【组件】SvForm-表单

SvForm表单

在ElementUI的Form表单组件进行二次封装，由输入框、选择器、单选框、多选框、附件、开关等组成，用以收集、校验、提交数据

示例说明

示例网址：Savor-Enterprise

账号：admin 密码：123456

源代码：src\components\SvForm\index.vue

示例代码：src\views\admin\demo\SvFormDemo.vue

使用说明

使用方式：

已经注册成全局组件SvForm，直接通过<sv-form ref="SvForm"><sv-form>的方式在template里面引用，通过fieldList配置表单中的属性。

表单属性

参数	说明	类型	是否必填	默认值
form	表单数据对象	Object	是	—
field-list	表单项的属性配置表，具体配置参考表1-fieldList配置表	Array	是	—
form-disabled	禁用表单	Boolean	否	false
label-position	表单域标签的位置，top/right，直接在SvForm中进行全局的设置	String	否	top
label-width	label-position为right是需要进行设置，表单域标签的宽度	String	否	110px

表单事件

方法名	说明	参数
validate	对整个表单进行校验的方法，返回布尔值。使用方法为调用this.$refs.SvForm.validate()，表单校验通过返回true，校验不通过返回false	—
@handle-event	除了singlePopup类型以外的其他类型，内容改变时会回调handle-event方法	Function(event: String, data: String|Boolean|Array)
@handle-click	singlePopup类型的回调事件，点击时回调	Function(event: String, prop: String)
@handle-clear	singlePopup类型的清空事件，和singlePopup的clearable属性配合使用	Function(event: String, prop: String)

表1-fieldList配置表(通用项)

参数	说明	类型	是否必填	默认值
type	表单项内容包括输入框，选择器等，具体参考表3-type类型配置表	String	是	—
label	标签文本	String	是	—
prop	表单域字段，表单数据对象中的字段	String	是	—
disabled	禁用	Boolean	否	false
rules	表单项验证规则，具体参考表2-rules表单验证规则配置表	Object	否	—
tooltip	提示信息	String/html	否	—
event	回调事件名称，每个表单项对应唯一的事件名，传入event，回调事件@handle-event，@handle-click，@handle-clear会把事件名称回传到会组件中	String	否	—
placeholder	占位符，文本框中的提示内容	String	否	请输入+label
clearable	清空表单项。如果除了清空表单项对应的属性值以外还要清空其他关联元素：非singlePopup类型可以在handleEvent中判断data的值，如果data为空清空其他关联元素；singlePopup类型通过回调事件handleClear来清空	Boolean	否	true
sm	col属性，768px 响应式栅格数或者栅格属性对象	Number	否	12

md	col992px 响应式栅格数或者栅格属性对象	Number	否	8
lg	1200px 响应式栅格数或者栅格属性对象	Number	否	6
xl	1920px 响应式栅格数或者栅格属性对象	Number	否	6
class	表单项的类名	String	否	—
style	表单项的样式	Object	否	—

表2-rules表单验证规则配置表

参数	说明
required	是否必填
validator	自定义校验规则
message	必填时自定义提示信息，默认为该输入项为必填项
min	字段最小长度
max	字段最大长度
equal	字段长度为
type	类型包括英文，英数字，中文，非中文等，如果要添加新类型需要修改inputRules.js和validate.js

表3-type类型配置表

序号	参数	名称	类型	说明
1	text	文本输入框	String	普通文本的输入
2	multipleInput	多文本输入框	String	多条文本信息输入，以","分隔
3	textarea	文本域	String	较长文本内容的输入
4	password	密码框	String	输入密码
5	passProgress	带进度条密码框	String	输入密码，会有密码强度提示
6	singlePopup	复合型输入框	String	通过弹出框选择数据
7	number	计数器	Number	基本的数量等
8	select	单选选择器	String/Number	下拉框选择，单选
9	multipleSelect	多选选择器	Array	下拉框选择，多选
10	cascader	级联选择器	Array	级联多级选择
11	citySelect	省市区选择器	Array	选择国家/省/市/区
12	date	日期	String	选择日期
13	datetime	时间	String	选择日期和时间
14	daterange	日期范围	Array	选择一个日期范围
15	datetimerange	时间范围	Array	选择一个时间范围
16	radio	单选框	String/Number/Boolean	单选某个内容
17	checkbox	多选框	String	多选某些内容
18	switch	开关	String/Number/Boolean	打开/关闭等状态
19	upload	上传	Array	上传文件
20	slot	插槽	—	以上类型不满足，需要自定义表单项

fieldList配置表(不同type类型中个性化的属性)

