Tablas

Visualiza múltiples datos con un formato en particular. Podrá ordenar, filtrar y comparar datos en una tabla.

Tabla básica

La tabla básica es solo para mostrar datos.

:::demo Después de haber establecido el atributo `data` de `el-table` con un arreglo de objetos, puede usar la propiedad `prop` (el correspondiente a la clave de un objeto dentro del arreglo `data`) en `el-table-column` para insertar datos a las columnas de la tabla, y establecer el atributo `label` para definir el nombre de la columna. También puede usar el atributo `width` para establecer el ancho de las columnas.

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

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-03',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-02',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-04',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-01',
```

```
            name: 'Tom',
            address: 'No. 189, Grove St, Los Angeles'
          }]
        }
      }
    }
  </script>
```

:::

Tabla con franjas

La tabla con franjas hace más fácil distinguir filas diferentes.

:::demo El atributo `stripe` también acepta un `Boolean`. Si se encuentra `true`, la tabla será con franjas.

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

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-03',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-02',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-04',
```

```
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-01',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }]
      }
    }
  }
</script>
```

:::

Tabla con bordes

:::demo Por defecto, la tabla no tiene bordes verticales. Si lo necesita, puede establecer el atributo `border` a `true` .

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

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-03',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-02',
```

```
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-04',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-01',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }]
      }
    }
  }
</script>
```

:::

Tabla con estados

Puede destacar el contenido de la tabla para distinguir entre "success, information, warning, danger" y otros estados.

:::demo Utilice `row-class-name` en `el-table` para agregar clases personalizadas a una fila en específico. Y entonces, podrá darle diseño con estas clases.

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%"
    :row-class-name="tableRowClassName">
    <el-table-column
      prop="date"
      label="Fecha"
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="Nombre"
      width="180">
    </el-table-column>
    <el-table-column
      prop="address"
      label="Dirección">
    </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: '2016-05-03',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-02',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-04',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-01',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }]
      }
    }
  }
</script>
```

:::

Tabla con cabecera fija

Cuando esta tiene demasiadas filas, puede utilizar una cabecera fija.

:::demo Al configurar el atributo `height` de `el-table`, puede fijar la cabecera de la tabla sin agregar otro código.

```
<template>
  <el-table
```

```
    :data="tableData"
    height="250"
    style="width: 100%">
    <el-table-column
      prop="date"
      label="Fecha"
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="Nombre"
      width="180">
    </el-table-column>
    <el-table-column
      prop="address"
      label="Dirección">
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-03',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-02',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-04',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-01',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-08',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-06',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-07',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
```

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

:::

Tabla con columnas fijas

Cuando se tienen demasiadas columnas, puede fijar alguna de estas.

:::demo El atributo `fixed` es utilizado en `el-table-column`, este acepta un `Boolean`. Si es `true`, la columna será fijada a la izquierda. También acepta dos tipos: `left` y `right`, ambos indican donde debe ser fijada la columna.

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%">
    <el-table-column
      fixed
      prop="date"
      label="Fecha"
      width="150">
    </el-table-column>
    <el-table-column
      prop="name"
      label="Nombre"
      width="120">
    </el-table-column>
    <el-table-column
      prop="state"
      label="Estado"
      width="120">
    </el-table-column>
    <el-table-column
      prop="city"
      label="Ciudad"
      width="120">
    </el-table-column>
    <el-table-column
      prop="address"
      label="Dirección"
      width="300">
    </el-table-column>
    <el-table-column
      prop="zip"
      label="Código postal"
      width="120">
    </el-table-column>
    <el-table-column
```

