Table 表格

用于展示多条结构类似的数据，可对数据进行排序、筛选、对比或其他自定义操作。

基础表格

基础的表格展示用法。

:::demo 当 `el-table` 元素中注入 `data` 对象数组后，在 `el-table-column` 中用 `prop` 属性来对应对象中的键名即可填入数据，用 `label` 属性来定义表格的列名。可以使用 `width` 属性来定义列宽。

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%">
    <el-table-column
      prop="date"
      label="日期"
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="姓名"
      width="180">
    </el-table-column>
    <el-table-column
      prop="address"
      label="地址">
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-02',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-04',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1517 弄'
        }, {
          date: '2016-05-01',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1519 弄'
        }, {
          date: '2016-05-03',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1516 弄'
```

```
          }]
        }
      }
    }
  </script>
```

:::

带斑马纹表格

使用带斑马纹的表格，可以更容易区分出不同行的数据。

:::demo `stripe` 属性可以创建带斑马纹的表格。它接受一个 `Boolean`，默认为 `false`，设置为 `true` 即为启用。

```
<template>
  <el-table
    :data="tableData"
    stripe
    style="width: 100%">
    <el-table-column
      prop="date"
      label="日期"
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="姓名"
      width="180">
    </el-table-column>
    <el-table-column
      prop="address"
      label="地址">
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-02',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-04',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1517 弄'
        }, {
          date: '2016-05-01',
          name: '王小虎',
```

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

:::

带边框表格

:::demo 默认情况下，Table 组件是不具有竖直方向的边框的，如果需要，可以使用 `border` 属性，它接受一个 `Boolean`，设置为 `true` 即可启用。

```
<template>
  <el-table
    :data="tableData"
    border
    style="width: 100%">
    <el-table-column
      prop="date"
      label="日期"
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="姓名"
      width="180">
    </el-table-column>
    <el-table-column
      prop="address"
      label="地址">
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-02',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-04',
          name: '王小虎',
```

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

:::

带状态表格

可将表格内容 highlight 显示，方便区分「成功、信息、警告、危险」等内容。

:::demo 可以通过指定 Table 组件的 `row-class-name` 属性来为 Table 中的某一行添加 class，表明该行处于某种状态。

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%"
    :row-class-name="tableRowClassName">
    <el-table-column
      prop="date"
      label="日期"
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="姓名"
      width="180">
    </el-table-column>
    <el-table-column
      prop="address"
      label="地址">
    </el-table-column>
  </el-table>
</template>

<style>
  .el-table .warning-row {
    background: oldlace;
  }

  .el-table .success-row {
```

```
    background: #f0f9eb;
  }
</style>

<script>
  export default {
    methods: {
      tableRowClassName({row, rowIndex}) {
        if (rowIndex === 1) {
          return 'warning-row';
        } else if (rowIndex === 3) {
          return 'success-row';
        }
        return '';
      }
    },
    data() {
      return {
        tableData: [{
          date: '2016-05-02',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄',
        }, {
          date: '2016-05-04',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-01',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄',
        }, {
          date: '2016-05-03',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }]
      }
    }
  }
</script>
```

:::

固定表头

纵向内容过多时，可选择固定表头。

:::demo 只要在 `el-table` 元素中定义了 `height` 属性，即可实现固定表头的表格，而不需要额外的代码。

```
<template>
  <el-table
    :data="tableData"
    height="250"
```

```
    border
    style="width: 100%">
    <el-table-column
      prop="date"
      label="日期"
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="姓名"
      width="180">
    </el-table-column>
    <el-table-column
      prop="address"
      label="地址">
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-03',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-02',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-04',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-01',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-08',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-06',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-07',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }]
```

```
      }
    }
  }
</script>
```

:::

固定列

横向内容过多时，可选择固定列。

:::demo 固定列需要使用 `fixed` 属性，它接受 Boolean 值或者 `left` `right`，表示左边固定还是右边固定。

```
<template>
  <el-table
    :data="tableData"
    border
    style="width: 100%">
    <el-table-column
      fixed
      prop="date"
      label="日期"
      width="150">
    </el-table-column>
    <el-table-column
      prop="name"
      label="姓名"
      width="120">
    </el-table-column>
    <el-table-column
      prop="province"
      label="省份"
      width="120">
    </el-table-column>
    <el-table-column
      prop="city"
      label="市区"
      width="120">
    </el-table-column>
    <el-table-column
      prop="address"
      label="地址"
      width="300">
    </el-table-column>
    <el-table-column
      prop="zip"
      label="邮编"
      width="120">
    </el-table-column>
    <el-table-column
      fixed="right"
      label="操作"
```

```
      width="100">
      <template slot-scope="scope">
        <el-button @click="handleClick(scope.row)" type="text" size="small">查看</el-
button>
        <el-button type="text" size="small">编辑</el-button>
      </template>
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    methods: {
      handleClick(row) {
        console.log(row);
      }
    },

    data() {
      return {
        tableData: [{
          date: '2016-05-02',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1518 弄',
          zip: 200333
        }, {
          date: '2016-05-04',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1517 弄',
          zip: 200333
        }, {
          date: '2016-05-01',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1519 弄',
          zip: 200333
        }, {
          date: '2016-05-03',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1516 弄',
          zip: 200333
        }]
      }
    }
```

```
  }
</script>
```

:::