1. text类型

```
// text
{
    type: 'text',
    label: 'text',
    prop: 'textType',
    rules: { required: true, equal: 3, validator: validateHaHaHa }, // equal, validator
    tooltip: '<li>type: text</li>' // html
    // event: 'textTypeEvent' // handle-event(event, form[prop])
    // placeholder: 'text', //
    // disabled: true, //
    // clearable: true, //
    // class: 'text-type-class' // <style></style>classel-input,
    // style: 'width: 60%' // el-input
}
```

2. multipleInput类型

```
// multipleInput
{
    type: 'multipleInput',
    label: 'multipleInput',
    prop: 'multipleInputType',
    rules: { required: true },
    tooltip: '<li>type: text","</li>' // html
    // event: 'multipleInputTypeEvent' // handle-event(event, form[prop])
    // placeholder: 'text', //
    // disabled: true, //
    // clearable: true, //
    // class: 'text-type-class' // <style></style>classel-input,
    // style: 'width: 60%' // el-input
}
```

3. textarea类型

```
{
        type: 'textarea',
        label: 'textarea',
        prop: 'textareaType',
        tooltip: 'type: textarealgmdsm',
        lg: 24, // 12001920241281/361/4
        md: 24, // 9921200241281/361/4
        sm: 24 // 768992241281/361/4
        // event: 'textareaTypeEvent' // handle-event(event, form[prop])
        // disabled: true, //
}
```

4. password类型

参数	说明	类型	是否必填	默认值
showPassword	密码是否可以查看	String	否	false
rules	{type: 'passwordStreflgth', strength: 'weak'}，校验时类型为passwordStreflgth，密码强度strength可选strong/medium/weak，默认strong	Boolean	否	true

```
{
    type: 'password',
    label: 'password',
    prop: 'passwordType',
    rules: { required: true, min: 6, max: 20 }, // min: max: , equal:
    tooltip: 'type: password', // html
    showPassword: true, //
    // event: 'passwordTypeEvent' // handle-event(event, form[prop])
    // placeholder: 'text', //
    // disabled: true, //
    // clearable: true, //
    class: 'text-type-class' // <style></style>classel-input,
    // style: 'width: 60%' // el-input
}
```

5. passProgress类型

参数	说明	类型	是否必填	默认值
showPassword	密码是否可以查看	String	否	false
rules	{type: 'passwordStreflgth', strength: 'weak'}，校验时类型为passwordStreflgth，密码强度strength可选strong/medium/weak，默认strong	Boolean	否	true

```
{
        type: 'passProgress',
        label: 'passProgress',
        prop: 'passProgressType',
        rules: { required: true, type: 'passwordStreflgth' }, // rulerequiredtype
        tooltip:
        'type: passProgress', // html
        showPassword: true //
        // event: 'passProgressTypeEvent' // handle-event(event, form[prop])
        // placeholder: 'text', //
        // disabled: true, //
        // class: 'text-type-class', // <style></style>classel-input
        // style: 'width: 60%' // el-input
}
```

6. number类型

参数	说明	类型	是否必填	默认值
min	最小值	Number	否	—
max	最大值	Number	否	—
precision	数值精度	Number	否	2

```
{
    type: 'number',
    label: 'number',
    prop: 'numberType',
    tooltip: 'type: number', // html
    min: -1111, // -10000
    max: 10000, //
    precision: 0 //
    // event: 'numberTypeEvent' // handle-event(event, form[prop])
    // placeholder: 'text', //
    // disabled: true, //
    // class: 'text-type-class', // <style></style>classel-input,
    // style: 'width: 60%' // el-input
}
```

7. singlePopup类型

```
{
    type: 'singlePopup',
    label: 'singlePopup',
    prop: 'singlePopupType',
    tooltip: 'type: singlePopup', // html
    event: 'singlePopupTypeEvent', // handle-click(event, prop)
    clearable: true // handle-clear(event, prop)
    // disabled: true, //
    // class: 'text-type-class', // <style></style>classel-input,
    // style: 'width: 60%' // el-input
}
```

8. select类型

参数	说明	类型	是否必填	默认值
filterable	是否可搜索	String	否	false
dict	数据字典，传入后自动生成下拉框选项	String	否	—
options	自定义下拉框选项，	Array	否	—
optSet	options的键名可通过 optSet 属性配置，例如optSet: { value: 'code', label: 'label' }，设置更改键名	Object	否	—

```
{
    type: 'select',
    label: 'select()',
    prop: 'selectCountry',
    tooltip: 'type: select', // html
    event: 'selectCountryEvent',
    filterable: true, //
    clearable: true, // handle-clear(event, prop)
    // { code: 'AD', id: 'AD', label: '', parentId: '0' }
    // keyvaluelabelvalue, value, label
    // optSet
    optSet: { value: 'code', label: 'label' }
    // disabled: true, //
    // class: 'text-type-class', // <style></style>classel-input,
    // style: 'width: 60%' // el-input
}
```