```
      fixed="right"
      label="Operaciones"
      width="120">
      <template slot-scope="scope">
        <el-button @click="handleClick" type="text" size="small">Detalle</el-button>
        <el-button type="text" size="small">Editar</el-button>
      </template>
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    methods: {
      handleClick() {
        console.log('click');
      }
    },
    data() {
      return {
        tableData: [{
          date: '2016-05-03',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036',
          tag: 'Home'
        }, {
          date: '2016-05-02',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036',
          tag: 'Office'
        }, {
          date: '2016-05-04',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036',
          tag: 'Home'
        }, {
          date: '2016-05-01',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036',
          tag: 'Office'
```

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

:::

Tabla con columnas y cabecera fija.

Cuando tienes grandes cantidades de datos para colocar en una tabla, puede fijar la cabecera y columnas al mismo tiempo.

:::demo Fije las columnas y cabecera al mismo tiempo combinando los dos ejemplos anteriores.

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%"
    height="250">
    <el-table-column
      fixed
      prop="date"
      label="Fecha"
      width="150">
    </el-table-column>
    <el-table-column
      prop="name"
      label="Nombre"
      width="120">
    </el-table-column>
    <el-table-column
      prop="state"
      label="Estado"
      width="120">
    </el-table-column>
    <el-table-column
      prop="city"
      label="Ciudad"
      width="120">
    </el-table-column>
    <el-table-column
      prop="address"
      label="Dirección"
      width="300">
    </el-table-column>
    <el-table-column
      prop="zip"
      label="Código postal"
      width="120">
    </el-table-column>
  </el-table>
```

```
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-03',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-02',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-04',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-01',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-08',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-06',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-07',
          name: 'Tom',
          state: 'California',
```

```
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }]
      }
    }
  }
</script>
```

:::

Altura fluido de tabla con cabecera fija (y columnas)

Cuando los datos se modifican dinámicamente, es posible que necesite que la tabla tenga una altura máxima en lugar de una altura fija, y además, que se muestre la barra de desplazamiento si es necesario.

:::demo Al configurar el atributo `max-height` de `el-table`, tu puedes fijar la cabecera de la tabla. La barra de desplazamiento únicamente se mostrará si la altura sobrepasa el valor de la altura máxima.

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%"
    max-height="250">
    <el-table-column
      fixed
      prop="date"
      label="Fecha"
      width="150">
    </el-table-column>
    <el-table-column
      prop="name"
      label="Nombre"
      width="120">
    </el-table-column>
    <el-table-column
      prop="state"
      label="Estado"
      width="120">
    </el-table-column>
    <el-table-column
      prop="city"
      label="Ciudad"
      width="120">
    </el-table-column>
    <el-table-column
      prop="address"
      label="Dirección"
      width="300">
    </el-table-column>
    <el-table-column
      prop="zip"
```

```
      label="Código postal"
      width="120">
    </el-table-column>
    <el-table-column
      fixed="right"
      label="Operaciones"
      width="120">
      <template slot-scope="scope">
        <el-button
          @click.native.prevent="deleteRow(scope.$index, tableData)"
          type="text"
          size="small">
          Eliminar
        </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: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-02',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-04',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-01',
          name: 'Tom',
```

```
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-08',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-06',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-07',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }]
      }
    }
  }
</script>
```

:::

Agrupando cabeceras de la tabla

Cuando la estructura de datos es compleja, tu puedes hacer uso de cabeceras agrupadas para mostrar datos por jerarquía.

:::demo Solo necesitas colocar `el-table-column` dentro de un `el-table-column`, de esta forma lograrás agruparlas.

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%">
    <el-table-column
      prop="date"
      label="Fecha"
      width="150">
    </el-table-column>
    <el-table-column label="Información de entrega">
      <el-table-column
```

```
        prop="name"
        label="Nombre"
        width="120">
      </el-table-column>
      <el-table-column label="Información de dirección">
        <el-table-column
          prop="state"
          label="Estado"
          width="120">
        </el-table-column>
        <el-table-column
          prop="city"
          label="Ciudad"
          width="120">
        </el-table-column>
        <el-table-column
          prop="address"
          label="Dirección"
          width="300">
        </el-table-column>
        <el-table-column
          prop="zip"
          label="Código postal"
          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: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-02',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-04',
          name: 'Tom',
          state: 'California',
```

```
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-01',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-08',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-06',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-07',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }]
      }
    }
  }
</script>
```

:::

Selección única

La selección de una fila esta soportada.

