, y: {

{

mouse.position.y

}

} <

/div> <

vue - context - menu: contextMenuData = "containerContextMenuData"

@flowInfo = "flowInfo"

@paste = "paste"

@selectAll = "selectAll"

@saveFlow = "saveFlow"

@verticaLeft = "verticaLeft"

@verticalCenter = "verticalCenter"

@verticalRight = "verticalRight"

@levelUp = "levelUp"

@levelCenter = "levelCenter"

@levelDown = "levelDown"

@addRemark = "addRemark" >

<

/vue-context-menu> <

vue - context - menu: contextMenuData = "nodeContextMenuData"

@copyNode = "copyNode"

@deleteNode = "deleteNode" >

<

/vue-context-menu> <

/div> <

/template>

<

script >

import jsplumb from 'jsplumb'

import {

flowConfig

} from '../config/args-config.js'

import $ from 'jquery'

import 'jquery-ui/ui/widgets/draggable'

import 'jquery-ui/ui/widgets/droppable'

import 'jquery-ui/ui/widgets/resizable'

import {

ZFSN

} from '../util/ZFSN.js'

import FlowNode from './FlowNode'

export default {

props: ['browserType', 'flowData', 'plumb', 'select', 'selectGroup', 'currentTool'],

components: {

jsplumb,

FlowNode

},

mounted() {

this.initFlowArea();

},

data() {

return {

ctx: null,

currentSelect: this.select,

currentSelectGroup: this.selectGroup,

container: {

pos: {

top: -3000,

left: -3000

},

dragFlag: false,

draging: false,

scale: flowConfig.defaultStyle.containerScale.init,

scaleFlag: false,

scaleOrigin: {

x: 0,

y: 0

},

scaleShow: ZFSN.mul(flowConfig.defaultStyle.containerScale.init, 100),

auxiliaryLine: {

isOpen: flowConfig.defaultStyle.isOpenAuxiliaryLine,

isShowXLine: false,

isShowYLine: false,

controlFnTimesFlag: true

}

},

auxiliaryLinePos: {

x: 0,

y: 0

},

mouse: {

position: {

x: 0,

y: 0

},

tempPos: {

x: 0,

y: 0

}

},

rectangleMultiple: {

flag: false,

multipling: false,

position: {

top: 0,

left: 0

},

height: 0,

width: 0

},

containerContextMenuData: flowConfig.contextMenu.container,

nodeContextMenuData: flowConfig.contextMenu.node,

tempLinkId: '',

clipboard: []

}

},

methods: {

initFlowArea() {

const that = this;

that.ctx = document.getElementById('flowContainer').parentNode;

$('.flow-container').droppable({

accept: function(t) {

if (t[0].className.indexOf('node-item') != -1) {

let event = window.event || 'firefox';

if (that.ctx.contains(event.srcElement) || event == 'firefox') {

return true;

}

}

return false;

},

hoverClass: 'flow-container-active',

drop: function(event, ui) {

let belongto = ui.draggable.attr('belongto');

let type = ui.draggable.attr('type');

that.$emit('selectTool', 'drag');

that.$emit('findNodeConfig', belongto, type, node => {

if (!node) {

that.$message.error('未知的节点类型！');

return;

}

that.addNewNode(node);

});

}

});

},

mousedownHandler(e) {

const that = this;

let event = window.event || e;

if (event.button == 0) {

if (that.container.dragFlag) {

that.mouse.tempPos = that.mouse.position;

that.container.draging = true;

}

that.currentSelectGroup = [];

if (that.rectangleMultiple.flag) {

that.mouse.tempPos = that.mouse.position;

that.rectangleMultiple.multipling = true;

}

}

},

mousemoveHandler(e) {

const that = this;

let event = window.event || e;

if (event.target.id == 'flowContainer') {

that.mouse.position = {

x: event.offsetX,

y: event.offsetY

};

} else {

let cn = event.target.className;

let tn = event.target.tagName;

if (cn != 'lane-text' && cn != 'lane-text-div' && tn != 'svg' && tn != 'path' && tn != 'I') {

that.mouse.position.x = event.target.offsetLeft + event.offsetX;

that.mouse.position.y = event.target.offsetTop + event.offsetY;

}

}

if (that.container.draging) {

let nTop = that.container.pos.top + (that.mouse.position.y - that.mouse.tempPos.y);

let nLeft = that.container.pos.left + (that.mouse.position.x - that.mouse.tempPos.x);

if (nTop >= 0) nTop = 0;

if (nLeft >= 0) nLeft = 0;

that.container.pos = {

top: nTop,

left: nLeft

}

}

if (that.rectangleMultiple.multipling) {

let h = that.mouse.position.y - that.mouse.tempPos.y;

let w = that.mouse.position.x - that.mouse.tempPos.x;

let t = that.mouse.tempPos.y;

let l = that.mouse.tempPos.x;

if (h >= 0 && w < 0) {

w = -w;

l -= w;

} else if (h < 0 && w >= 0) {

h = -h;

t -= h;

} else if (h < 0 && w < 0) {

h = -h;

w = -w;

t -= h;

l -= w;

}

that.rectangleMultiple.height = h;

that.rectangleMultiple.width = w;

that.rectangleMultiple.position.top = t;

that.rectangleMultiple.position.left = l;