9. multipleSelect类型

参数	说明	类型	是否必填	默认值
filterable	是否可搜索	String	否	false
dict	数据字典，传入后自动生成下拉框选项	String	否	—
options	自定义下拉框选项，	Array	否	—
optSet	options的键名可通过 optSet 属性配置，例如optSet: { value: 'code', label: 'label' }，设置更改键名	Object	否	—
collapseTags	是否将选中值按文字的形式展示	Boolean	否	false

```
{
        type: 'multipleSelect',
        label: 'multipleSelect',
        prop: 'multipleSelectType',
        tooltip: 'type: multipleSelect', // html
        // event: 'multipleSelectTypeEvent', // handle-event(event, form[prop])
        collapseTags: false, //
        filterable: true, //
        clearable: true, // handle-clear(event, prop)
        // { code: 'AD', id: 'AD', label: '', parentId: '0' }
        // keyvaluelabelvalue, value, label
        // optSet
        optSet: { value: 'code', label: 'label' }
        // disabled: true, //
        // class: 'text-type-class', // <style></style>classel-input
        // style: 'width: 60%' // el-input
}
```

10. cascader类型

参数	说明	类型	是否必填	默认值
placeholder	占位符，文本框中的提示内容	String	否	请输入+label
clearable	清空表单项	Boolean	否	true

```
{
        type: 'cascader',
        label: 'cascader',
        prop: 'cascaderType',
        tooltip: 'type: cascader', // html
        // event: 'cascaderTypeEvent', // handle-event(event, form[prop])
        filterable: true, //
        clearable: true, // handle-clear(event, prop)
        // { code: 'AD', id: 'AD', label: '', parentId: '0' }
        // keyvaluelabelvalue, value, label
        // optSet
        optSet: { value: 'code', label: 'label', children: 'children' }
        // disabled: true, //
        // class: 'text-type-class', // <style></style>classel-input
        // style: 'width: 60%' // el-input
}
```

11. citySelect类型

参数	说明	类型	是否必填	默认值
region	默认显示中国，传入world，显示世界	String	否	CN
level	显示层级，默认中国level为2时显示省市	Number	否	4

```
{
        type: 'citySelect',
        label: 'citySelect',
        prop: 'citySelectType',
        tooltip: 'type: citySelect',
        region: 'CN', // world
        level: 2 // level2
}
```

12. date类型

参数	说明	类型	是否必填	默认值
pickerOptions	日期设置，disabledDate函数，选择某个范围的日期	Object	否	—

```
{
        type: 'date',
        label: 'date',
        prop: 'dateType',
        tooltip: 'type: date',
        pickerOptions: {
                disabledDate(time) {
                        //  return time.getTime() > new Date('2020/05/01')
                        //  return time.getTime() > Date.now() - 8.64e7
                        //  return time.getTime() > Date.now() - 8.64e7
                        // 2020/06/012020/06/30
                        return time.getTime() > new Date('2021/06/01') && time.getTime() < new Date('2021/06/31')
                }
        }
        // event: 'dateTypeEvent', // handle-event(event, form[prop])
        // disabled: true, //
        // class: 'text-type-class', // <style></style>classel-input
        // style: 'width: 60%' // el-input
}
```

13. datetime类型

参数	说明	类型	是否必填	默认值
pickerOptions	日期设置，disabledDate函数，选择某个范围的日期	Object	否	—

```
{
        type: 'datetime',
        label: 'datetime',
        prop: 'datetimeType',
        tooltip: 'type: datetime',
        pickerOptions: {
                disabledDate(time) {
                        return time.getTime() > Date.now() - 8.64e7
                }
        }
        // event: 'datetimeTypeEvent', // handle-event(event, form[prop])
        // disabled: true, //
        // class: 'text-type-class', // <style></style>classel-input
        // style: 'width: 60%' // el-input
}
```

14. daterange类型

参数	说明	类型	是否必填	默认值
pickerOptions	日期设置，disabledDate函数，选择某个范围的日期	Object	否	—
rangeSeparator	连接符	String	否	至
startPlaceholder	占位符，文本框中的提示内容	String	否	开始日期
endPlaceholder	占位符，文本框中的提示内容	String	否	结束日期

```
{
        type: 'daterange',
        label: 'daterange',
        prop: 'daterangeType',
        tooltip: 'type: daterange'
        // pickerOptions: {
        //   disabledDate(time) {
        //     return time.getTime() > Date.now() - 8.64e7
        //   }
        // }
        // rangeSeparator: '', //
        // startPlaceholder: '', //
        // endPlaceholder: '', //
        // event: 'daterangeTypeEvent', // handle-event(event, form[prop])
        // disabled: true, //
        // class: 'text-type-class', // <style></style>classel-input
        // style: 'width: 60%' // el-input
}
```