:::demo La tabla permite la selección de una sola fila. Puede activarlo agregando el atributo `highlight-current-row`. Un evento llamado `current-change` será disparado cuando la selección de la fila cambie, sus parámetros son la fila antes y después de que ocurre el cambio: `currentRow` y `oldCurrentRow`. Si necesita mostrar el índice de la fila, puede agregar un nuevo `el-table-column` con el atributo `type` asignado al `index` y podrá ver el índice iniciando desde 1.

```
<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="Fecha"
      width="120">
    </el-table-column>
    <el-table-column
      property="name"
      label="Nombre"
      width="120">
    </el-table-column>
    <el-table-column
      property="address"
      label="Dirección">
    </el-table-column>
  </el-table>
  <div style="margin-top: 20px">
    <el-button @click="setCurrent(tableData[1])">Seleccionar segunda fila</el-
button>
    <el-button @click="setCurrent()">Limpiar selección</el-button>
  </div>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-03',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-02',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-04',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-01',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }],
```

```
        currentRow: null
      }
    },

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

:::

Selección multiple

También puede seleccionar múltiples filas.

:::demo Activar la selección múltiple es sencillo: Solo debe agregar a `el-table-column` con su `type` establecido en `selection`. Además de la selección múltiple, este ejemplo también utiliza `show-overflow-tooltip`: por defecto, si el contenido es demasiado largo, este permite córtalo dentro de múltiples líneas. Si lo que busca es mantener una línea, utilice el atributo `show-overflow-tooltip`, que acepta un valor `Boolean`. Cuando este está establecido en `true`, el contenido extra puede mostrar un *tooltip* cuando se hace *hover* sobre la celda.

```
<template>
  <el-table
    ref="multipleTable"
    :data="tableData"
    style="width: 100%"
    @selection-change="handleSelectionChange">
    <el-table-column
      type="selection"
      width="55">
    </el-table-column>
    <el-table-column
      label="Fecha"
      width="120">
      <template slot-scope="scope">{{ scope.row.date }}</template>
    </el-table-column>
    <el-table-column
      property="name"
      label="Nombre"
      width="120">
    </el-table-column>
    <el-table-column
      property="address"
      label="Dirección"
```

```
      show-overflow-tooltip>
    </el-table-column>
  </el-table>
  <div style="margin-top: 20px">
    <el-button @click="toggleSelection([tableData[1], tableData[2]])">Cambia el
estado de selección de la segunda y tercera fila.</el-button>
    <el-button @click="toggleSelection()">Limpiar selección</el-button>
  </div>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-03',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-02',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-04',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-01',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-08',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-06',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-07',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }],
        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>
```

:::

Ordenamiento

Ordena los datos para encontrar o comparar información rápidamente.

:::demo Establezca el atributo `sortable` para ordenar los datos de una columna. Este acepta un `Boolean` con un valor por defecto `false`. Establezca el atributo `default-sort` para determinar la columna y orden por defecto. Para aplicar sus propias reglas de ordenamiento, utilice `sort-method` o `sort-by`. Si lo que necesita es ordenar de forma remota desde backend, establezca `sortable` a `custom`, y escuche el evento `sort-change` de la tabla. Al dispararse el evento, tendrá acceso a la columna ordenada y orden para que pueda obtener los datos de la tabla ordenada desde su API. En este ejemplo utilizamos otro atributo llamado `formatter` para darle un formato al valor de ciertas columnas. Este acepta una función que tiene dos parámetros: `row` y `column`. Puede disparar este de acuerdo a sus propias necesidades.

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

```
<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-03',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-02',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-04',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-01',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }]
      }
    },
    methods: {
      formatter(row, column) {
        return row.address;
      }
    }
  }
</script>
```

:::

Filtros

Filtra la tabla para encontrar la información que necesita.

:::demo Establezca el atributo `filters` y `filter-method` en `el-table-column` haciendo esta columna filtrable. `filters` es un arreglo, y `filter-method` es una función que decide que filas se muestra. Esta tiene tres parámetros: `value`, `row` y `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="Fecha"
      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="Nombre"
      width="180">
    </el-table-column>
    <el-table-column
      prop="address"
      label="Dirección"
      :formatter="formatter">
    </el-table-column>
    <el-table-column
      prop="tag"
      label="Etiqueta"
      width="100"
      :filters="[{ text: 'Home', value: 'Home' }, { text: 'Office', value: 'Office'
}]"
      :filter-method="filterTag"
      filter-placement="bottom-end">
      <template slot-scope="scope">
        <el-tag
          :type="scope.row.tag === 'Home' ? 'primary' : 'success'"
          disable-transitions>{{scope.row.tag}}</el-tag>
      </template>
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-03',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles',
          tag: 'Home'
        }, {
          date: '2016-05-02',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles',
          tag: 'Office'
        }, {
          date: '2016-05-04',
          name: 'Tom',
```

```
          address: 'No. 189, Grove St, Los Angeles',
          tag: 'Home'
        }, {
          date: '2016-05-01',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles',
          tag: 'Office'
        }]
      }
    },
    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>
```

:::

Plantilla de columna personalizada

Personalice la columna de la tabla para que pueda integrarse con otros componentes.