}

},

mouseupHandler() {

const that = this;

if (that.container.draging) that.container.draging = false;

if (that.rectangleMultiple.multipling) {

that.judgeSelectedNode();

that.rectangleMultiple.multipling = false;

that.rectangleMultiple.width = 0;

that.rectangleMultiple.height = 0;

}

},

judgeSelectedNode() {

const that = this;

let ay = that.rectangleMultiple.position.top;

let ax = that.rectangleMultiple.position.left;

let by = ay + that.rectangleMultiple.height;

let bx = ax + that.rectangleMultiple.width;

let nodeList = that.flowData.nodeList;

nodeList.forEach(function(node, index) {

if (node.y >= ay && node.x >= ax && node.y <= by && node.x <= bx) {

that.plumb.addToDragSelection(node.id);

that.currentSelectGroup.push(node);

}

});

},

scaleContainer(e) {

const that = this;

let event = window.event || e;

if (that.container.scaleFlag) {

if (that.browserType == 2) {

if (event.detail < 0) {

that.enlargeContainer();

} else {

that.narrowContainer();

}

} else {

if (event.deltaY < 0) {

that.enlargeContainer();

} else if (that.container.scale) {

that.narrowContainer();

}

}

}

},

enlargeContainer() {

const that = this;

that.container.scaleOrigin.x = that.mouse.position.x;

that.container.scaleOrigin.y = that.mouse.position.y;

let newScale = ZFSN.add(that.container.scale, flowConfig.defaultStyle.containerScale.onceEnlarge);

if (newScale <= flowConfig.defaultStyle.containerScale.max) {

that.container.scale = newScale;

that.container.scaleShow = ZFSN.mul(that.container.scale, 100);

that.plumb.setZoom(that.container.scale);

}

},

narrowContainer() {

const that = this;

that.container.scaleOrigin.x = that.mouse.position.x;

that.container.scaleOrigin.y = that.mouse.position.y;

let newScale = ZFSN.sub(that.container.scale, flowConfig.defaultStyle.containerScale.onceNarrow);

if (newScale >= flowConfig.defaultStyle.containerScale.min) {

that.container.scale = newScale;

that.container.scaleShow = ZFSN.mul(that.container.scale, 100);

that.plumb.setZoom(that.container.scale);

}

},

showContainerContextMenu(e) {

let event = window.event || e;

event.preventDefault();

$('.vue-contextmenuName-node-menu').css('display', 'none');

$('.vue-contextmenuName-link-menu').css('display', 'none');

this.selectContainer();

let x = event.clientX;

let y = event.clientY;

this.containerContextMenuData.axis = {

x,

y

};

},

showNodeContextMenu(e) {

let event = window.event || e;

event.preventDefault();

$('.vue-contextmenuName-flow-menu').css('display', 'none');

$('.vue-contextmenuName-link-menu').css('display', 'none');

let x = event.clientX;

let y = event.clientY;

this.nodeContextMenuData.axis = {

x,

y

};

},

flowInfo() {

const that = this;

let nodeList = that.flowData.nodeList;

let linkList = that.flowData.linkList;

alert('当前流程图中有 ' + nodeList.length + ' 个节点，有 ' + linkList.length + ' 条连线。');

},

paste() {

const that = this;

let dis = 0;

that.clipboard.forEach(function(node, index) {

let newNode = Object.assign({}, node);

newNode.id = newNode.type + '-' + ZFSN.getId();

let nodePos = that.computeNodePos(that.mouse.position.x + dis, that.mouse.position.y + dis);

newNode.x = nodePos.x;

newNode.y = nodePos.y;

dis += 20;

that.flowData.nodeList.push(newNode);

});

},

selectAll() {

const that = this;

that.flowData.nodeList.forEach(function(node, index) {

that.plumb.addToDragSelection(node.id);

that.currentSelectGroup.push(node);

});

},

saveFlow() {

this.$emit('saveFlow');

},

checkAlign() {

if (this.currentSelectGroup.length < 2) {

this.$message.error('请选择至少两个节点！');

return false;

}

return true;

},

verticaLeft() {

const that = this;

if (!that.checkAlign()) return;

let nodeList = that.flowData.nodeList;

let selectGroup = that.currentSelectGroup;

let baseX = selectGroup[0].x;

let baseY = selectGroup[0].y;

for (let i = 1; i < selectGroup.length; i++) {

baseY = baseY + selectGroup[i - 1].height + flowConfig.defaultStyle.alignSpacing.vertical;

let f = nodeList.filter(n => n.id == selectGroup[i].id)[0];

f.tx = baseX;

f.ty = baseY;

that.plumb.animate(selectGroup[i].id, {

top: baseY,

left: baseX

}, {

duration: flowConfig.defaultStyle.alignDuration,

complete: function() {

f.x = f.tx;

f.y = f.ty;

}

});

}

},

verticalCenter() {

const that = this;

if (!that.checkAlign()) return;

let nodeList = that.flowData.nodeList;

let selectGroup = that.currentSelectGroup;

let baseX = selectGroup[0].x;

let baseY = selectGroup[0].y;

let firstX = baseX;

for (let i = 1; i < selectGroup.length; i++) {

baseY = baseY + selectGroup[i - 1].height + flowConfig.defaultStyle.alignSpacing.vertical;

baseX = firstX + ZFSN.div(selectGroup[0].width, 2) - ZFSN.div(selectGroup[i].width, 2);

let f = nodeList.filter(n => n.id == selectGroup[i].id)[0];

f.tx = baseX;

f.ty = baseY;

that.plumb.animate(selectGroup[i].id, {

top: baseY,

left: baseX

}, {

duration: flowConfig.defaultStyle.alignDuration,

complete: function() {

f.x = f.tx;

f.y = f.ty;