15. datetimerange类型

参数	说明	类型	是否必填	默认值
pickerOptions	日期设置，disabledDate函数，选择某个范围的日期	Object	否	—
rangeSeparator	连接符	String	否	至
startPlaceholder	占位符，文本框中的提示内容	String	否	开始时间
endPlaceholder	占位符，文本框中的提示内容	String	否	结束时间

```
{
        type: 'datetimerange',
        label: 'datetimerange',
        prop: 'datetimerangeType',
        tooltip: 'type: datetimerange'
        // pickerOptions: {
        //   disabledDate(time) {
                //     return time.getTime() > Date.now() - 8.64e7
        //   }
        // }
        // rangeSeparator: '', //
        // startPlaceholder: '', //
        // endPlaceholder: '', //
        // event: 'datetimerangeTypeEvent', // handle-event(event, form[prop])
        // disabled: true, //
        // class: 'text-type-class', // <style></style>classel-input
        // style: 'width: 60%' // el-input
}
```

16. radio类型

参数	说明	类型	是否必填	默认值
label	显示的内容和绑定的内容	String|Boolean|Number	否	—
optSet	{ label: '', name: ''}，如果不添加name绑定的和显示的都是teaCode，如果添加则绑定的时teaCode，显示的是teaName	Object	否	—

```
{
        type: 'radio',
        label: 'radio',
        prop: 'radioType',
        tooltip: 'type: radiolgmdsm',
        options: [{ label: '' }, { label: '' }, { label: '' }]
        // options: [{ teaName: '', teaCode: 'blackTea' }, { teaName: '', teaCode: 'flowTea' }],
        // optSetoptSet: { label: '', name: ''}nameteaCodeteaCodeteaName
        // optSet: { label: 'teaCode', name: 'teaName' }
        // event: 'radioTypeEvent', // handle-event(event, form[prop])
        // disabled: true, //
}
```

17. checkbox类型

参数	说明	类型	是否必填	默认值
label	显示的内容和绑定的内容	String|Boolean|Number	否	—
optSet	{ label: '', name: ''}，如果不添加name绑定的和显示的都是teaCode，如果添加则绑定的时teaCode，显示的是teaName	Object	否	—

```
{
        type: 'checkbox',
        label: 'checkbox',
        prop: 'checkboxType',
        tooltip: 'type: checkboxlgmdsm',
        options: [{ label: '' }, { label: '' }, { label: '' }]
// options: [{ foodName: '', foodCode: 'durian' }, { foodName: '', foodCode: 'pineapple' }],
        // optSetoptSet: { label: '', name: ''}namefoodCodefoodCodefoodName
        // optSet: { label: 'foodCode', name: 'foodName' }
        // event: 'checkboxTypeEvent', // handle-event(event, form[prop])
        // disabled: true, //
}
```

18. switch类型

参数	说明	类型	是否必填	默认值
activeText	switch 打开时的文字描述	String	否	开
inactiveText	switch 关闭时的文字描述	String	否	关
activeValue	switch 打开时的值	Boolean / String / Number	否	1
inactiveValue	switch 关闭时的值	Boolean / String / Number	否	0

```
{
        type: 'switch',
        label: 'switch',
        prop: 'switchType',
        tooltip: 'type: switch',
        // disabled: true, // ,
        activeText: '', //
        inactiveText: '', //
        activeValue: true, // 1
        inactiveValue: false // 0
        // event: 'switchTypeEvent', // handle-event(event, form[prop])
}
```

19. upload类型

参数	说明	类型	是否必填	默认值
storageType	存储方式，minio/ali-oss/local(本地存储)	String	否	minio
multiple	是否多附件上传，默认false	Boolean	否	false
limit	multiple为true，限制上传附件个数，默认3个	Boolean	否	3
badgeHidden	如果有limit限制，是否显示文件个数	Boolean	否	false
fileType	文件类型	String	否	all
listType	文件列表类型，text/picture/picture-card	String	否	text
buttonStyle	按钮样式，button/icon	String	否	button
bizDir	文件上传后文件夹名称	String	是	—

```
{
        type: 'upload',
        label: 'upload()',
        prop: 'uploadPictureType',
        rules: { required: true }, // rulerequiredtype
        // storageType: 'minio', // minio/ali-oss/local()minio
        multiple: true, // false
        // limit: 5, // 3
        // badgeHidden: true, // limit
        // fileType: 'all', // all
        // event: 'uploadPictureTypeEvent' // handle-event(event, form[prop])
        // disabled: true, //
        listType: 'picture-card', // text/picture/picture-card, text
        bizDir: 'SvFormDemo', //
        buttonStyle: 'icon', // , button/icon
        tooltip: 'type: uploadlgmdsm',
        lg: 24, // 12001920241281/361/4
        md: 24, // 9921200241281/361/4
        sm: 24 // 768992241281/361/4
}
```