:::demo Tiene acceso a la siguiente información: row, column, $index, store (gestor de estados de la tabla) por Scoped slots.

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%">
    <el-table-column
      label="Fecha"
      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="Nombre"
      width="180">
      <template slot-scope="scope">
        <el-popover trigger="hover" placement="top">
          <p>Name: {{ scope.row.name }}</p>
          <p>Addr: {{ 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="Operaciones">
      <template slot-scope="scope">
        <el-button
          size="mini"
          @click="handleEdit(scope.$index, scope.row)">Editar</el-button>
        <el-button
          size="mini"
          type="danger"
          @click="handleDelete(scope.$index, scope.row)">Eliminar</el-button>
      </template>
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-03',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-02',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-04',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-01',
          name: 'Tom',
          address: 'No. 189, Grove St, Los Angeles'
        }]
      }
    },
    methods: {
```

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

:::

Table con cabecera personalizada

Se puede personalizar el encabezado de la tabla para que se pueda adaptar aún más. :::demo Puede personalizar el aspecto del encabezado con header scoped slots.

```
<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="Type to search"/>
      </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: 'Tom',
          address: 'No. 189, Grove St, Los Angeles',
        }, {
          date: '2016-05-04',
          name: 'John',
          address: 'No. 189, Grove St, Los Angeles',
        }, {
          date: '2016-05-01',
          name: 'Morgan',
          address: 'No. 189, Grove St, Los Angeles',
        }, {
          date: '2016-05-03',
          name: 'Jessy',
          address: 'No. 189, Grove St, Los Angeles',
        }],
        search: ''
      }
    },
    methods: {
      handleEdit(index, row) {
        console.log(index, row);
      },
      handleDelete(index, row) {
        console.log(index, row);
      }
    },
  }
</script>
```

:::

Fila expandible

Cuando el contenido de la fila es demasiado largo y busca no mostrar la barra de desplazamiento horizontal, puede utilizar la característica de fila expandible.

:::demo Puede activar la fila expandible estableciendo la propiedad `type` a `expand` y Scoped Slots. La plantilla para `el-table-column` se representará como el contenido de la fila expandible, y puede acceder a algunos atributos cuando está usando `Scoped Slots` en plantillas de columna personalizadas.

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%">
    <el-table-column type="expand">
      <template slot-scope="props">
        <p>Estado: {{ props.row.state }}</p>
```

```
        <p>Ciudad: {{ props.row.city }}</p>
        <p>Dirección: {{ props.row.address }}</p>
        <p>Código postal: {{ props.row.zip }}</p>
      </template>
    </el-table-column>
    <el-table-column
      label="Fecha"
      prop="date">
    </el-table-column>
    <el-table-column
      label="Nombre"
      prop="name">
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-03',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-02',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-04',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-01',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-08',
          name: 'Tom',
          state: 'California',
```

```
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-06',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }, {
          date: '2016-05-07',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036'
        }]
      }
    }
  }
</script>
```

:::