}

});

}

},

verticalRight() {

const that = this;

if (!that.checkAlign()) return;

let nodeList = that.flowData.nodeList;

let selectGroup = that.currentSelectGroup;

let baseX = selectGroup[0].x;

let baseY = selectGroup[0].y;

let firstX = baseX;

for (let i = 1; i < selectGroup.length; i++) {

baseY = baseY + selectGroup[i - 1].height + flowConfig.defaultStyle.alignSpacing.vertical;

baseX = firstX + selectGroup[0].width - selectGroup[i].width;

let f = nodeList.filter(n => n.id == selectGroup[i].id)[0];

f.tx = baseX;

f.ty = baseY;

that.plumb.animate(selectGroup[i].id, {

top: baseY,

left: baseX

}, {

duration: flowConfig.defaultStyle.alignDuration,

complete: function() {

f.x = f.tx;

f.y = f.ty;

}

});

}

},

levelUp() {

const that = this;

if (!that.checkAlign()) return;

let nodeList = that.flowData.nodeList;

let selectGroup = that.currentSelectGroup;

let baseX = selectGroup[0].x;

let baseY = selectGroup[0].y;

for (let i = 1; i < selectGroup.length; i++) {

baseX = baseX + selectGroup[i - 1].width + flowConfig.defaultStyle.alignSpacing.level;

let f = nodeList.filter(n => n.id == selectGroup[i].id)[0];

f.tx = baseX;

f.ty = baseY;

that.plumb.animate(selectGroup[i].id, {

top: baseY,

left: baseX

}, {

duration: flowConfig.defaultStyle.alignDuration,

complete: function() {

f.x = f.tx;

f.y = f.ty;

}

});

}

},

levelCenter() {

const that = this;

if (!that.checkAlign()) return;

let nodeList = that.flowData.nodeList;

let selectGroup = that.currentSelectGroup;

let baseX = selectGroup[0].x;

let baseY = selectGroup[0].y;

let firstY = baseY;

for (let i = 1; i < selectGroup.length; i++) {

baseY = firstY + ZFSN.div(selectGroup[0].height, 2) - ZFSN.div(selectGroup[i].height, 2);

baseX = baseX + selectGroup[i - 1].width + flowConfig.defaultStyle.alignSpacing.level;

let f = nodeList.filter(n => n.id == selectGroup[i].id)[0];

f.tx = baseX;

f.ty = baseY;

that.plumb.animate(selectGroup[i].id, {

top: baseY,

left: baseX

}, {

duration: flowConfig.defaultStyle.alignDuration,

complete: function() {

f.x = f.tx;

f.y = f.ty;

}

});

}

},

levelDown() {

const that = this;

if (!that.checkAlign()) return;

let nodeList = that.flowData.nodeList;

let selectGroup = that.currentSelectGroup;

let baseX = selectGroup[0].x;

let baseY = selectGroup[0].y;

let firstY = baseY;

for (let i = 1; i < selectGroup.length; i++) {

baseY = firstY + selectGroup[0].height - selectGroup[i].height;

baseX = baseX + selectGroup[i - 1].width + flowConfig.defaultStyle.alignSpacing.level;

let f = nodeList.filter(n => n.id == selectGroup[i].id)[0];

f.tx = baseX;

f.ty = baseY;

that.plumb.animate(selectGroup[i].id, {

top: baseY,

left: baseX

}, {

duration: flowConfig.defaultStyle.alignDuration,

complete: function() {

f.x = f.tx;

f.y = f.ty;

}

});

}

},

addRemark() {

const that = this;

alert('添加备注(待完善)...');

},

copyNode() {

const that = this;

that.clipboard = [];

if (that.currentSelectGroup.length > 0) {

that.clipboard = Object.assign([], that.currentSelectGroup);

} else if (that.currentSelect.id) {

that.clipboard.push(that.currentSelect);

}

},

getConnectionsByNodeId(nodeId) {

const that = this;

let conns1 = that.plumb.getConnections({

source: nodeId

});

let conns2 = that.plumb.getConnections({

target: nodeId

});

return conns1.concat(conns2);

},

deleteNode() {

const that = this;

let nodeList = that.flowData.nodeList;

let linkList = that.flowData.linkList;

let arr = [];

arr.push(Object.assign({}, that.currentSelect));

arr.forEach(function(c, index) {

let conns = that.getConnectionsByNodeId(c.id);

conns.forEach(function(conn, index) {

linkList.splice(linkList.findIndex(link => (link.sourceId == conn.sourceId || link.targetId ==

conn.targetId)), 1);

});

that.plumb.deleteEveryEndpoint();

let inx = nodeList.findIndex(node => node.id == c.id);

nodeList.splice(inx, 1);

that.$nextTick(() => {

linkList.forEach(function(link, index) {

let conn = that.plumb.connect({

source: link.sourceId,

target: link.targetId,

anchor: flowConfig.jsPlumbConfig.anchor.default,

connector: [

link.cls.linkType,

{

gap: 5,

cornerRadius: 8,

alwaysRespectStubs: true

}

],

paintStyle: {

stroke: link.cls.linkColor,

strokeWidth: link.cls.linkThickness

}

});

if (link.label != '') {

conn.setLabel({

label: link.label,

cssClass: 'linkLabel'