固定列和表头

横纵内容过多时，可选择固定列和表头。

:::demo 固定列和表头可以同时使用，只需要将上述两个属性分别设置好即可。

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%"
    height="250">
    <el-table-column
      fixed
      prop="date"
      label="日期"
      width="150">
    </el-table-column>
    <el-table-column
      prop="name"
      label="姓名"
      width="120">
    </el-table-column>
    <el-table-column
      prop="province"
      label="省份"
      width="120">
    </el-table-column>
    <el-table-column
      prop="city"
      label="市区"
      width="120">
    </el-table-column>
    <el-table-column
      prop="address"
      label="地址"
      width="300">
    </el-table-column>
    <el-table-column
      prop="zip"
      label="邮编"
      width="120">
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
```

```
data() {
  return {
    tableData: [{
      date: '2016-05-03',
      name: '王小虎',
      province: '上海',
      city: '普陀区',
      address: '上海市普陀区金沙江路 1518 弄',
      zip: 200333
    }, {
      date: '2016-05-02',
      name: '王小虎',
      province: '上海',
      city: '普陀区',
      address: '上海市普陀区金沙江路 1518 弄',
      zip: 200333
    }, {
      date: '2016-05-04',
      name: '王小虎',
      province: '上海',
      city: '普陀区',
      address: '上海市普陀区金沙江路 1518 弄',
      zip: 200333
    }, {
      date: '2016-05-01',
      name: '王小虎',
      province: '上海',
      city: '普陀区',
      address: '上海市普陀区金沙江路 1518 弄',
      zip: 200333
    }, {
      date: '2016-05-08',
      name: '王小虎',
      province: '上海',
      city: '普陀区',
      address: '上海市普陀区金沙江路 1518 弄',
      zip: 200333
    }, {
      date: '2016-05-06',
      name: '王小虎',
      province: '上海',
      city: '普陀区',
      address: '上海市普陀区金沙江路 1518 弄',
      zip: 200333
    }, {
      date: '2016-05-07',
      name: '王小虎',
      province: '上海',
      city: '普陀区',
      address: '上海市普陀区金沙江路 1518 弄',
      zip: 200333
    }]
```

```
      }
    }
  }
</script>
```

:::

流体高度

当数据量动态变化时，可以为 Table 设置一个最大高度。

:::demo 通过设置 `max-height` 属性为 Table 指定最大高度。此时若表格所需的高度大于最大高度，则会显示一个滚动条。

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%"
    max-height="250">
    <el-table-column
      fixed
      prop="date"
      label="日期"
      width="150">
    </el-table-column>
    <el-table-column
      prop="name"
      label="姓名"
      width="120">
    </el-table-column>
    <el-table-column
      prop="province"
      label="省份"
      width="120">
    </el-table-column>
    <el-table-column
      prop="city"
      label="市区"
      width="120">
    </el-table-column>
    <el-table-column
      prop="address"
      label="地址"
      width="300">
    </el-table-column>
    <el-table-column
      prop="zip"
      label="邮编"
      width="120">
    </el-table-column>
    <el-table-column
      fixed="right"
```

```
      label="操作"
      width="120">
      <template slot-scope="scope">
        <el-button
          @click.native.prevent="deleteRow(scope.$index, tableData)"
          type="text"
          size="small">
          移除
        </el-button>
      </template>
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    methods: {
      deleteRow(index, rows) {
        rows.splice(index, 1);
      }
    },
    data() {
      return {
        tableData: [{
          date: '2016-05-03',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1518 弄',
          zip: 200333
        }, {
          date: '2016-05-02',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1518 弄',
          zip: 200333
        }, {
          date: '2016-05-04',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1518 弄',
          zip: 200333
        }, {
          date: '2016-05-01',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1518 弄',
          zip: 200333
        }, {
```

```
          date: '2016-05-08',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1518 弄',
          zip: 200333
        }, {
          date: '2016-05-06',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1518 弄',
          zip: 200333
        }, {
          date: '2016-05-07',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1518 弄',
          zip: 200333
        }]
      }
    }
  }
</script>
```

:::

多级表头

数据结构比较复杂的时候，可使用多级表头来展现数据的层次关系。

:::demo 只需要在 el-table-column 里面嵌套 el-table-column，就可以实现多级表头。

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%">
    <el-table-column
      prop="date"
      label="日期"
      width="150">
    </el-table-column>
    <el-table-column label="配送信息">
      <el-table-column
        prop="name"
        label="姓名"
        width="120">
      </el-table-column>
      <el-table-column label="地址">
        <el-table-column
          prop="province"
```

```
          label="省份"
          width="120">
        </el-table-column>
        <el-table-column
          prop="city"
          label="市区"
          width="120">
        </el-table-column>
        <el-table-column
          prop="address"
          label="地址"
          width="300">
        </el-table-column>
        <el-table-column
          prop="zip"
          label="邮编"
          width="120">
        </el-table-column>
      </el-table-column>
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-03',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1518 弄',
          zip: 200333
        }, {
          date: '2016-05-02',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1518 弄',
          zip: 200333
        }, {
          date: '2016-05-04',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1518 弄',
          zip: 200333
        }, {
          date: '2016-05-01',
          name: '王小虎',
          province: '上海',
```

```
          city: '普陀区',
          address: '上海市普陀区金沙江路 1518 弄',
          zip: 200333
        }, {
          date: '2016-05-08',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1518 弄',
          zip: 200333
        }, {
          date: '2016-05-06',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1518 弄',
          zip: 200333
        }, {
          date: '2016-05-07',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1518 弄',
          zip: 200333
        }]
      }
    }
  }
</script>
```

:::

单选

选择单行数据时使用色块表示。

:::demo Table 组件提供了单选的支持，只需要配置 `highlight-current-row` 属性即可实现单选。之后由 `current-change` 事件来管理选中时触发的事件，它会传入 `currentRow`， `oldCurrentRow`。如果需要显示索引，可以增加一列 `el-table-column`，设置 `type` 属性为 `index` 即可显示从 1 开始的索引号。