Datos tree y modo lazy

:::demo Puede visualizar datos de estructura de árbol. Cuando la fila contiene el campo `children`, se trata como datos anidados. Para renderizar datos anidados, la `row-key` es requerida。Ademas, los datos de las filas secundarias pueden ser cargados asincrónicamente. Poner la propiedad `lazy` de Table a true y la función `load`. Especifique el atributo `hasChildren` en la fila para determinar qué fila contiene descendencia. Tanto `children` como `hasChildren` pueden configurarse a través de `tree-props`.

```
<template>
<div>
  <el-table
    :data="tableData"
    style="width: 100%;margin-bottom: 20px;"
    row-key="id"
    border
    default-expand-all>
    <el-table-column
      prop="date"
      label="日期"
      sortable
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="name"
      sortable
```

```
      width="180">
    </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="date"
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="name"
      width="180">
    </el-table-column>
  </el-table>
</div>
</template>
<script>
  export default {
    data() {
      return {
        tableData: [{
          id: 1,
          date: '2016-05-02',
          name: 'wangxiaohu'
        }, {
          id: 2,
          date: '2016-05-04',
          name: 'wangxiaohu'
        }, {
          id: 3,
          date: '2016-05-01',
          name: 'wangxiaohu',
          children: [{
              id: 31,
              date: '2016-05-01',
              name: 'wangxiaohu'
            }, {
              id: 32,
              date: '2016-05-01',
              name: 'wangxiaohu'
          }]
        }, {
          id: 4,
```

```
          date: '2016-05-03',
          name: 'wangxiaohu'
        }],
        tableData1: [{
          id: 1,
          date: '2016-05-02',
          name: 'wangxiaohu'
        }, {
          id: 2,
          date: '2016-05-04',
          name: 'wangxiaohu'
        }, {
          id: 3,
          date: '2016-05-01',
          name: 'wangxiaohu',
          hasChildren: true
        }, {
          id: 4,
          date: '2016-05-03',
          name: 'wangxiaohu'
        }]
      }
    },
    methods: {
      load(tree, treeNode, resolve) {
        setTimeout(() => {
          resolve([
            {
              id: 31,
              date: '2016-05-01',
              name: 'wangxiaohu'
            }, {
              id: 32,
              date: '2016-05-01',
              name: 'wangxiaohu'
            }
          ])
        }, 1000)
      }
    },
  }
</script>
```

:::

Fila de resumen

Para una tabla de números, puede agregar una fila extra en el pie de página de la tabla que muestra la suma de cada columna.

:::demo Puede agregar la fila de resumen configurando `show-summary` a `true`. Por defecto, para la fila de resumen, la primera columna no resume nada, pero siempre muestra la suma (puede configurar el texto mostrado

usando `sum-text`), mientras que otras columnas suman todos los números en esa columna y los muestran. Por supuesto, puede definir su propio comportamiento de suma. Para hacerlo, utiliza un método `summary-method`, que devuelve un array, y cada elemento que fue retornado desde el arreglo puede ser mostrado en las columnas del resumen de fila. La segunda tabla de este ejemplo es una demostración detallada.

```
<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="Nombre">
    </el-table-column>
    <el-table-column
      prop="amount1"
      sortable
      label="Monto 1">
    </el-table-column>
    <el-table-column
      prop="amount2"
      sortable
      label="Monto 2">
    </el-table-column>
    <el-table-column
      prop="amount3"
      sortable
      label="Monto 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="Nombre">
    </el-table-column>
    <el-table-column
      prop="amount1"
      label="Costo 1 ($)">
    </el-table-column>
    <el-table-column
      prop="amount2"
      label="Costo 2 ($)">
    </el-table-column>
    <el-table-column
      prop="amount3"
      label="Costo 3 ($)">
    </el-table-column>
  </el-table>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          id: '12987122',
          name: 'Tom',
          amount1: '234',
          amount2: '3.2',
          amount3: 10
        }, {
          id: '12987123',
          name: 'Tom',
          amount1: '165',
          amount2: '4.43',
          amount3: 12
        }, {
          id: '12987124',
          name: 'Tom',
          amount1: '324',
          amount2: '1.9',
          amount3: 9
        }, {
          id: '12987125',
          name: 'Tom',
          amount1: '621',
          amount2: '2.2',
          amount3: 17
        }, {
          id: '12987126',
          name: 'Tom',
          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] = 'Costo total';
            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);
          } else {
            sums[index] = 'N/A';
          }
        });

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

:::

Fusión de filas y columnas

Configurar *rowspan* y *colspan* le permite fusionar celdas.

:::demo Utilice el atributo `span-method` para configurar rowspan y colspan. Este acepta un método, y pasa un objeto a ese método incluyendo la fila actual `row`, columna actual `column`, índice de fila actual `rowIndex` y índice de columna actual `columnIndex`. El método debe devolver un arreglo de dos números, el primer número siendo `rowspan` y el segundo `colspan`. También puede devolver un objeto con las propiedades `rowspan` y `colspan`.