});

}

});

});

});

that.selectContainer();

},

addNewNode(node) {

const that = this;

let x = that.mouse.position.x;

let y = that.mouse.position.y;

let nodePos = that.computeNodePos(x, y);

x = nodePos.x;

y = nodePos.y;

let newNode = Object.assign({}, node);

newNode.id = newNode.type + '-' + ZFSN.getId();

newNode.height = 50;

if (newNode.type == 'start' || newNode.type == 'end' ||

newNode.type == 'event' || newNode.type == 'gateway') {

newNode.x = x - 25;

newNode.width = 50;

} else {

newNode.x = x - 60;

newNode.width = 120;

}

newNode.y = y - 25;

if (newNode.type == 'x-lane') {

newNode.height = 200;

newNode.width = 600;

} else if (newNode.type == 'y-lane') {

newNode.height = 600;

newNode.width = 200;

}

that.flowData.nodeList.push(newNode);

},

computeNodePos(x, y) {

const pxx = flowConfig.defaultStyle.alignGridPX[0];

const pxy = flowConfig.defaultStyle.alignGridPX[1];

if (x % pxx) x = pxx - (x % pxx) + x;

if (y % pxy) y = pxy - (y % pxy) + y;

return {

x: x,

y: y

};

},

containerHandler() {

const that = this;

that.selectContainer();

let toolType = that.currentTool.type;

if (toolType == 'zoom-in') {

that.enlargeContainer();

} else if (toolType == 'zoom-out') {

that.narrowContainer();

}

},

selectContainer() {

this.currentSelect = {};

this.$emit('getShortcut');

},

isMultiple(callback) {

callback(this.rectangleMultiple.flag);

},

updateNodePos() {

const that = this;

let nodeList = that.flowData.nodeList;

that.currentSelectGroup.forEach(function(node, index) {

let l = parseInt($('#' + node.id).css('left'));

let t = parseInt($('#' + node.id).css('top'));

let f = nodeList.filter(n => n.id == node.id)[0];

f.x = l;

f.y = t;

});

},

alignForLine(e) {

const that = this;

if (that.selectGroup.length > 1) return;

if (that.container.auxiliaryLine.controlFnTimesFlag) {

let elId = e.el.id;

let nodeList = that.flowData.nodeList;

nodeList.forEach(function(node, index) {

if (elId != node.id) {

let dis = flowConfig.defaultStyle.showAuxiliaryLineDistance,

elPos = e.pos,

elH = e.el.offsetHeight,

elW = e.el.offsetWidth,

disX = elPos[0] - node.x,

disY = elPos[1] - node.y;

if ((disX >= -dis && disX <= dis) || ((disX + elW) >= -dis && (disX + elW) <= dis)) {

that.container.auxiliaryLine.isShowYLine = true;

that.auxiliaryLinePos.x = node.x + that.container.pos.left;

let nodeMidPointX = node.x + (node.width / 2);

if (nodeMidPointX == (elPos[0] + (elW / 2))) {

that.auxiliaryLinePos.x = nodeMidPointX + that.container.pos.left;

}

}

if ((disY >= -dis && disY <= dis) || ((disY + elH) >= -dis && (disY + elH) <= dis)) {

that.container.auxiliaryLine.isShowXLine = true;

that.auxiliaryLinePos.y = node.y + that.container.pos.top;

let nodeMidPointY = node.y + (node.height / 2);

if (nodeMidPointY == (elPos[1] + (elH / 2))) {

that.auxiliaryLinePos.y = nodeMidPointY + that.container.pos.left;

}

}

}

});

that.container.auxiliaryLine.controlFnTimesFlag = false;

setTimeout(function() {

that.container.auxiliaryLine.controlFnTimesFlag = true;

}, 200);

}

},

hideAlignLine() {

if (this.container.auxiliaryLine.isOpen) {

this.container.auxiliaryLine.isShowXLine = false;

this.container.auxiliaryLine.isShowYLine = false;

}

}

},

watch: {

select(val) {

this.currentSelect = val;

if (this.tempLinkId != '') {

$('#' + this.tempLinkId).removeClass('link-active');

this.tempLinkId = '';

}

if (this.currentSelect.type == 'link') {

this.tempLinkId = this.currentSelect.id;

$('#' + this.currentSelect.id).addClass('link-active');

}

},

currentSelect: {

handler(val) {

this.$emit('update:select', val);

},

deep: true

},

selectGroup(val) {

this.currentSelectGroup = val;

if (this.currentSelectGroup.length <= 0) this.plumb.clearDragSelection();

},

currentSelectGroup: {

handler(val) {

this.$emit('update:selectGroup', val);

},

deep: true

}

}

} <

/script>

<

style lang = "scss" >

@import '../style/flow-area.scss' <

/style>

< template >

<

div >

<

a - tabs size = "small"

defaultActiveKey = "flow-attr": activeKey = "activeKey" >

<

a - tab - pane key = "flow-attr" >

<

span slot = "tab" >

<

a - icon type = "cluster" / >

流程属性 <

/span> <

a - form layout = "horizontal" >

<

a - form - item label = "流程id": label - col = "formItemLayout.labelCol": wrapper - col =