```
<template>
  <el-table
    ref="singleTable"
    :data="tableData"
    highlight-current-row
    @current-change="handleCurrentChange"
    style="width: 100%">
    <el-table-column
      type="index"
      width="50">
    </el-table-column>
    <el-table-column
```

```
      property="date"
      label="日期"
      width="120">
    </el-table-column>
    <el-table-column
      property="name"
      label="姓名"
      width="120">
    </el-table-column>
    <el-table-column
      property="address"
      label="地址">
    </el-table-column>
  </el-table>
  <div style="margin-top: 20px">
    <el-button @click="setCurrent(tableData[1])">选中第二行</el-button>
    <el-button @click="setCurrent()">取消选择</el-button>
  </div>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-02',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-04',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1517 弄'
        }, {
          date: '2016-05-01',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1519 弄'
        }, {
          date: '2016-05-03',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1516 弄'
        }],
        currentRow: null
      }
    },

    methods: {
      setCurrent(row) {
        this.$refs.singleTable.setCurrentRow(row);
      },
      handleCurrentChange(val) {
        this.currentRow = val;
      }
```

```
    }
  }
</script>
```

:::

多选

选择多行数据时使用 Checkbox。

:::demo 实现多选非常简单: 手动添加一个 `el-table-column`，设 `type` 属性为 `selection` 即可；默认情况下若内容过多会折行显示，若需要单行显示可以使用 `show-overflow-tooltip` 属性，它接受一个 `Boolean`，为 `true` 时多余的内容会在 hover 时以 tooltip 的形式显示出来。

```
<template>
  <el-table
    ref="multipleTable"
    :data="tableData"
    tooltip-effect="dark"
    style="width: 100%"
    @selection-change="handleSelectionChange">
    <el-table-column
      type="selection"
      width="55">
    </el-table-column>
    <el-table-column
      label="日期"
      width="120">
      <template slot-scope="scope">{{ scope.row.date }}</template>
    </el-table-column>
    <el-table-column
      prop="name"
      label="姓名"
      width="120">
    </el-table-column>
    <el-table-column
      prop="address"
      label="地址"
      show-overflow-tooltip>
    </el-table-column>
  </el-table>
  <div style="margin-top: 20px">
    <el-button @click="toggleSelection([tableData[1], tableData[2]])">切换第二、第三行的选中状态</el-button>
    <el-button @click="toggleSelection()">取消选择</el-button>
  </div>
</template>

<script>
  export default {
    data() {
      return {
```

```
        tableData: [{
          date: '2016-05-03',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-02',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-04',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-01',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-08',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-06',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-07',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }],
        multipleSelection: []
      }
    },

    methods: {
      toggleSelection(rows) {
        if (rows) {
          rows.forEach(row => {
            this.$refs.multipleTable.toggleRowSelection(row);
          });
        } else {
          this.$refs.multipleTable.clearSelection();
        }
      },
      handleSelectionChange(val) {
        this.multipleSelection = val;
      }
    }
  }
</script>
```

:::

排序

对表格进行排序，可快速查找或对比数据。

:::demo 在列中设置 `sortable` 属性即可实现以该列为基准的排序，接受一个 `Boolean`，默认为 `false`。可以通过 Table 的 `default-sort` 属性设置默认的排序列和排序顺序。可以使用 `sort-method` 或者 `sort-by` 使用自定义的排序规则。如果需要后端排序，需将 `sortable` 设置为 `custom`，同时在 Table 上监听 `sort-change` 事件，在事件回调中可以获取当前排序的字段名和排序顺序，从而向接口请求排序后的表格数据。在本例中，我们还使用了 `formatter` 属性，它用于格式化指定列的值，接受一个 `Function`，会传入两个参数：`row` 和 `column`，可以根据自己的需求进行处理。

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%"
    :default-sort = "{prop: 'date', order: 'descending'}"
    >
    <el-table-column
      prop="date"
      label="日期"
      sortable
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="姓名"
      sortable
      width="180">
    </el-table-column>
    <el-table-column
      prop="address"
      label="地址"
      :formatter="formatter">
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-02',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-04',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1517 弄'
        }, {
          date: '2016-05-01',
```

```
          name: '王小虎',
          address: '上海市普陀区金沙江路 1519 弄'
        }, {
          date: '2016-05-03',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1516 弄'
        }]
      }
    },
    methods: {
      formatter(row, column) {
        return row.address;
      }
    }
  }
</script>
```

:::

筛选

对表格进行筛选，可快速查找到自己想看的数据。

:::demo 在列中设置 `filters` `filter-method` 属性即可开启该列的筛选，filters 是一个数组，`filter-method` 是一个方法，它用于决定某些数据是否显示，会传入三个参数：`value`，`row` 和 `column`。

```
<template>
  <el-button @click="resetDateFilter">清除日期过滤器</el-button>
  <el-button @click="clearFilter">清除所有过滤器</el-button>
  <el-table
    ref="filterTable"
    :data="tableData"
    style="width: 100%">
    <el-table-column
      prop="date"
      label="日期"
      sortable
      width="180"
      column-key="date"
      :filters="[{text: '2016-05-01', value: '2016-05-01'}, {text: '2016-05-02',
value: '2016-05-02'}, {text: '2016-05-03', value: '2016-05-03'}, {text: '2016-05-
04', value: '2016-05-04'}]"
      :filter-method="filterHandler"
    >
    </el-table-column>
    <el-table-column
      prop="name"
      label="姓名"
      width="180">
    </el-table-column>
    <el-table-column
      prop="address"
```

```
      label="地址"
      :formatter="formatter">
    </el-table-column>
    <el-table-column
      prop="tag"
      label="标签"
      width="100"
      :filters="[{ text: '家', value: '家' }, { text: '公司', value: '公司' }]"
      :filter-method="filterTag"
      filter-placement="bottom-end">
      <template slot-scope="scope">
        <el-tag
          :type="scope.row.tag === '家' ? 'primary' : 'success'"
          disable-transitions>{{scope.row.tag}}</el-tag>
      </template>
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-02',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄',
          tag: '家'
        }, {
          date: '2016-05-04',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1517 弄',
          tag: '公司'
        }, {
          date: '2016-05-01',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1519 弄',
          tag: '家'
        }, {
          date: '2016-05-03',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1516 弄',
          tag: '公司'
        }]
      }
    },
    methods: {
      resetDateFilter() {
        this.$refs.filterTable.clearFilter('date');
      },
      clearFilter() {
        this.$refs.filterTable.clearFilter();
```

```
      },
      formatter(row, column) {
        return row.address;
      },
      filterTag(value, row) {
        return row.tag === value;
      },
      filterHandler(value, row, column) {
        const property = column['property'];
        return row[property] === value;
      }
    }
  }
</script>
```