```
<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="Nombre">
    </el-table-column>
    <el-table-column
      prop="amount1"
      sortable
      label="Monto 1">
    </el-table-column>
    <el-table-column
      prop="amount2"
      sortable
      label="Monto 2">
    </el-table-column>
    <el-table-column
      prop="amount3"
      sortable
      label="Monto 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="Nombre">
    </el-table-column>
    <el-table-column
      prop="amount1"
      label="Monto 1">
    </el-table-column>
    <el-table-column
      prop="amount2"
      label="Monto 2">
    </el-table-column>
    <el-table-column
      prop="amount3"
      label="Monto 3">
    </el-table-column>
```

```
    </el-table>
  </div>
</template>

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

:::

Índice personalizado

Puede personalizar el índice de la fila con la propiedad `type=index` de las columnas.

:::demo Para personalizar el índice de la fila, utilice el atributo `index` en `<el-table-column>` con `type=index`. Si este es asignado a un número, todos los índices tendrán un desplazamiento de ese número. Este también acepta un método con cada índice (iniciando desde 0) como un parámetro, y este devuelve un valor que puede ser mostrado como índice.

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%">
    <el-table-column
      type="index"
      :index="indexMethod">
    </el-table-column>
    <el-table-column
      prop="date"
      label="Fecha"
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="Nombre"
      width="180">
    </el-table-column>
    <el-table-column
      prop="address"
      label="Dirección">
    </el-table-column>
```