20. slot类型

```
{
        type: 'slot', // slot
        label: 'slot',
        prop: 'slotType',
        rules: { required: true }
}
```

基础示例完整版

```
<template>
  <sv-form-container :buttons="buttons" @handle-click="handleClick">
    <sv-card title="SvForm">
      <sv-form
        ref="SvForm"
        :form.sync="form"
        :fieldList="fieldList"
        :formDisabled="formDisabled"
        @handle-event="handleEvent"
        @handle-click="handleClick"
        @handle-clear="handleClear"
      >
        <!-- type: slot  form-item-slotType, 'slotType'feildListprop -->
        <template v-slot:form-item-slotType>
          <el-input v-model="form.slotType" placeholder="slot" style="width: 75%;"></el-input>
          <el-checkbox v-model="form.isSlot" style="width: 20%;"></el-checkbox>
        </template>
      </sv-form>
    </sv-card>
    <!--  -->
    <el-dialog title="" :visible.sync="dialogUserVisible" :close-on-click-modal="false" width="60%">
      <select-user-demo @handle-cancel="dialogUserVisible = false" @handle-close="handleClose"></select-user-demo>
    </el-dialog>
  </sv-form-container>
</template>

<script>
import { getZipcode } from '@/api/admin/zipcode'
export default {
  name: 'SvFormDemo',
  components: {
    'select-user-demo': () => import('./components/SelectUserDemo')
  },
  data() {
    //
    const validateHaHaHa = (rule, value, callback) => {
      if (value !== '') {
        callback(new Error(''))
      } else {
        callback()
      }
    }
    return {
      // SvFormContainer
      buttons: [
        {
          type: 'primary',
          event: 'save',
          name: ''
        }
      ],
      form: {},
      //
      formDisabled: false,
      fieldList: [
        // text
        {
          type: 'text',
          label: 'text',
          prop: 'textType',
          rules: { required: true, equal: 3, validator: validateHaHaHa }, // equal, validator
          tooltip: '<li>type: text</li>' // html
          // event: 'textTypeEvent' // handle-event(event, form[prop])
          // placeholder: 'text', //
          // disabled: true, //
          // clearable: true, //
          // class: 'text-type-class' // <style></style>classel-input,
          // style: 'width: 60%' // el-input
        },
        // text
        {
          type: 'multipleInput',
          label: 'multipleInput',
          prop: 'multipleInputType',
          rules: { required: true },
          tooltip: '<li>type: text","</li>' // html
          // event: 'multipleInputTypeEvent' // handle-event(event, form[prop])
          // placeholder: 'text', //
          // disabled: true, //
          // clearable: true, //
          // class: 'text-type-class' // <style></style>classel-input,
          // style: 'width: 60%' // el-input
        },
```

```
{
  type: 'password',
  label: 'password',
  prop: 'passwordType',
  rules: { required: true, min: 6, max: 20 }, // min: max: , equal:
  tooltip: 'type: password', // html
  showPassword: true, //
  // event: 'passwordTypeEvent' // handle-event(event, form[prop])
  // placeholder: 'text', //
  // disabled: true, //
  // clearable: true, //
  class: 'text-type-class' // <style></style>classel-input,
  // style: 'width: 60%' // el-input
},
{
  type: 'number',
  label: 'number',
  prop: 'numberType',
  tooltip: 'type: number', // html
  min: -1111, // -10000
  max: 10000, //
  precision: 0 //
  // event: 'numberTypeEvent' // handle-event(event, form[prop])
  // placeholder: 'text', //
  // disabled: true, //
  // class: 'text-type-class', // <style></style>classel-input,
  // style: 'width: 60%' // el-input
},
{
  type: 'singlePopup',
  label: 'singlePopup',
  prop: 'singlePopupType',
  tooltip: 'type: singlePopup', // html
  event: 'singlePopupTypeEvent', // handle-click(event, prop)
  clearable: true // handle-clear(event, prop)
  // disabled: true, //
  // class: 'text-type-class', // <style></style>classel-input,
  // style: 'width: 60%' // el-input
},
{
  type: 'select',
  label: 'select()',
  prop: 'selectType',
  dict: 'yes_no',
  tooltip: 'type: select', // html
  event: 'selectTypeEvent',
  // filterable: true, //
  clearable: true // handle-clear(event, prop)