:::

自定义列模板

自定义列的显示内容，可组合其他组件使用。 :::demo 通过 `Scoped slot` 可以获取到 row, column, $index 和 store（table 内部的状态管理）的数据，用法参考 demo。

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%">
    <el-table-column
      label="日期"
      width="180">
      <template slot-scope="scope">
        <i class="el-icon-time"></i>
        <span style="margin-left: 10px">{{ scope.row.date }}</span>
      </template>
    </el-table-column>
    <el-table-column
      label="姓名"
      width="180">
      <template slot-scope="scope">
        <el-popover trigger="hover" placement="top">
          <p>姓名: {{ scope.row.name }}</p>
          <p>住址: {{ scope.row.address }}</p>
          <div slot="reference" class="name-wrapper">
            <el-tag size="medium">{{ scope.row.name }}</el-tag>
          </div>
        </el-popover>
      </template>
    </el-table-column>
    <el-table-column label="操作">
      <template slot-scope="scope">
        <el-button
          size="mini"
          @click="handleEdit(scope.$index, scope.row)">编辑</el-button>
```

```
        <el-button
          size="mini"
          type="danger"
          @click="handleDelete(scope.$index, scope.row)">删除</el-button>
      </template>
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-02',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-04',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1517 弄'
        }, {
          date: '2016-05-01',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1519 弄'
        }, {
          date: '2016-05-03',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1516 弄'
        }]
      }
    },
    methods: {
      handleEdit(index, row) {
        console.log(index, row);
      },
      handleDelete(index, row) {
        console.log(index, row);
      }
    }
  }
</script>
```

:::

展开行

当行内容过多并且不想显示横向滚动条时，可以使用 Table 展开行功能。 :::demo 通过设置 type="expand" 和 `Scoped slot` 可以开启展开行功能， `el-table-column` 的模板会被渲染成为展开行的内容，展开行可访问的属性与使用自定义列模板时的 `Scoped slot` 相同。

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%">
    <el-table-column type="expand">
      <template slot-scope="props">
        <el-form label-position="left" inline class="demo-table-expand">
          <el-form-item label="商品名称">
            <span>{{ props.row.name }}</span>
          </el-form-item>
          <el-form-item label="所属店铺">
            <span>{{ props.row.shop }}</span>
          </el-form-item>
          <el-form-item label="商品 ID">
            <span>{{ props.row.id }}</span>
          </el-form-item>
          <el-form-item label="店铺 ID">
            <span>{{ props.row.shopId }}</span>
          </el-form-item>
          <el-form-item label="商品分类">
            <span>{{ props.row.category }}</span>
          </el-form-item>
          <el-form-item label="店铺地址">
            <span>{{ props.row.address }}</span>
          </el-form-item>
          <el-form-item label="商品描述">
            <span>{{ props.row.desc }}</span>
          </el-form-item>
        </el-form>
      </template>
    </el-table-column>
    <el-table-column
      label="商品 ID"
      prop="id">
    </el-table-column>
    <el-table-column
      label="商品名称"
      prop="name">
    </el-table-column>
    <el-table-column
      label="描述"
      prop="desc">
    </el-table-column>
  </el-table>
</template>

<style>
  .demo-table-expand {
    font-size: 0;
  }
  .demo-table-expand label {
```

```
    width: 90px;
    color: #99a9bf;
  }
  .demo-table-expand .el-form-item {
    margin-right: 0;
    margin-bottom: 0;
    width: 50%;
  }
</style>

<script>
  export default {
    data() {
      return {
        tableData: [{
          id: '12987122',
          name: '好滋好味鸡蛋仔',
          category: '江浙小吃、小吃零食',
          desc: '荷兰优质淡奶，奶香浓而不腻',
          address: '上海市普陀区真北路',
          shop: '王小虎夫妻店',
          shopId: '10333'
        }, {
          id: '12987123',
          name: '好滋好味鸡蛋仔',
          category: '江浙小吃、小吃零食',
          desc: '荷兰优质淡奶，奶香浓而不腻',
          address: '上海市普陀区真北路',
          shop: '王小虎夫妻店',
          shopId: '10333'
        }, {
          id: '12987125',
          name: '好滋好味鸡蛋仔',
          category: '江浙小吃、小吃零食',
          desc: '荷兰优质淡奶，奶香浓而不腻',
          address: '上海市普陀区真北路',
          shop: '王小虎夫妻店',
          shopId: '10333'
        }, {
          id: '12987126',
          name: '好滋好味鸡蛋仔',
          category: '江浙小吃、小吃零食',
          desc: '荷兰优质淡奶，奶香浓而不腻',
          address: '上海市普陀区真北路',
          shop: '王小虎夫妻店',
          shopId: '10333'
        }]
      }
    }
  }
</script>
```