```
  </el-table>
</template>

<script>
  export default {
    data() {
      return {
        tableData: [{
          date: '2016-05-03',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036',
          tag: 'Home'
        }, {
          date: '2016-05-02',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036',
          tag: 'Office'
        }, {
          date: '2016-05-04',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036',
          tag: 'Home'
        }, {
          date: '2016-05-01',
          name: 'Tom',
          state: 'California',
          city: 'Los Angeles',
          address: 'No. 189, Grove St, Los Angeles',
          zip: 'CA 90036',
          tag: 'Office'
        }],
      }
    },
    methods: {
      indexMethod(index) {
        return index * 2;
      }
    }
  };
</script>
```

:::

Atributos de la tabla

Atributo	Descripción	Tipo	Valores aceptados	Por defecto
data	Datos de la tabla	array	—	—
height	Altura de la tabla. Por defecto esta tiene un tamaño `auto`. Si este valor es un número, la altura es medido en pixeles; si este valor es una cadena, la altura es afectada por estilos externos.	string/number	—	—
max-height	El max-height de la tabla. El valor puede ser un numero o el alto en px.	string/number	—	—
stripe	especifica si la tabla será en franjas	boolean	—	false
border	especifica si la tabla tiene bordes verticales	boolean	—	false
size	tamaño de la tabla	string	medium / small / mini	—
fit	especifica si el ancho de la columna se adapta automáticamente a su contenedor	boolean	—	true
show-header	especifica si la cabecera de la tabla es visible	boolean	—	true
highlight-current-row	especifica si la fila actual es resaltado	boolean	—	false
current-row-key	clave de la fila actual, un ajuste de propiedad única	string,number	—	—
row-class-name	función que devuelve nombres de clases personalizadas para una fila, o una cadena asignando nombres de clases para cada fila	Function({row, rowIndex})/String	—	—
row-style	función que devuelve el estilo personalizado para una fila, o un objeto asignando estilos personalizado para cada fila	Function({row, rowIndex})/Object	—	—
cell-class-	función que devuelve nombres	Function({row,	—	—

name	de clases personalizadas para una celda, o una cadena asignando nombres de clases para cada celda	column, rowIndex, columnIndex})/String		
cell-style	función que devuelve estilos personalizados para una celda, o un objeto asignado a estilos personalizados para cada celda	Function({row, column, rowIndex, columnIndex})/Object	—	—
header-row-class-name	función que devuelve nombre de clases personalizadas para una fila en la cabecera de la tabla, o una cadena asignando nombres de clases para cada fila en la cabecera de la tabla	Function({row, rowIndex})/String	—	—
header-row-style	función que devuelve estilos personalizados para una fila en la cabecera de la tabla, o un objeto asignando estilos personalizados para cada fila en la cabecera de la tabla	Function({row, rowIndex})/Object	—	—
header-cell-class-name	función que devuelve nombre de clases personalizadas para una celda en la cabecera de la tabla, o una cadena asignando nombres de clases para cada celda en la cabecera de la tabla	Function({row, column, rowIndex, columnIndex})/String	—	—
header-cell-style	función que devuelve estilos personalizados para una celda en la cabecera de la tabla, o un objeto asignando estilos personalizados para cada celda en la cabecera de la tabla	Function({row, column, rowIndex, columnIndex})/Object	—	—
row-key	key de los datos de las filas, utilizada para optimizar el renderizado. Requerido si `reserve-selection` está activada o muestra los datos del árbol. Cuando su tipo es String, se admite el acceso multinivel, por ejemplo, `user.info.id`, pero `user.info[0].id` no se admite, en cuyo caso se debe utilizar la función.	Function(row)/String	—	—
empty-text	Texto mostrado cuando no existen datos. Puede	String	—	No Data

	personalizar esta área con `slot="empty"`			
default-expand-all	especifica si todas las filas se expanden por defecto, solo funciona cuando la tabla tiene una columna `type="expand"`	Boolean	—	false
expand-row-keys	establece las filas expandidas a través de esta propiedad, este valor es la clave de filas expandidas, debería establecer `row-key` antes de usar esta propiedad	Array	—	
default-sort	establece la columna y orden por defecto. La propiedad `prop` es utilizada para establecer la columna de ordenamiento por defecto, la propiedad `order` es utilizada para definir el tipo de orden por defecto	Object	`order`: ascending, descending	if `prop` is set, and `order` is not set, then `order` is default to ascending
tooltip-effect	propiedad `effect` para efectos en tooltip	String	dark/light	
show-summary	especifica si debe mostrar la fila de resumen	Boolean	—	false
sum-text	texto a mostrar para la primer columna de la fila de resumen	String	—	Sum
summary-method	método personalizado para resumen	Function({ columns, data })	—	—
span-method	método que devuelve *rowspan* y *colspan*	Function({ row, column, rowIndex, columnIndex })	—	—
select-on-indeterminate	controla el comportamiento del checkbox maestro en tablas de selección múltiple cuando sólo se seleccionan algunas filas (pero no todas). Si es true, todas las filas serán seleccionadas, de lo contrario deseleccionadas.	Boolean	—	true
indent	indentación horizontal de los datos en formato tree	Number	—	16
lazy	si se realiza un lazy loading de los datos	Boolean	—	—
load	método para cargar las filas de los hijos, solamente funciona	Function(row, treeNode, resolve)	—	—

	cuando `lazy`es true			
tree-props	configuración para renderizar datos anidados	Object	—	{ hasChildren: 'hasChildren', children: 'children' }

Eventos de la tabla

Nombre del evento	Descripción	Parámetros
select	se dispara cuando el usuario hace clic al *checkbox* (caja de selección) en una fila	selection, row
select-all	se dispara cando el usuario hace clic al *checkbox* (caja de selección) en una cabecera de la tabla	selection