  // disabled: true, //
  // class: 'text-type-class', // <style></style>classel-input,
  // style: 'width: 60%' // el-input
},
{
  type: 'text',
  label: 'select',
  prop: 'relateText',
  disabled: true,
  tooltip: 'selectType' // html
},
{
  type: 'select',
  label: 'select()',
  prop: 'selectDisabledType',
  options: [
    // labelDefault
    { labelDefault: '', value: 'prawns', disabled: true }, // diabledtrue
    { labelDefault: '', value: 'fish' }
  ],
  tooltip: 'type: select', // html
  // event: 'selectDisabledTypeEvent', // handle-event(event, form[prop])
  // filterable: true, //
  clearable: true // handle-clear(event, prop)
  // disabled: true, //
  // class: 'text-type-class', // <style></style>classel-input,
  // style: 'width: 60%' // el-input
},
{
  type: 'select',
  label: 'select()',
  prop: 'selectCountry',
  tooltip: 'type: select', // html
  event: 'selectCountryEvent',
  filterable: true, //
  clearable: true, // handle-clear(event, prop)
```

```
  // { code: 'AD', id: 'AD', label: '', parentId: '0' }
  // keyvaluelabelvalue, value, label
  // optSet
  optSet: { value: 'code', label: 'label' }
  // disabled: true, //
  // class: 'text-type-class', // <style></style>classel-input,
  // style: 'width: 60%' // el-input
},
{
  type: 'select',
  label: 'select()',
  prop: 'selectProvince',
  tooltip: 'type: select', // html
  // event: 'selectProvinceEvent', // handle-event(event, form[prop])
  filterable: true, //
  clearable: true, // handle-clear(event, prop)
  // { code: 'AD', id: 'AD', label: '', parentId: '0' }
  // keyvaluelabelvalue, value, label
  // optSet
  optSet: { value: 'code', label: 'label' }
  // disabled: true, //
  // class: 'text-type-class', // <style></style>classel-input,
  // style: 'width: 60%' // el-input
},
{
  type: 'multipleSelect',
  label: 'multipleSelect',
  prop: 'multipleSelectType',
  tooltip: 'type: multipleSelect', // html
  // event: 'multipleSelectTypeEvent', // handle-event(event, form[prop])
  collapseTags: false, //
  filterable: true, //
  clearable: true, // handle-clear(event, prop)
  // { code: 'AD', id: 'AD', label: '', parentId: '0' }
  // keyvaluelabelvalue, value, label
  // optSet
  optSet: { value: 'code', label: 'label' }
  // disabled: true, //
  // class: 'text-type-class', // <style></style>classel-input,
  // style: 'width: 60%' // el-input
},
{
  type: 'cascader',
  label: 'cascader',
  prop: 'cascaderType',
  tooltip: 'type: cascader', // html
  // event: 'cascaderTypeEvent', // handle-event(event, form[prop])
  filterable: true, //
  clearable: true, // handle-clear(event, prop)
  // { code: 'AD', id: 'AD', label: '', parentId: '0' }
  // keyvaluelabelvalue, value, label
  // optSet
  optSet: { value: 'code', label: 'label', children: 'children' }
  // disabled: true, //
  // class: 'text-type-class', // <style></style>classel-input,
  // style: 'width: 60%' // el-input
},
{
  type: 'citySelect',
  label: 'citySelect',
  prop: 'citySelectType',
  tooltip: 'type: citySelect',
  region: 'CN', // world
  level: 2 // level2
},
{
  type: 'date',
  label: 'date',
  prop: 'dateType',
  tooltip: 'type: date',
  pickerOptions: {
    disabledDate(time) {
      //  return time.getTime() > new Date('2020/05/01')
      //  return time.getTime() > Date.now() - 8.64e7
      //  return time.getTime() > Date.now() - 8.64e7
      // 2020/06/012020/06/30
      return time.getTime() > new Date('2021/06/01') && time.getTime() < new Date('2021/06/31')
    }
  }
  // event: 'dateTypeEvent', // handle-event(event, form[prop])
  // disabled: true, //
  // class: 'text-type-class', // <style></style>classel-input,
  // style: 'width: 60%' // el-input
},
```

```
{
  type: 'datetime',
  label: 'datetime',
  prop: 'datetimeType',
  tooltip: 'type: datetime',
  pickerOptions: {
    disabledDate(time) {
      return time.getTime() > Date.now() - 8.64e7
    }
  }
  // event: 'datetimeTypeEvent', // handle-event(event, form[prop])
  // disabled: true, //
  // class: 'text-type-class', // <style></style>classel-input,
  // style: 'width: 60%' // el-input
},
{
  type: 'daterange',
  label: 'daterange',
  prop: 'daterangeType',
  tooltip: 'type: daterange'
  // pickerOptions: {
  //   disabledDate(time) {