:::

树形数据与懒加载

:::demo 支持树类型的数据的显示。当 row 中包含 `children` 字段时，被视为树形数据。渲染树形数据时，必须要指定 `row-key`。支持子节点数据异步加载。设置 Table 的 `lazy` 属性为 true 与加载函数 `load`。通过指定 row 中的 `hasChildren` 字段来指定哪些行是包含子节点。`children` 与 `hasChildren` 都可以通过 `tree-props` 配置。

```
<template>
<div>
  <el-table
    :data="tableData"
    style="width: 100%;margin-bottom: 20px;"
    row-key="id"
    border
    default-expand-all
    :tree-props="{children: 'children', hasChildren: 'hasChildren'}">
    <el-table-column
      prop="date"
      label="日期"
      sortable
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="姓名"
      sortable
      width="180">
    </el-table-column>
    <el-table-column
      prop="address"
      label="地址">
    </el-table-column>
  </el-table>

  <el-table
    :data="tableData1"
    style="width: 100%"
    row-key="id"
    border
    lazy
    :load="load"
    :tree-props="{children: 'children', hasChildren: 'hasChildren'}">
    <el-table-column
      prop="date"
      label="日期"
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
```

```
      label="姓名"
      width="180">
    </el-table-column>
    <el-table-column
      prop="address"
      label="地址">
    </el-table-column>
  </el-table>
</div>
</template>
<script>
  export default {
    data() {
      return {
        tableData: [{
          id: 1,
          date: '2016-05-02',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          id: 2,
          date: '2016-05-04',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1517 弄'
        }, {
          id: 3,
          date: '2016-05-01',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1519 弄',
          children: [{
              id: 31,
              date: '2016-05-01',
              name: '王小虎',
              address: '上海市普陀区金沙江路 1519 弄'
            }, {
              id: 32,
              date: '2016-05-01',
              name: '王小虎',
              address: '上海市普陀区金沙江路 1519 弄'
          }]
        }, {
          id: 4,
          date: '2016-05-03',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1516 弄'
        }],
        tableData1: [{
          id: 1,
          date: '2016-05-02',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
```

```
          id: 2,
          date: '2016-05-04',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1517 弄'
        }, {
          id: 3,
          date: '2016-05-01',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1519 弄',
          hasChildren: true
        }, {
          id: 4,
          date: '2016-05-03',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1516 弄'
        }]
      }
    },
    methods: {
      load(tree, treeNode, resolve) {
        setTimeout(() => {
          resolve([
            {
              id: 31,
              date: '2016-05-01',
              name: '王小虎',
              address: '上海市普陀区金沙江路 1519 弄'
            }, {
              id: 32,
              date: '2016-05-01',
              name: '王小虎',
              address: '上海市普陀区金沙江路 1519 弄'
            }
          ])
        }, 1000)
      }
    },
  }
</script>
```

:::

自定义表头

表头支持自定义。

:::demo 通过设置 Scoped slot 来自定义表头。

```
<template>
  <el-table
    :data="tableData.filter(data => !search ||
data.name.toLowerCase().includes(search.toLowerCase()))"
```

```
    style="width: 100%">
    <el-table-column
      label="Date"
      prop="date">
    </el-table-column>
    <el-table-column
      label="Name"
      prop="name">
    </el-table-column>
    <el-table-column
      align="right">
      <template slot="header" slot-scope="scope">
        <el-input
          v-model="search"
          size="mini"
          placeholder="输入关键字搜索"/>
      </template>
      <template slot-scope="scope">
        <el-button
          size="mini"
          @click="handleEdit(scope.$index, scope.row)">Edit</el-button>
        <el-button
          size="mini"
          type="danger"
          @click="handleDelete(scope.$index, scope.row)">Delete</el-button>
      </template>
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-02',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        }, {
          date: '2016-05-04',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1517 弄'
        }, {
          date: '2016-05-01',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1519 弄'
        }, {
          date: '2016-05-03',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1516 弄'
        }],
        search: ''
```

```
      }
    },
    methods: {
      handleEdit(index, row) {
        console.log(index, row);
      },
      handleDelete(index, row) {
        console.log(index, row);
      }
    },
  }
</script>
```

:::

