JDate 日期组件 使用文档

说明: antd-vue日期组件需要用moment中转一下，用起来不是很方便，特二次封装，使用时只需要传字符串即可

参数配置

参数	类型	必填	说明
placeholder	string		placeholder
readOnly	boolean		true/false 默认false
value	string		绑定v-model或是v-decorator后不需要设置
showTime	boolean		是否展示时间true/false 默认false
dateFormat	string		日期格式 默认'YYYY-MM-DD' 若showTime设置为true则需要将其设置成对应的时间格式(如:YYYY-MM-DD HH:mm:ss)
triggerChange	string		触发组件值改变的事件是否是change,当使用v-decorator时且没有设置decorator的option.trigger为input需要设置该值为true

使用示例

1.组件带有v-model的使用方法

```
<j-date v-model="dateStr"></j-date>
```

2.组件带有v-decorator的使用方法
a).设置trigger-change属性为true

```
  <j-date :trigger-change="true" v-decorator="['dateStr',{}]"></j-date>
```

b).设置decorator的option.trigger为input

```
 <j-date v-decorator="['dateStr',{trigger:'input'}]"></j-date>
```

3.其他使用 添加style

```
<j-date v-model="dateStr" style="width:100%"></j-date>
```

添加placeholder

```
<j-date v-model="dateStr" placeholder="请输入dateStr"></j-date>
```

添加readOnly

```
<j-date v-model="dateStr" :read-only="true"></j-date>
```

备注: script内需引入jdate

```
<script>
  import JDate from '@/components/jeecg/JDate'
  export default {
    name: "demo",
    components: {
      JDate
    }
    //...
  }
</script>
```

JSuperQuery 高级查询 使用文档

参数配置

参数	类型	必填	说明
fieldList	array	✔	需要查询的列集合示例如下，type类型有:date/datetime/string/int/number
callback	array		回调函数名称(非必须)默认handleSuperQuery

fieldList结构示例：

```
  const superQueryFieldList=[{
    type:"date",
    value:"birthday",
    text:"生日"
  },{
    type:"string",
    value:"name",
    text:"用户名"
  },{
    type:"int",
    value:"age",
```

```
    text:"年龄"
  }]
```

页面代码概述:

1.import之后再components之内声明

```
import JSuperQuery from '@/components/jeecg/JSuperQuery.vue';
  export default {
    name: "JeecgDemoList",
    components: {
      JSuperQuery
    },
```

2.页面引用

```
  <!-- 高级查询区域 -->
  <j-super-query :fieldList="fieldList" ref="superQueryModal"
@handleSuperQuery="handleSuperQuery"></j-super-query>
```

3.list页面data中需要定义三个属性：

```
  fieldList:superQueryFieldList,
  superQueryFlag:false,
  superQueryParams:""
```

4.list页面声明回调事件handleSuperQuery(与组件的callback对应即可)

```
//高级查询方法
handleSuperQuery(arg) {
  if(!arg){
    this.superQueryParams=''
    this.superQueryFlag = false
  }else{
    this.superQueryFlag = true
    this.superQueryParams=JSON.stringify(arg)
  }
  this.loadData()
},
```

5.改造list页面方法

```
  // 获取查询条件
  getQueryParams() {
    let sqp = {}
    if(this.superQueryParams){
      sqp['superQueryParams']=encodeURI(this.superQueryParams)
    }
    var param = Object.assign(sqp, this.queryParam, this.isorter);
    param.field = this.getQueryField();
    param.pageNo = this.ipagination.current;
    param.pageSize = this.ipagination.pageSize;
    return filterObj(param);
  },
```

6.打开弹框调用show方法：

```
this.$refs.superQueryModal.show();
```

JEllipsis 字符串超长截取省略号显示

说明: 遇到超长文本展示，通过此标签可以截取省略号显示，鼠标放置会提示全文本

参数配置

参数	类型	必填	说明
value	string	必填	字符串文本
length	number	非必填	默认25

使用示例

1.组件带有v-model的使用方法

```
<j-ellipsis :value="text"/>


# Modal弹框实现最大化功能

1.定义modal的宽度：
```vue
  <a-modal
    :width="modalWidth"