  //     return time.getTime() > Date.now() - 8.64e7
  //   }
  // }
  // rangeSeparator: '', //
  // startPlaceholder: '', //
  // endPlaceholder: '', //
  // event: 'daterangeTypeEvent', // handle-event(event, form[prop])
  // disabled: true, //
  // class: 'text-type-class', // <style></style>classel-input,
  // style: 'width: 60%' // el-input
},
{
  type: 'datetimerange',
  label: 'datetimerange',
  prop: 'datetimerangeType',
  tooltip: 'type: datetimerange'
  // pickerOptions: {
  //   disabledDate(time) {
  //     return time.getTime() > Date.now() - 8.64e7
  //   }
  // }
  // rangeSeparator: '', //
  // startPlaceholder: '', //
  // endPlaceholder: '', //
  // event: 'datetimerangeTypeEvent', // handle-event(event, form[prop])
  // disabled: true, //
  // class: 'text-type-class', // <style></style>classel-input,
  // style: 'width: 60%' // el-input
},
{
  type: 'radio',
  label: 'radio',
  prop: 'radioType',
  tooltip: 'type: radiolgmdsm',
  options: [{ label: '' }, { label: '' }, { label: '' }]
  // options: [{ teaName: '', teaCode: 'blackTea' }, { teaName: '', teaCode: 'flowTea' }],
  // optSetoptSet: { label: '', name: ''}nameteaCodeteaCodeteaName
  // optSet: { label: 'teaCode', name: 'teaName' }
  // event: 'radioTypeEvent', // handle-event(event, form[prop])
  // disabled: true, //
},
{
  type: 'checkbox',
  label: 'checkbox',
  prop: 'checkboxType',
  tooltip: 'type: checkboxlgmdsm',
  options: [{ label: '' }, { label: '' }, { label: '' }]
  // options: [{ foodName: '', foodCode: 'durian' }, { foodName: '', foodCode: 'pineapple' }],
  // optSetoptSet: { label: '', name: ''}namefoodCodefoodCodefoodName
  // optSet: { label: 'foodCode', name: 'foodName' }
  // event: 'checkboxTypeEvent', // handle-event(event, form[prop])
  // disabled: true, //
},
{
  type: 'switch',
  label: 'switch',
  prop: 'switchType',
  tooltip: 'type: switch',
  // disabled: true, // ,
  activeText: '', //
  inactiveText: '', //
  activeValue: true, // 1
```

```
  inactiveValue: false // 0
  // event: 'switchTypeEvent', // handle-event(event, form[prop])
},
{
  type: 'passProgress',
  label: 'passProgress',
  prop: 'passProgressType',
  rules: { required: true, type: 'passwordStreflgth' }, // rulerequiredtype
  tooltip:
    'type: passProgress', // html
  showPassword: true //
  // event: 'passProgressTypeEvent' // handle-event(event, form[prop])
  // placeholder: 'text', //
  // disabled: true, //
  // class: 'text-type-class', // <style></style>classel-input,
  // style: 'width: 60%' // el-input
},
{
  type: 'slot', // slot
  label: 'slot',
  prop: 'slotType',
  rules: { required: true }
},
{
  type: 'textarea',
  label: 'textarea',
  prop: 'textareaType',
  tooltip: 'type: textarealgmdsm',
  lg: 24, // 12001920241281/361/4
  md: 24, // 9921200241281/361/4
  sm: 24 // 768992241281/361/4
  // event: 'textareaTypeEvent' // handle-event(event, form[prop])
  // disabled: true, //
},
{
  type: 'upload',
  label: 'upload()',
  prop: 'uploadIconType',
  rules: { required: true }, // rulerequiredtype
  // storageType: 'minio', // minio/ali-oss/local()minio
  multiple: true, // false
  // limit: 5, // 3
  // badgeHidden: false, // limitfalse
  // fileType: 'all', // all
  // event: 'uploadIconTypeEvent' // handle-event(event, form[prop])
  // disabled: true, //
  listType: 'text', // text/picture/picture-card, text
  bizDir: 'SvFormDemo', //
  buttonStyle: 'icon', // , button/icon
  tooltip: 'type: upload5lgmdsm',
  lg: 24, // 12001920241281/361/4
  md: 24, // 9921200241281/361/4
  sm: 24 // 768992241281/361/4
},
{
  type: 'upload',
  label: 'upload()',
  prop: 'uploadSingleType',
  rules: { required: true }, // rulerequiredtype
  // storageType: 'minio', // minio/ali-oss/local()minio
  // multiple: true, // false
  // limit: 5, // 3
  // badgeHidden: false, // limitfalse
  // fileType: 'all', // all
  // event: 'uploadSingleTypeEvent' // handle-event(event, form[prop])
  // disabled: true, //
  listType: 'text', // text/picture/picture-card, text
  bizDir: 'SvFormDemo', //
  // buttonStyle: 'icon', // , button/icon
  tooltip: 'type: uploadlgmdsm',
  lg: 24, // 12001920241281/361/4
  md: 24, // 9921200241281/361/4
  sm: 24 // 768992241281/361/4