表尾合计行

若表格展示的是各类数字，可以在表尾显示各列的合计。 :::demo 将 `show-summary` 设置为 `true` 就会在表格尾部展示合计行。默认情况下，对于合计行，第一列不进行数据求合操作，而是显示「合计」二字（可通过 `sum-text` 配置），其余列会将本列所有数值进行求合操作，并显示出来。当然，你也可以定义自己的合计逻辑。使用 `summary-method` 并传入一个方法，返回一个数组，这个数组中的各项就会显示在合计行的各列中，具体可以参考本例中的第二个表格。

```
<template>
  <el-table
    :data="tableData"
    border
    show-summary
    style="width: 100%">
    <el-table-column
      prop="id"
      label="ID"
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="姓名">
    </el-table-column>
    <el-table-column
      prop="amount1"
      sortable
      label="数值 1">
    </el-table-column>
    <el-table-column
      prop="amount2"
      sortable
      label="数值 2">
    </el-table-column>
    <el-table-column
      prop="amount3"
      sortable
```

```
      label="数值 3">
    </el-table-column>
  </el-table>

  <el-table
    :data="tableData"
    border
    height="200"
    :summary-method="getSummaries"
    show-summary
    style="width: 100%; margin-top: 20px">
    <el-table-column
      prop="id"
      label="ID"
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="姓名">
    </el-table-column>
    <el-table-column
      prop="amount1"
      label="数值 1（元）">
    </el-table-column>
    <el-table-column
      prop="amount2"
      label="数值 2（元）">
    </el-table-column>
    <el-table-column
      prop="amount3"
      label="数值 3（元）">
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          id: '12987122',
          name: '王小虎',
          amount1: '234',
          amount2: '3.2',
          amount3: 10
        }, {
          id: '12987123',
          name: '王小虎',
          amount1: '165',
          amount2: '4.43',
          amount3: 12
        }, {
```

```
          id: '12987124',
          name: '王小虎',
          amount1: '324',
          amount2: '1.9',
          amount3: 9
        }, {
          id: '12987125',
          name: '王小虎',
          amount1: '621',
          amount2: '2.2',
          amount3: 17
        }, {
          id: '12987126',
          name: '王小虎',
          amount1: '539',
          amount2: '4.1',
          amount3: 15
        }]
      };
    },
    methods: {
      getSummaries(param) {
        const { columns, data } = param;
        const sums = [];
        columns.forEach((column, index) => {
          if (index === 0) {
            sums[index] = '总价';
            return;
          }
          const values = data.map(item => Number(item[column.property]));
          if (!values.every(value => isNaN(value))) {
            sums[index] = values.reduce((prev, curr) => {
              const value = Number(curr);
              if (!isNaN(value)) {
                return prev + curr;
              } else {
                return prev;
              }
            }, 0);
            sums[index] += ' 元';
          } else {
            sums[index] = 'N/A';
          }
        });

        return sums;
      }
    }
  };
</script>
```

:::

合并行或列

多行或多列共用一个数据时，可以合并行或列。 :::demo 通过给 `table` 传入 `span-method` 方法可以实现合并行或列，方法的参数是一个对象，里面包含当前行 `row` 、当前列 `column` 、当前行号 `rowIndex` 、当前列号 `columnIndex` 四个属性。该函数可以返回一个包含两个元素的数组，第一个元素代表 `rowspan` ，第二个元素代表 `colspan` 。 也可以返回一个键名为 `rowspan` 和 `colspan` 的对象。

```
<template>
  <div>
    <el-table
      :data="tableData"
      :span-method="arraySpanMethod"
      border
      style="width: 100%">
      <el-table-column
        prop="id"
        label="ID"
        width="180">
      </el-table-column>
      <el-table-column
        prop="name"
        label="姓名">
      </el-table-column>
      <el-table-column
        prop="amount1"
        sortable
        label="数值 1">
      </el-table-column>
      <el-table-column
        prop="amount2"
        sortable
        label="数值 2">
      </el-table-column>
      <el-table-column
        prop="amount3"
        sortable
        label="数值 3">
      </el-table-column>
    </el-table>

    <el-table
      :data="tableData"
      :span-method="objectSpanMethod"
      border
      style="width: 100%; margin-top: 20px">
      <el-table-column
        prop="id"
        label="ID"
        width="180">
      </el-table-column>
      <el-table-column
```

```
        prop="name"
        label="姓名">
      </el-table-column>
      <el-table-column
        prop="amount1"
        label="数值 1（元）">
      </el-table-column>
      <el-table-column
        prop="amount2"
        label="数值 2（元）">
      </el-table-column>
      <el-table-column
        prop="amount3"
        label="数值 3（元）">
      </el-table-column>
    </el-table>
  </div>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          id: '12987122',
          name: '王小虎',
          amount1: '234',
          amount2: '3.2',
          amount3: 10
        }, {
          id: '12987123',
          name: '王小虎',
          amount1: '165',
          amount2: '4.43',
          amount3: 12
        }, {
          id: '12987124',
          name: '王小虎',
          amount1: '324',
          amount2: '1.9',
          amount3: 9
        }, {
          id: '12987125',
          name: '王小虎',
          amount1: '621',
          amount2: '2.2',
          amount3: 17
        }, {
          id: '12987126',
          name: '王小虎',
          amount1: '539',
          amount2: '4.1',
```

```
          amount3: 15
        }]
      };
    },
    methods: {
      arraySpanMethod({ row, column, rowIndex, columnIndex }) {
        if (rowIndex % 2 === 0) {
          if (columnIndex === 0) {
            return [1, 2];
          } else if (columnIndex === 1) {
            return [0, 0];
          }
        }
      },

      objectSpanMethod({ row, column, rowIndex, columnIndex }) {
        if (columnIndex === 0) {
          if (rowIndex % 2 === 0) {
            return {
              rowspan: 2,
              colspan: 1
            };
          } else {
            return {
              rowspan: 0,
              colspan: 0
            };
          }
        }
      }
    }
  };
</script>
```

:::

自定义索引

自定义 `type=index` 列的行号。 :::demo 通过给 `type=index` 的列传入 `index` 属性，可以自定义索引。该属性传入数字时，将作为索引的起始值。也可以传入一个方法，它提供当前行的行号（从 `0` 开始）作为参数，返回值将作为索引展示。

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%">
    <el-table-column
      type="index"
      :index="indexMethod">
    </el-table-column>
    <el-table-column
```

```
      prop="date"
      label="日期"
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="姓名"
      width="180">
    </el-table-column>
    <el-table-column
      prop="address"
      label="地址">
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-02',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1518 弄',
          zip: 200333,
          tag: '家'
        }, {
          date: '2016-05-04',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1517 弄',
          zip: 200333,
          tag: '公司'
        }, {
          date: '2016-05-01',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1519 弄',
          zip: 200333,
          tag: '家'
        }, {
          date: '2016-05-03',
          name: '王小虎',
          province: '上海',
          city: '普陀区',
          address: '上海市普陀区金沙江路 1516 弄',
          zip: 200333,
          tag: '公司'
```