"formItemLayout.wrapperCol" >

<

a - input: value = "flowData.attr.id"

disabled / >

<

/a-form-item> <

/a-form> <

/a-tab-pane> <

a - tab - pane key = "node-attr" >

<

span slot = "tab" >

<

a - icon type = "profile" / >

节点属性 <

/span> <

template v -

if = "currentSelect.type == 'start'" >

<

a - form layout = "horizontal" >

<

a - form - item label = "类型": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - tag color = "purple" > {

{

currentSelect.type

}

} < /a-tag> <

/a-form-item> <

a - form - item label = "id": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input: value = "currentSelect.id"

disabled / >

<

/a-form-item> <

a - form - item label = "名称": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input placeholder = "请输入节点名称": value = "currentSelect.nodeName"

@change = "nodeNameChange" / >

<

/a-form-item> <

/a-form> <

/template> <

template v -

if = "currentSelect.type == 'end'" >

<

a - form layout = "horizontal" >

<

a - form - item label = "类型": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - tag color = "purple" > {

{

currentSelect.type

}

} < /a-tag> <

/a-form-item> <

a - form - item label = "id": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input: value = "currentSelect.id"

disabled / >

<

/a-form-item> <

a - form - item label = "名称": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input placeholder = "请输入节点名称": value = "currentSelect.nodeName"

@change = "nodeNameChange" / >

<

/a-form-item> <

/a-form> <

/template> <

template v -

if = "currentSelect.type == 'common'" >

<

a - form layout = "horizontal" >

<

a - form - item label = "类型": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - tag color = "purple" > {

{

currentSelect.type

}

} < /a-tag> <

/a-form-item> <

a - form - item label = "id": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input: value = "currentSelect.id"

disabled / >

<

/a-form-item> <

a - form - item label = "名称": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input placeholder = "请输入节点名称": value = "currentSelect.nodeName"

@change = "nodeNameChange" / >

<

/a-form-item> <

/a-form> <

/template> <

template v -

else -

if = "currentSelect.type == 'freedom'" >

<

a - form layout = "horizontal" >

<

a - form - item label = "类型": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - tag color = "purple" > {

{

currentSelect.type

}

} < /a-tag> <

/a-form-item> <

a - form - item label = "id": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input: value = "currentSelect.id"

disabled / >

<

/a-form-item> <

a - form - item label = "名称": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input placeholder = "请输入节点名称": value = "currentSelect.nodeName"

@change = "nodeNameChange" / >

<

/a-form-item> <

/a-form> <

/template> <

template v -

else -

if = "currentSelect.type == 'event'" >

<

a - form layout = "horizontal" >

<

a - form - item label = "类型": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - tag color = "purple" > {

{

currentSelect.type

}

} < /a-tag> <

/a-form-item> <

a - form - item label = "id": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input: value = "currentSelect.id"

disabled / >

<

/a-form-item> <

a - form - item label = "名称": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input placeholder = "请输入节点名称": value = "currentSelect.nodeName"

@change = "nodeNameChange" / >

<

/a-form-item> <

/a-form> <

/template> <

template v -

else -

if = "currentSelect.type == 'gateway'" >

<

a - form layout = "horizontal" >

<

a - form - item label = "类型": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - tag color = "purple" > {

{

currentSelect.type

}

} < /a-tag> <

/a-form-item> <

a - form - item label = "id": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input: value = "currentSelect.id"

disabled / >

<

/a-form-item> <

a - form - item label = "名称": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input placeholder = "请输入节点名称": value = "currentSelect.nodeName"

@change = "nodeNameChange" / >

<

/a-form-item> <

/a-form> <

/template> <

template v -

else -

if = "currentSelect.type == 'child-flow'" >

<

a - form layout = "horizontal" >

<

a - form - item label = "类型": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - tag color = "purple" > {

{

currentSelect.type

}

} < /a-tag> <

/a-form-item> <

a - form - item label = "id": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input: value = "currentSelect.id"

disabled / >

<

/a-form-item> <

a - form - item label = "名称": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input placeholder = "请输入节点名称": value = "currentSelect.nodeName"

@change = "nodeNameChange" / >

<

/a-form-item> <

/a-form> <

/template> <

template v -

else -

if = "currentSelect.type == 'x-lane' || currentSelect.type == 'y-lane'" >

<

a - form layout = "horizontal" >

<

a - form - item label = "类型": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - tag color = "purple" > {

{

currentSelect.type

}

} < /a-tag> <

/a-form-item> <

a - form - item label = "id": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input: value = "currentSelect.id"

disabled / >

<

/a-form-item> <

a - form - item label = "名称": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input placeholder = "请输入节点名称": value = "currentSelect.nodeName"

@change = "nodeNameChange" / >

<

/a-form-item> <

/a-form> <

/template> <

/a-tab-pane> <

a - tab - pane key = "link-attr" >

<

span slot = "tab" >

<

a - icon type = "branches" / >

连线属性 <

/span> <

a - form layout = "horizontal" >

<

a - form - item label = "id": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input: value = "currentSelect.id"

disabled / >

<

/a-form-item> <

a - form - item label = "源节点": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input: value = "currentSelect.sourceId"

disabled / >

<

/a-form-item> <

a - form - item label = "目标节点": label - col = "formItemLayout.labelCol": wrapper - col =