    />
```
```

2.自定义modal的title,居右显示切换图标

```
  <template slot="title">
    <div style="width: 100%;">
      <span>{{ title }}</span>
      <span style="display:inline-block;width:calc(100% - 51px);padding-
right:10px;text-align: right">
        <a-button @click="toggleScreen" icon="appstore"
style="height:20px;width:20px;border:0px"></a-button>
      </span>
    </div>
  </template>
```

3.定义toggleScreen事件,用于切换modal宽度

```
  toggleScreen(){
      if(this.modaltoggleFlag){
        this.modalWidth = window.innerWidth;
      }else{
        this.modalWidth = 800;
      }
      this.modaltoggleFlag = !this.modaltoggleFlag;
    },
```

4.data中声明上述用到的属性

```
    data () {
      return {
        modalWidth:800,
        modaltoggleFlag:true,
```

下拉选项滚动错位的解决方法

问题描述

当使用了 `a-modal` 或其他带有滚动条的组件时，使用`a-select`组件并打开下拉框时滚动滚动条，就会导致错位的问题产生。

解决方法

大多数情况下，在 `a-select` 上添加一个 `getPopupContainer` 属性，值为`node => node.parentNode`即可解决。 但是如果遇到 `a-select` 标签层级过深的情况，可能仍然会显示异常，只需要多加几个`.parentNode`（例：node => node.parentNode.parentNode.parentNode）多尝试几次直到解决问题即可。

代码示例

```
<a-select
    placeholder="请选择展示模板"
    :options="dicts.displayTemplate"
    :getPopupContainer="node => node.parentNode"
/>
```

JAsyncTreeList 异步数列表组件使用说明

引入组件

```
import JTreeTable from '@/components/jeecg/JTreeTable'
export default {
  components: { JTreeTable }
}
```

所需参数

参数	类型	必填	说明
rowKey	String	非必填	表格行 key 的取值，默认为"id"
columns	Array	必填	表格列的配置描述，具体见Antd官方文档
url	String	必填	数据查询url
childrenUrl	String	非必填	查询子级时的url，若不填则使用url参数查询子级
queryKey	String	非必填	根据某个字段查询，如果传递 id 就根据 id 查询，默认为parentId
queryParams	Object	非必填	查询参数，当查询参数改变的时候会自动重新查询，默认为{}
topValue	String	非必填	查询顶级时的值，如果顶级为0，则传0，默认为null
tableProps	Object	非必填	自定义给内部table绑定的props

代码示例

```
<template>
  <a-card :bordered="false">
    <j-tree-table :url="url" :columns="columns" :tableProps="tableProps"/>
  </a-card>
</template>

<script>
  import JTreeTable from '@/components/jeecg/JTreeTable'

  export default {
    name: 'AsyncTreeTable',
```

```
    components: { JTreeTable },
    data() {
      return {
        url: '/api/asynTreeList',
        columns: [
          { title: '菜单名称', dataIndex: 'name' },
          { title: '组件', dataIndex: 'component' },
          { title: '排序', dataIndex: 'orderNum' }
        ],
        selectedRowKeys: []
      }
    },
     computed: {
       tableProps() {
         let _this = this
         return {
           // 列表项是否可选择
           // 配置项见：https://vue.ant.design/components/table-cn/#rowSelection
           rowSelection: {
             selectedRowKeys: _this.selectedRowKeys,
             onChange: (selectedRowKeys) => _this.selectedRowKeys =
selectedRowKeys
           }
         }
       }
     }
  }
</script>
```

JCheckbox 使用文档

说明: antd-vue checkbox组件处理的是数组，用起来不是很方便，特二次封装，使用时只需处理字符串即可

参数配置

参数	类型	必填	说明
options	array	✔	checkbox需要配置的项，是个数组，数组中每个对象包含两个属性:label(用于显示)和value(用于存储)