```
        }],
      }
    },
    methods: {
      indexMethod(index) {
        return index * 2;
      }
    }
  };
</script>
```

:::

Table Attributes

参数	说明	类型	可选值	默认值
data	显示的数据	array	—	—
height	Table 的高度，默认为自动高度。如果 height 为 number 类型，单位 px；如果 height 为 string 类型，则这个高度会设置为 Table 的 style.height 的值，Table 的高度会受控于外部样式。	string/number	—	—
max-height	Table 的最大高度。合法的值为数字或者单位为 px 的高度。	string/number	—	—
stripe	是否为斑马纹 table	boolean	—	false
border	是否带有纵向边框	boolean	—	false
size	Table 的尺寸	string	medium / small / mini	—
fit	列的宽度是否自撑开	boolean	—	true
show-header	是否显示表头	boolean	—	true
highlight-current-row	是否要高亮当前行	boolean	—	false
current-row-key	当前行的 key，只写属性	String,Number	—	—
row-class-name	行的 className 的回调方法，也可以使用字符串为所有行设置一个固定的 className。	Function({row, rowIndex})/String	—	—
row-style	行的 style 的回调方法，也可以使用一个固定的 Object 为所有行设置一样的 Style。	Function({row, rowIndex})/Object	—	—

cell-class-name	单元格的 className 的回调方法，也可以使用字符串为所有单元格设置一个固定的 className。	Function({row, column, rowIndex, columnIndex})/String	—	—
cell-style	单元格的 style 的回调方法，也可以使用一个固定的 Object 为所有单元格设置一样的 Style。	Function({row, column, rowIndex, columnIndex})/Object	—	—
header-row-class-name	表头行的 className 的回调方法，也可以使用字符串为所有表头行设置一个固定的 className。	Function({row, rowIndex})/String	—	—
header-row-style	表头行的 style 的回调方法，也可以使用一个固定的 Object 为所有表头行设置一样的 Style。	Function({row, rowIndex})/Object	—	—
header-cell-class-name	表头单元格的 className 的回调方法，也可以使用字符串为所有表头单元格设置一个固定的 className。	Function({row, column, rowIndex, columnIndex})/String	—	—
header-cell-style	表头单元格的 style 的回调方法，也可以使用一个固定的 Object 为所有表头单元格设置一样的 Style。	Function({row, column, rowIndex, columnIndex})/Object	—	—
row-key	行数据的 Key，用来优化 Table 的渲染；在使用 reserve-selection 功能与显示树形数据时，该属性是必填的。类型为 String 时，支持多层访问：`user.info.id`，但不支持 `user.info[0].id`，此种情况请使用 `Function`。	Function(row)/String	—	—
empty-text	空数据时显示的文本内容，也可以通过 `slot="empty"` 设置	String	—	暂无数据
default-expand-all	是否默认展开所有行，当 Table 包含展开行存在或者为树形表格时有效	Boolean	—	false
expand-row-keys	可以通过该属性设置 Table 目前的展开行，需要设置 row-key 属性才能使用，该属性为展开行的 keys 数组。	Array	—	
default-sort	默认的排序列的 prop 和顺序。它的`prop`属性指定默认的排序的列，`order`指定默认排序的顺序	Object	`order`: ascending, descending	如果只指定了`prop`, 没有指定`order`, 则默认顺序是ascending

tooltip-effect	tooltip `effect` 属性	String	dark/light	
show-summary	是否在表尾显示合计行	Boolean	—	false
sum-text	合计行第一列的文本	String	—	合计
summary-method	自定义的合计计算方法	Function({ columns, data })	—	—
span-method	合并行或列的计算方法	Function({ row, column, rowIndex, columnIndex })	—	—
select-on-indeterminate	在多选表格中，当仅有部分行被选中时，点击表头的多选框时的行为。若为 true，则选中所有行；若为 false，则取消选择所有行	Boolean	—	true
indent	展示树形数据时，树节点的缩进	Number	—	16
lazy	是否懒加载子节点数据	Boolean	—	—
load	加载子节点数据的函数，lazy 为 true 时生效，函数第二个参数包含了节点的层级信息	Function(row, treeNode, resolve)	—	—
tree-props	渲染嵌套数据的配置选项	Object	—	{ hasChildren: 'hasChildren', children: 'children' }

Table Events

事件名	说明	参数
select	当用户手动勾选数据行的 Checkbox 时触发的事件	selection, row
select-all	当用户手动勾选全选 Checkbox 时触发的事件	selection
selection-change	当选择项发生变化时会触发该事件	selection
cell-mouse-enter	当单元格 hover 进入时会触发该事件	row, column, cell, event
cell-mouse-leave	当单元格 hover 退出时会触发该事件	row, column, cell, event
cell-click	当某个单元格被点击时会触发该事件	row, column, cell, event
cell-dblclick	当某个单元格被双击击时会触发该事件	row, column, cell, event

row-click	当某一行被点击时会触发该事件	row, column, event
row-contextmenu	当某一行被鼠标右键点击时会触发该事件	row, column, event
row-dblclick	当某一行被双击时会触发该事件	row, column, event
header-click	当某一列的表头被点击时会触发该事件	column, event
header-contextmenu	当某一列的表头被鼠标右键点击时触发该事件	column, event
sort-change	当表格的排序条件发生变化的时候会触发该事件	{ column, prop, order }