"formItemLayout.wrapperCol" >

<

a - input: value = "currentSelect.targetId"

disabled / >

<

/a-form-item> <

a - form - item label = "文本": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - input: value = "currentSelect.label"

@change = "linkLabelChange" / >

<

/a-form-item> <

/a-form> <

/a-tab-pane> <

/a-tabs> <

/div> <

/template>

<

script >

import jsplumb from 'jsplumb'

import {

flowConfig

} from '../config/args-config.js'

export default {

props: ['plumb', 'flowData', 'select'],

components: {

jsplumb

},

data() {

return {

currentSelect: this.select,

formItemLayout: {

labelCol: {

span: 6

},

wrapperCol: {

span: 16

}

},

activeKey: 'flow-attr'

}

},

methods: {

nodeNameChange(e) {

this.currentSelect.nodeName = e.target.value;

},

linkLabelChange(e) {

const that = this;

let label = e.target.value;

that.currentSelect.label = label;

let conn = that.plumb.getConnections({

source: that.currentSelect.sourceId,

target: that.currentSelect.targetId

})[0];

if (label != '') {

conn.setLabel({

label: label,

cssClass: 'linkLabel'

});

} else {

let labelOverlay = conn.getLabelOverlay();

if (labelOverlay) conn.removeOverlay(labelOverlay.id);

}

}

},

watch: {

select(val) {

this.currentSelect = val;

if (this.currentSelect.type == 'link') {

this.activeKey = 'link-attr';

} else if (!this.currentSelect.type) {

this.activeKey = 'flow-attr';

} else {

this.activeKey = 'node-attr';

}

},

currentSelect: {

handler(val) {

this.$emit('update:select', val);

},

deep: true

}

}

} <

/script>

<

style lang = "scss" >

@import '../style/flow-attr.scss' <

/style>

< template >

<

div v -

if = "node.type == 'start'": id = "node.id"

class = "common-circle-node": class = "{ active: isActive() }": style = "{ top: node.y + 'px', left: node.x + 'px',

cursor: currentTool.type == 'drag' ? 'move' : (currentTool.type == 'connection' ? 'crosshair' :

(currentTool.type == 'zoom-in' ? 'zoom-in' :

(currentTool.type == 'zoom-out' ? 'zoom-out' : 'default')))

}

"

@click.stop = "selectNode"

@contextmenu.stop = "showNodeContextMenu" > {

{

node.nodeName

}

} <

/div>

<

div v -

else -

if = "node.type == 'end'": id = "node.id"

class = "common-circle-node": class = "{ active: isActive() }": style = "{ top: node.y + 'px', left: node.x + 'px',

cursor: currentTool.type == 'drag' ? 'move' : (currentTool.type == 'connection' ? 'crosshair' :

(currentTool.type == 'zoom-in' ? 'zoom-in' :

(currentTool.type == 'zoom-out' ? 'zoom-out' : 'default')))

}

"

@click.stop = "selectNode"

@contextmenu.stop = "showNodeContextMenu" > {

{

node.nodeName

}

} <

/div>

<

div v -

else -

if = "node.type == 'common'": id = "node.id"

class = "common-rectangle-node": class = "{ active: isActive() }": style = "{ top: node.y + 'px', left: node.x + 'px',

cursor: currentTool.type == 'drag' ? 'move' : (currentTool.type == 'connection' ? 'crosshair' :

(currentTool.type == 'zoom-in' ? 'zoom-in' :

(currentTool.type == 'zoom-out' ? 'zoom-out' : 'default')))

}

"

@click.stop = "selectNode"

@contextmenu.stop = "showNodeContextMenu" >

<

a - icon type = "user"

class = "node-icon" / > {

{

node.nodeName

}

} <

/div>

<

div v -

else -

if = "node.type == 'freedom'": id = "node.id"

class = "common-rectangle-node": class = "{ active: isActive() }": style = "{ top: node.y + 'px', left: node.x + 'px',

cursor: currentTool.type == 'drag' ? 'move' : (currentTool.type == 'connection' ? 'crosshair' :

(currentTool.type == 'zoom-in' ? 'zoom-in' :

(currentTool.type == 'zoom-out' ? 'zoom-out' : 'default')))

}

"

@click.stop = "selectNode"

@contextmenu.stop = "showNodeContextMenu" >

<

a - icon type = "sync"

class = "node-icon" / > {

{

node.nodeName

}

} <

/div>

<

div v -

else -

if = "node.type == 'event'": id = "node.id"

class = "common-circle-node": class = "{ active: isActive() }": style = "{ top: node.y + 'px', left: node.x + 'px',

cursor: currentTool.type == 'drag' ? 'move' : (currentTool.type == 'connection' ? 'crosshair' :

(currentTool.type == 'zoom-in' ? 'zoom-in' :

(currentTool.type == 'zoom-out' ? 'zoom-out' : 'default')))

}

"

@click.stop = "selectNode"

@contextmenu.stop = "showNodeContextMenu" > {

{

node.nodeName

}

} <

/div>

<

div v -

else -

if = "node.type == 'gateway'": id = "node.id"

class = "common-diamond-node": class = "{ active: isActive() }": style = "{ top: node.y + 'px', left: node.x + 'px',

cursor: currentTool.type == 'drag' ? 'move' : (currentTool.type == 'connection' ? 'crosshair' :