},
{
  type: 'upload',
  label: 'upload()',
  prop: 'uploadMultiType',
  rules: { required: true }, // rulerequiredtype
  // storageType: 'minio', // minio/ali-oss/local()minio
  multiple: true, // false
  limit: 5, // 3
  badgeHidden: false, // limitfalse
  // fileType: 'all', // all
```

```
        // event: 'uploadMultiTypeEvent' // handle-event(event, form[prop])
        // disabled: true, //
        listType: 'text', // text/picture/picture-card, text
        bizDir: 'SvFormDemo', //
        // buttonStyle: 'icon', // , button/icon
        tooltip: 'type: upload5lgmdsm',
        lg: 24, // 12001920241281/361/4
        md: 24, // 9921200241281/361/4
        sm: 24 // 768992241281/361/4
      },
      {
        type: 'upload',
        label: 'upload()',
        prop: 'uploadPictureType',
        rules: { required: true }, // rulerequiredtype
        // storageType: 'minio', // minio/ali-oss/local()minio
        multiple: true, // false
        // limit: 5, // 3
        // badgeHidden: true, // limit
        // fileType: 'all', // all
        // event: 'uploadPictureTypeEvent' // handle-event(event, form[prop])
        // disabled: true, //
        listType: 'picture-card', // text/picture/picture-card, text
        bizDir: 'SvFormDemo', //
        buttonStyle: 'icon', // , button/icon
        tooltip: 'type: uploadlgmdsm',
        lg: 24, // 12001920241281/361/4
        md: 24, // 9921200241281/361/4
        sm: 24 // 768992241281/361/4
      }
    ],
    countryOptions: [],
    dialogUserVisible: false
  }
},
props: {},
computed: {},
watch: {},
created() {
  this.initData()
},
methods: {
  //
  async initData() {
    // fieldList
    const response = await getZipcode()
    this.countryOptions = response.data
    // selectsetfeildList
    this.$set(this.fieldList.find(item => item.prop === 'selectCountry'), 'options', this.countryOptions)
    this.$set(this.fieldList.find(item => item.prop === 'multipleSelectType'), 'options', this.countryOptions)
    this.$set(this.fieldList.find(item => item.prop === 'cascaderType'), 'options', this.countryOptions)
  },
  // singlePopup
  handleClick(event) {
    if (event === 'save') {
      this.save()
    } else if (event === 'singlePopupEvent') {
      this.dialogUserVisible = true
    }
  },
  //
  handleClose(data) {
    this.dialogUserVisible = false
    this.form.singlePopupType = data.name
    this.form.username = data.username
  },
  // singlePopup
  handleClear(event) {
    if (event === 'singlePopupEvent') {
      this.form.singlePopupType = undefined
      this.form.username = undefined
    }
  },
  // change
  handleEvent(event, data) {
    if (event === 'selectTypeEvent') {
      this.selectEvent(data)
    } else if (event === 'selectCountryEvent') {
      this.selectCountryEvent(data)
    }
  },
  //
  selectEvent(data) {
    if (data === 'Y') {
```

```
        this.form.relateText = ''
      } else if (data === 'N') {
        this.form.relateText = ''
      } else {
        this.form.relateText = ''
      }
    },
    //
    selectCountryEvent(data) {
      if (data) {
        this.form.selectProvince = undefined
        const country = this.countryOptions.find(item => item.code === data)
        const provinceOptions = country.children ? country.children : []
        this.$set(this.fieldList.find(item => item.prop === 'selectProvince'), 'options', provinceOptions)
      } else {
        this.form.selectProvince = undefined
        this.$set(this.fieldList.find(item => item.prop === 'selectProvince'), 'options', [])
      }
    },
    // SvFormvalidate
    async save() {
      // this.$refs.SvForm.validate()
      if (!this.$refs.SvForm.validate()) return
      //
      // this.form.multipleSelectType = this.form.multipleSelectType.join(',')
      console.log('this.form', this.form)
      // const response = await save(this.form)
      // if (response && response.status === 200) {
      //   this.$notify({
      //     title: '',
      //     message: '',
      //     type: 'success',
      //     duration: 2000
      //   })
      // }
    }
  }
}
</script>

<style lang="scss" scoped>
::v-deep .text-type-class .el-input__inner {
  background-color: pink;
}
</style>
```