filter-change	当表格的筛选条件发生变化的时候会触发该事件，参数的值是一个对象，对象的 key 是 column 的 columnKey，对应的 value 为用户选择的筛选条件的数组。	filters
current-change	当表格的当前行发生变化的时候会触发该事件，如果要高亮当前行，请打开表格的 highlight-current-row 属性	currentRow, oldCurrentRow
header-dragend	当拖动表头改变了列的宽度的时候会触发该事件	newWidth, oldWidth, column, event
expand-change	当用户对某一行展开或者关闭的时候会触发该事件（展开行时，回调的第二个参数为 expandedRows；树形表格时第二参数为 expanded）	row, (expandedRows | expanded)

Table Methods

方法名	说明	参数
clearSelection	用于多选表格，清空用户的选择	—
toggleRowSelection	用于多选表格，切换某一行的选中状态，如果使用了第二个参数，则是设置这一行选中与否（selected 为 true 则选中）	row, selected
toggleAllSelection	用于多选表格，切换所有行的选中状态	-
toggleRowExpansion	用于可展开表格与树形表格，切换某一行的展开状态，如果使用了第二个参数，则是设置这一行展开与否（expanded 为 true 则展开）	row, expanded
setCurrentRow	用于单选表格，设定某一行为选中行，如果调用时不加参数，则会取消目前高亮行的选中状态。	row
clearSort	用于清空排序条件，数据会恢复成未排序的状态	—
clearFilter	不传入参数时用于清空所有过滤条件，数据会恢复成未过滤的状态，也可传入由columnKey组成的数组以清除指定列的过滤条件	columnKey
doLayout	对 Table 进行重新布局。当 Table 或其祖先元素由隐藏切换为显示时，可能需要调用此方法	—

sort	手动对 Table 进行排序。参数`prop`属性指定排序列，`order`指定排序顺序。	prop: string, order: string

Table Slot

name	说明
append	插入至表格最后一行之后的内容，如果需要对表格的内容进行无限滚动操作，可能需要用到这个 slot。若表格有合计行，该 slot 会位于合计行之上。

Table-column Attributes

参数	说明	类型	可选值	默认值
type	对应列的类型。如果设置了 `selection` 则显示多选框；如果设置了 `index` 则显示该行的索引（从 1 开始计算）；如果设置了 `expand` 则显示为一个可展开的按钮	string	selection/index/expand	—
index	如果设置了 `type=index`，可以通过传递 `index` 属性来自定义索引	number, Function(index)	-	-
column-key	column 的 key，如果需要使用 filter-change 事件，则需要此属性标识是哪个 column 的筛选条件	string	—	—
label	显示的标题	string	—	—
prop	对应列内容的字段名，也可以使用 property 属性	string	—	—
width	对应列的宽度	string	—	—
min-width	对应列的最小宽度，与 width 的区别是 width 是固定的，min-width 会把剩余宽度按比例	string	—	—

	分配给设置了 min-width 的列			
fixed	列是否固定在左侧或者右侧，true 表示固定在左侧	string, boolean	true, left, right	—
render-header	列标题 Label 区域渲染使用的 Function	Function(h, { column, $index })	—	—
sortable	对应列是否可以排序，如果设置为 'custom'，则代表用户希望远程排序，需要监听 Table 的 sort-change 事件	boolean, string	true, false, 'custom'	false
sort-method	对数据进行排序的时候使用的方法，仅当 sortable 设置为 true 的时候有效，需返回一个数字，和 Array.sort 表现一致	Function(a, b)	—	—
sort-by	指定数据按照哪个属性进行排序，仅当 sortable 设置为 true 且没有设置 sort-method 的时候有效。如果 sort-by 为数组，则先按照第 1 个属性排序，如果第 1 个相等，再按照第 2 个排序，以此类推	String/Array/Function(row, index)	—	—
sort-orders	数据在排序时所使用排序策略的轮转顺序，仅当 sortable 为 true 时有效。需传入一个数组，随着用户点击表头，该列依次按照数组中元素的顺序进行排序	array	数组中的元素需为以下三者之一：`ascending` 表示升序，`descending` 表示降序，`null` 表示还原为原始顺序	['ascending', 'descending', null]
resizable	对应列是否可以通过拖动改变宽度（需要在 el-table	boolean	—	true

	上设置 border 属性为真)			
formatter	用来格式化内容	Function(row, column, cellValue, index)	—	—
show-overflow-tooltip	当内容过长被隐藏时显示 tooltip	Boolean	—	false
align	对齐方式	String	left/center/right	left
header-align	表头对齐方式，若不设置该项，则使用表格的对齐方式	String	left/center/right	—
class-name	列的 className	string	—	—
label-class-name	当前列标题的自定义类名	string	—	—
selectable	仅对 type=selection 的列有效，类型为 Function，Function 的返回值用来决定这一行的 CheckBox 是否可以勾选	Function(row, index)	—	—
reserve-selection	仅对 type=selection 的列有效，类型为 Boolean，为 true 则会在数据更新之后保留之前选中的数据（需指定 `row-key`）	Boolean	—	false
filters	数据过滤的选项，数组格式，数组中的元素需要有 text 和 value 属性。	Array[{ text, value }]	—	—
filter-placement	过滤弹出框的定位	String	与 Tooltip 的 `placement` 属性相同	—
filter-multiple	数据过滤的选项是否多选	Boolean	—	true
filter-method	数据过滤使用的方法，如果是多选的筛选项，对每一条	Function(value, row, column)	—	—

	数据会执行多次，任意一次返回 true 就会显示。			
filtered-value	选中的数据过滤项，如果需要自定义表头过滤的渲染方式，可能会需要此属性。	Array	—	—

Table-column Scoped Slot

name	说明
—	自定义列的内容，参数为 { row, column, $index }
header	自定义表头的内容. 参数为 { column, $index }