(currentTool.type == 'zoom' ? 'zoom-in' :

(currentTool.type == 'zoom-out' ? 'zoom-out' : 'default')))

}

"

@click.stop = "selectNode"

@contextmenu.stop = "showNodeContextMenu" >

<

/div>

<

div v -

else -

if = "node.type == 'child-flow'": id = "node.id"

class = "common-rectangle-node": class = "{ active: isActive() }": style = "{ top: node.y + 'px', left: node.x + 'px',

cursor: currentTool.type == 'drag' ? 'move' : (currentTool.type == 'connection' ? 'crosshair' :

(currentTool.type == 'zoom-in' ? 'zoom-in' :

(currentTool.type == 'zoom-out' ? 'zoom-out' : 'default')))

}

"

@click.stop = "selectNode"

@contextmenu.stop = "showNodeContextMenu" >

<

a - icon type = "api"

class = "node-icon" / > {

{

node.nodeName

}

} <

/div>

<

div v -

else -

if = "node.type == 'x-lane'": id = "node.id"

class = "common-x-lane-node": class = "{ active: isActive() }": style =

"{ top: node.y + 'px', left: node.x + 'px', height: node.height + 'px', width: node.width + 'px',

cursor: currentTool.type == 'zoom-in' ? 'zoom-in' : (currentTool.type == 'zoom-out' ? 'zoom-out' : 'default')

}

"> <

div class = "lane-text-div": style = "{ cursor: currentTool.type == 'drag' ? 'move' : 'default' }"

@click.stop = "selectNode"

@contextmenu.stop = "showNodeContextMenu" >

<

span class = "lane-text" > {

{

node.nodeName

}

} < /span> <

/div> <

/div>

<

div v -

else -

if = "node.type == 'y-lane'": id = "node.id"

class = "common-y-lane-node": class = "{ active: isActive() }": style =

"{ top: node.y + 'px', left: node.x + 'px', height: node.height + 'px', width: node.width + 'px',

cursor: currentTool.type == 'zoom-in' ? 'zoom-in' : (currentTool.type == 'zoom-out' ? 'zoom-out' : 'default')

}

"> <

div class = "lane-text-div": style = "{ cursor: currentTool.type == 'drag' ? 'move' : 'default' }"

@click.stop = "selectNode"

@contextmenu.stop = "showNodeContextMenu" >

<

span class = "lane-text" > {

{

node.nodeName

}

} < /span> <

/div> <

/div>

<

div v -

else > < /div> <

/template>

<

script >

import jsplumb from 'jsplumb'

import {

flowConfig

} from '../config/args-config.js'

import $ from 'jquery'

import 'jquery-ui/ui/widgets/draggable'

import 'jquery-ui/ui/widgets/droppable'

import 'jquery-ui/ui/widgets/resizable'

import {

ZFSN

} from '../util/ZFSN.js'

export default {

props: ['select', 'selectGroup', 'node', 'plumb', 'currentTool'],

components: {

jsplumb

},

mounted() {

this.registerNode();

},

data() {

return {

currentSelect: this.select,

currentSelectGroup: this.selectGroup

}

},

methods: {

registerNode() {

const that = this;

that.plumb.draggable(that.node.id, {

containment: 'parent',

handle: function(e, el) {

var possibles = el.parentNode.querySelectorAll(

'.common-circle-node,.common-rectangle-node,.common-diamond-node,.lane-text-div'

);

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

if (possibles[i] === el || e.target.className == 'lane-text') return true;

}

return false;

},

grid: flowConfig.defaultStyle.alignGridPX,

drag: function(e) {

if (flowConfig.defaultStyle.isOpenAuxiliaryLine) {

that.$emit('alignForLine', e);

}

},

stop: function(e) {

that.node.x = e.pos[0];

that.node.y = e.pos[1];

if (that.currentSelectGroup.length > 1) {

that.$emit('updateNodePos');

}

that.$emit('hideAlignLine');

}

});

if (that.node.type == 'x-lane' || that.node.type == 'y-lane') {

$('#' + that.node.id).resizable({

minHeight: 200,

minWidth: 200,

maxHeight: 2000,

maxWidth: 2000,

ghost: true,

autoHide: true,

stop: function(event, ui) {

that.node.height = ui.size.height;

that.node.width = ui.size.width;

}

});

}

that.currentSelect = that.node;

that.currentSelectGroup = [];

},

selectNode() {

const that = this;

that.currentSelect = this.node;

that.$emit('isMultiple', flag => {

if (!flag) {

that.currentSelectGroup = [];

} else {

let f = that.currentSelectGroup.filter(s => s.id == that.node.id);

if (f.length <= 0) {

that.plumb.addToDragSelection(that.node.id);

that.currentSelectGroup.push(that.node);

}

}

});

},

showNodeContextMenu(e) {

this.$emit('showNodeContextMenu', e);

this.selectNode();

},

isActive() {

const that = this;

if (that.currentSelect.id == that.node.id) return true;

let f = that.currentSelectGroup.filter(n => n.id == that.node.id);

if (f.length > 0) return true;

return false;

}

},

watch: {

select(val) {

this.currentSelect = val;

