table实现

思路分析

KTable-header: table头部组组成部分

KTable-body: 根据KTable-column返回内容进行table主题部分渲染

KTable-column: 生成column，并为column绑定`renderCell函数`供table-body使用

测试code

```
<template>
  <div id="app">
    <k-table :data="tableData">
      <k-table-column prop="date" label="日期" sortable> </k-table-column>
      <k-table-column prop="name" label="姓名"> </k-table-column>
      <k-table-column prop="address" label="地址"> </k-table-column>
      <k-table-column label="操作">
        <template slot-scope="scope">
          {{ scope.row.operation }}
        </template>
      </k-table-column>
    </k-table>
  </div>
</template>

<script>
import KTable from './components/table/KTable'
import KTableColumn from './components/table/KTable-column.js'

export default {
  name: 'App',
  components: {
    KTable,
    KTableColumn,
  },
  data() {
    return {
      tableData: [
        {
          date: '2016-05-02',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄',
          operation: '添加'
        },
```

```
        {
          date: '2016-05-04',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1517 弄',
          operation: '删除'
        },
        {
          date: '2016-05-01',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1519 弄',
          operation: '修改'
        },
        {
          date: '2016-05-03',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1516 弄',
          operation: '查找'
        },
      ],
    }
  },
}
</script>
```

KTable.vue

```
<template>
  <div>
    <slot></slot>
    <k-table-header :columns="columns" :data="data"></k-table-header>
    <table ref="table">
      <k-table-body :columns="columns" :data="data"></k-table-body>
    </table>
  </div>
</template>

<script>
import KTableHeader from './KTable-header.vue'
import KTableBody from './KTable-body.js'
export default {
  props: {
    data: Array,
  },
  data() {
    return {
      columns: [],
    }
```

```
  },
  components: {
    KTableHeader,
    KTableBody,
  },
}
</script>
```

KTable-header.vue

```
<template>
  <div>
    <table>
      <thead>
        <th v-for="(column, index) in columns" :key="column.label + index">
          {{ column.label }}
          <em v-if="column.sortable">
            <i @click="descending(column)">降序</i> ||
            <i @click="ascending(column)">升序</i>
          </em>
        </th>
      </thead>
    </table>
  </div>
</template>

<script>
export default {
  name: 'KTableHeader',
  props: {
    columns: Array,
    data: Array,
  },
  methods: {
    compareUp(data, propertyName) {
      // 升序排序
      if (typeof data[0][propertyName] != 'number') {
        // 属性值为非数字
        return function (object1, object2) {
          var value1 = object1[propertyName]
          var value2 = object2[propertyName]
          return value1.localeCompare(value2)
        }
      } else {
        return function (object1, object2) {
          // 属性值为数字
          var value1 = object1[propertyName]
```

```
            var value2 = object2[propertyName]
            return value1 - value2
          }
        }
      },
      compareDown(data, propertyName) {
        // 降序排序
        if (typeof data[0][propertyName] != 'number') {
          // 属性值为非数字
          return function (object1, object2) {
            var value1 = object1[propertyName]
            var value2 = object2[propertyName]
            return value2.localeCompare(value1)
          }
        } else {
          return function (object1, object2) {
            // 属性值为数字
            var value1 = object1[propertyName]
            var value2 = object2[propertyName]
            return value2 - value1
          }
        }
      },
      descending(column) {
        this.data.sort(this.compareDown(this.data, column.prop))
      },

      ascending(column) {
        this.data.sort(this.compareUp(this.data, column.prop))
      },
    },
}
</script>
```

KTable-column.js

```
export default {
  name: 'KTableColumn',
  componentName: 'KTableColumn',
  props: {
    prop: String,
    label: String,
    sortable: Boolean,
  },
  data() {
    return {
      column: {},
```

```
    }
  },
  render() {},
  watch: {
    prop(val) {
      this.column.prop = val
    },
    label(val) {
      this.column.lable = val
    },
    sortable(val) {
      this.column.sortable = val
    },
  },
  created() {
    let column = {
      ...this.$props,
    }
    let self = this
    column.renderCell = function (h, data) {
      let render = (h, data) => {
        return data.row[column.prop]
      }

      //    自定义列模板
      if (self.$scopedSlots.default) {
        render = () => self.$scopedSlots.default(data)
      }

      // 返回渲染内容
      return <div>{render(h, data)}</div>
    }

    this.column = column
  },
  mounted() {
    this.$parent.columns.push(this.column)
  },
}
```

KTable-body.js

```
export default {
  name: 'KTableBody',
  props: {
```

```
    columns: {
      type: Array,
      dafault: [],
    },
    data: {
      type: Array,
      dafault: [],
    },
  },
  render(h) {
    return (
      <tbody>
        {/* this._l类似于v-for遍历,vue内置 */}
        {this._l(this.data, (row, rindex) => {
          return (
            <tr>
              {this._l(this.columns, (column, cindex) => {
                return <td>{column.renderCell(h, { row, rindex })}</td>
              })}
            </tr>
          )
        })}
      </tbody>
    )
  },
}
```

table作业要求

1.仿照element-ui table组价实现KTable表格的基本展示

使用形式如下：

```
<k-table :data="tableData">
  <k-table-column prop="date" label="日期" sortable> </k-table-column>
</k-table>
```

展示效果图（css对齐不限制）

2：可以自定义列模板

形式如下

```
<k-table-column label="操作">
   <template slot-scope="scope"> {{ scope.row.operation }} </template>
 </k-table-column>
```

3.实现表格排序方法

要求：在k-table-column中传入sortable参数即可在该列表头出现升序和降序排序方法，进行相应的排序

效果如下（css不做限制）：

针对表格编写一个测试用例