使用示例

```
<template>
  <a-form :form="form">
    <a-form-item label="v-model式用法">
      <j-checkbox v-model="sport" :options="sportOptions"></j-checkbox><span>{{
sport }}</span>
    </a-form-item>
```

```
    <a-form-item label="v-decorator式用法">
      <j-checkbox v-decorator="['sport']" :options="sportOptions"></j-checkbox>
<span>{{ getFormFieldValue('sport') }}</span>
    </a-form-item>
  </a-form>
</template>

<script>
  import JCheckbox from '@/components/jeecg/JCheckbox'
  export default {
    components: {JCheckbox},
    data() {
      return {
        form: this.$form.createForm(this),
        sport:'',
        sportOptions:[
          {
            label:"足球",
            value:"1"
          },{
            label:"篮球",
            value:"2"
          },{
            label:"乒乓球",
            value:"3"
          }]
      }
    },
    methods: {
     getFormFieldValue(field){
       return this.form.getFieldValue(field)
     }
    }
  }
</script>
```

JCodeEditor 使用文档

说明: 一个简易版的代码编辑器，支持语法高亮

参数配置

参数	类型	必填	说明
language	string		表示当前编写代码的类型 javascript/html/css/sql
placeholder	string		placeholder
lineNumbers	Boolean		是否显示行号

参数	类型	必填	说明
fullScreen	Boolean		是否显示全屏按钮
zIndex	string		全屏以后的z-index

使用示例

```
<template>
  <div>
    <j-code-editor
      language="javascript"
      v-model="editorValue"
      :fullScreen="true"
      style="min-height: 100px"/>
    {{ editorValue }}
  </div>
</template>

<script>
  import JCodeEditor from '@/components/jeecg/JCodeEditor'
  export default {
    components: {JCodeEditor},
    data() {
      return {
        form: this.$form.createForm(this),
        editorValue:'',
      }
    }
  }
</script>
```

JFormContainer 使用文档

说明: 暂用于表单禁用

使用示例

```
<!-- 在form下直接写这个组件，设置disabled为true就能将此form中的控件禁用 -->
  <a-form layout="inline" :form="form" >
    <j-form-container disabled>
      <!-- 表单内容省略..... -->
    </j-form-container>
  </a-form>
```

JImportModal 使用文档

说明: 用于列表页面导入excel功能

使用示例

```
<template>
  <!--  此处省略部分代码...... -->
  <a-button @click="handleImportXls" type="primary" icon="upload">导入</a-button>
  <!--  此处省略部分代码...... -->
  <j-import-modal ref="importModal" :url="getImportUrl()" @ok="importOk"></j-
import-modal>
  <!--  此处省略部分代码...... -->
</template>

<script>
  import JCodeEditor from '@/components/jeecg/JCodeEditor'
  export default {
    components: {JCodeEditor},
    data() {
      return {
        //省略代码......
      }
    },
    methods:{
      //省略部分代码......
      handleImportXls(){
        this.$refs.importModal.show()
      },
      getImportUrl(){
         return '你自己处理上传业务的后台地址'
      },
      importOk(){
        this.loadData(1)
      }
    }
  }
</script>
```

JSelectMultiple 多选下拉组件件

online用 实际开发请使用components/dict/JMultiSelectTag

JSlider 滑块验证码

使用示例

```
<template>
  <div style="width: 300px">
    <j-slider @onSuccess="sliderSuccess"></j-slider>
  </div>
</template>

<script>
  import JSlider from '@/components/jeecg/JSlider'
  export default {
    components: {JSlider},
    data() {
      return {
        form: this.$form.createForm(this),
        editorValue:'',
      }
    },
    methods:{
      sliderSuccess(){
        console.log("验证完成")
      }
    }
  }
</script>
```

JTreeSelect 树形下拉组件

异步加载的树形下拉组件

参数配置

参数	类型	必填	说明
placeholder	string		placeholder
dict	string	✔	表名,显示字段名,存储字段名拼接的字符串
pidField	string	✔	父ID的字段名
pidValue	string		根节点父ID的值 默认'0' 不可以设置为空,如果想使用此组件，而数据库根节点父ID为空，请修改之
multiple	boolean		是否支持多选

使用示例

```
<template>
  <a-form>
    <a-form-item label="树形下拉测试" style="width: 300px">
```

```
      <j-tree-select
        v-model="departId"
        placeholder="请选择部门"
        dict="sys_depart,depart_name,id"
        pidField="parent_id">
      </j-tree-select>
      {{ departId }}
    </a-form-item>
  </a-form >
</template>

<script>
  import JTreeSelect from '@/components/jeecg/JTreeSelect'
  export default {
    components: {JTreeSelect},
    data() {
      return {
        departId:""
      }
    }
  }
</script>
```