},

currentSelect: {

handler(val) {

this.$emit('update:select', val);

},

deep: true

},

selectGroup(val) {

this.currentSelectGroup = val;

},

currentSelectGroup: {

handler(val) {

this.$emit('update:selectGroup', val);

},

deep: true

}

}

} <

/script>

<

style lang = "scss" >

@import '../style/flow-node.scss' <

/style>

< template >

<

div >

<

a - drawer

title = "设置"

placement = "left": width = "600": visible = "settingVisible"

@close = "close" >

<

a - form: form = "settingForm"

layout = "horizontal" >

<

a - divider orientation = "left" > 画布 < /a-divider> <

a - form - item label = "缩小比例": label - col = "formItemLayout.labelCol": wrapper - col =

"formItemLayout.wrapperCol" >

<

a - slider: min = "0.05": max = "0.5": step = "0.05": tipFormatter = "formatterContainerOnceNarrow"

v - decorator = "['containerOnceNarrow', {}]"

@afterChange = "setContainerOnceNarrow" / >

<

/a-form-item> <

a - form - item label = "放大比例": label - col = "formItemLayout.labelCol": wrapper - col =

"formItemLayout.wrapperCol" >

<

a - slider: min = "0.05": max = "0.5": step = "0.05": tipFormatter = "formatterContainerOnceEnlarge"

v - decorator = "['containerOnceEnlarge', {}]"

@afterChange = "setContainerOnceEnlarge" / >

<

/a-form-item>

<

a - divider orientation = "left" > 连线 < /a-divider> <

a - form - item label = "类型": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - select v - decorator = "['linkType', {}]"

@change = "setFlowType" >

<

a - select - option value = "Bezier" > 贝塞尔曲线 < /a-select-option> <

a - select - option value = "Straight" > 直线 < /a-select-option> <

a - select - option value = "Flowchart" > 流程图线 < /a-select-option> <

a - select - option value = "StateMachine" > 状态线 < /a-select-option> <

/a-select> <

/a-form-item> <

a - form - item label = "颜色": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

colorPicker v - model = "linkColor"

@change = "setLinkColor" / >

<

/a-form-item> <

a - form - item label = "粗细": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a - slider: min = "1": max = "10"

v - decorator = "['linkThickness', {}]"

@afterChange = "setStrokeWidth" / >

<

/a-form-item>

<

a - divider orientation = "left" > 默认样式 < /a-divider> <

a - form - item label = "辅助线": label - col = "formItemLayout.labelCol": wrapper - col = "formItemLayout.wrapperCol" >

<

a -

switch: checked = "isOpenAuxiliaryLine"

v - decorator = "['isOpenAuxiliaryLine', {}]"

checkedChildren = "开"

unCheckedChildren = "关"

@change = 'toggleOpenAuxiliaryLine' / >

<

/a-form-item> <

a - form - item label = "自动对齐水平间距": label - col = "formItemLayout.labelCol": wrapper - col =

"formItemLayout.wrapperCol" >

<

a - slider: min = "10": max = "800": step = "5"

v - decorator = "['alignLevelDistance', {}]"

@afterChange = "setAlignLevelDistance" / >

<

/a-form-item> <

a - form - item label = "自动对齐垂直间距": label - col = "formItemLayout.labelCol": wrapper - col =

"formItemLayout.wrapperCol" >

<

a - slider: min = "10": max = "800": step = "5"

v - decorator = "['alignVerticalDistance', {}]"

@afterChange = "setAlignVerticalDistance" / >

<

/a-form-item> <

a - form - item label = "微移距离": label - col = "formItemLayout.labelCol": wrapper - col =

"formItemLayout.wrapperCol" >

<

a - slider: min = "1"

v - decorator = "['movePx', {}]"

@afterChange = "setMovePx" / >

<

/a-form-item> <

/a-form> <

/a-drawer> <

/div> <

/template>

<

script >

import {

flowConfig

} from '../config/args-config.js'

export default {

data() {

return {

settingVisible: false,

formItemLayout: {

labelCol: {

span: 6

},

wrapperCol: {

span: 15

}

},

setLinkColor(v) {

this.linkColor = v;

flowConfig.jsPlumbInsConfig.PaintStyle.stroke = v;

},

setStrokeWidth(v) {

flowConfig.jsPlumbInsConfig.PaintStyle.strokeWidth = v;

},

setAlignLevelDistance(v) {

flowConfig.defaultStyle.alignSpacing.level = v;

},

setAlignVerticalDistance(v) {

flowConfig.defaultStyle.alignSpacing.vertical = v;

},

formatterContainerOnceNarrow(v) {

return `${v\*100}%`;

},

setContainerOnceNarrow(v) {

flowConfig.defaultStyle.containerScale.onceNarrow = v;

},

formatterContainerOnceEnlarge(v) {

return `${v\*100}%`;

},

setContainerOnceEnlarge(v) {

flowConfig.defaultStyle.containerScale.onceEnlarge = v;

}

}

} <

/script>

<

style >

.m - colorPicker.box {

z - index: 9999!important;

width: 220 px!important;

}

.ant - divider - horizontal.ant - divider - with - text, .ant - divider - horizontal.ant - divider - with - text -

left, .ant - divider - horizontal.ant - divider - with - text - right {

font - weight: 800;

margin: 24 px 0 4 px;

} <

/style>