selection-change	se dispara cuando selección cambia	selection
cell-mouse-enter	se dispara cuando se desplaza dentro de una celda	row, column, cell, event
cell-mouse-leave	se dispara cuando se desplaza fuera de una celda	row, column, cell, event
cell-click	se dispara cuando se hace clic en una celda	row, column, cell, event
cell-dblclick	se dispara cuando se hace doble clic en una celda	row, column, cell, event
row-click	se dispara cuando se hace clic en una fila	row, column, event
row-contextmenu	se dispara cuando el usuario hace clic derecho en una fila	row, column, event
row-dblclick	se dispara cuando se hace doble clic en una fila	row, column, event
header-click	se dispara cuando se hace clic en una cabecera de columna	column, event
header-contextmenu	se dispara cuando el usuario hace clic derecho en una cabecera de columna	column, event
sort-change	se dispara cuando el ordenamiento de la tabla cambia	{ column, prop, order }
filter-change	clave de la columna. Si necesitas utilizar el evento filter-change, este atributo es obligatorio para identificar cuál columna está siendo filtrada	filters
current-change	se dispara cuando la fila actual cambia	currentRow, oldCurrentRow

header-dragend	se dispara después de modificar el ancho de una columna arrastrando el borde de la cabecera.	newWidth, oldWidth, column, event
expand-change	se activa cuando el usuario expande o colapsa una fila (para la tabla expansible, el segundo parámetro es expandedRows; para la tabla de árbol, el segundo parámetro es expanded)	row, (expandedRows | expanded)

Métodos de la tabla

Metodo	Descripción	Parametros
clearSelection	utilizado en selección múltiple de la tabla, limpiar selección	—
toggleRowSelection	utilizado en selección múltiple de la tabla, alterna si una cierta fila es seleccionada. Con el segundo parámetro, puede directamente establecer si la fila es seleccionada	row, selected
toggleAllSelection	usado en Table de selección múltiple, cambia los estados de selección de todas las filas.	-
toggleRowExpansion	utilizado en la Tabla expandible o en la Tabla de árbol, conmutar si se desglosa una determinada línea. Con el segundo parámetro, se puede fijar directamente si esta línea se desglosa o se pliega	row, expanded
setCurrentRow	utilizado en tabla con selección sencilla, establece una cierta fila seleccionada. Si es llamado sin ningún parámetro, este puede limpiar la selección	row
clearSort	limpiar ordenamiento, restaurar datos a orden original	—
clearFilter	Se utiliza para borrar todas las condiciones de filtro cuando no se pasan parámetros, los datos se restaurarán a un estado sin filtrar, o se puede pasar una matriz de columnas para borrar las condiciones de filtro de la columna especificada.	columnKey
doLayout	refresca el layout del Table. Cuando la visibilidad de Table cambia, puede que necesite llamar a este método para obtener un diseño correcto	—
sort	Ordenar tabla manualmente. La propiedad `prop` se utiliza para establecer la columna de ordenación, la propiedad `order` se utiliza para establecer el orden.	prop: string, order: string

Slots de la tabla

Nombre	Descripción
append	El contenido será insertado después de la última fila. Es posible que necesites este espacio si deseas implementar *scroll* infinito para la tabla. Este espacio se mostrará sobre la fila de resumen si hay uno.

Atributos para las columnas de la tabla

Atributo	Descripción	Tipo	Valores aceptados	Por defecto
type	tipo de la columna. Si se establece a `selection`, la columna puede mostrar un *checkbox*. Si se establece a `index`, la columna puede mostrar el índice de la fila (iniciando desde 1). Si se establece a `expand`, la columna puede mostrar un ícono para expandir.	string	selection/index/expand	—
index	personalice los índices para cada fila, funciona en columnas con `type=index`	number, Function(index)	-	-
label	etiqueta de la columna	string	—	—
column-key	clave de la columna. Si necesita utilizar el evento `filter-change`, necesita el atributo para identificar cual columna está siendo filtrada	string	string	—
prop	nombre del campo. También puede usar su alias: `property`	string	—	—
width	ancho de la columna	string	—	—
min-width	ancho mínimo de la columna. Columnas con `width` tienen un ancho fijo, mientras que las columnas con `min-width` tienen un ancho que se distribuye en proporción.	string	—	—
fixed	especifica si la columna se fija a la izquierda o a la derecha. Se fijará a la izquierda si es `true`	string/boolean	true/left/right	—
render-header	Función de renderizado para la cabecera de la tabla de esta columna	Function(h, { column, $index })	—	—
sortable	especifica que columna puede ser ordenado. El ordenamiento remoto puede ser hecho	boolean, string	true, false, custom	false

	configurando el atributo `custom` y escucha al evento de tabla `sort-change`			
sort-method	método de ordenamiento, funciona cuando `sortable` está en `true`. Debería devolver un número, al igual que Array.sort	Function(a, b)	—	—
sort-by	especifica por cual propiedad de va a ordenar, funciona cuando `sortable` es `true` y `sort-method` es `undefined`. Si se establece a un arreglo, la columna ordenara secuencialmente por la siguiente propiedad si la anterior es igual	Function(row, index)/String/Array	—	—
sort-orders	el orden de las estrategias de ordenación utilizadas al ordenar los datos, funciona cuando `sortable` es `true`. Acepta un array, a medida que el usuario hace clic en el encabezado, la columna se ordena en el orden de los elementos del array.	array	los elementos en el arreglo necesitan ser uno de los siguientes: `ascending`, `descending` y `null` (restaura el orden original)	['ascending', 'descending', null]
resizable	especifica si el ancho de la columna puede ser redimensionado, funciona cuando `border` de `el-table` está en `true`	boolean	—	false
formatter	función que formatea el contenido de la celda	Function(row, column, cellValue, index)	—	—
show-overflow-tooltip	especifica si el *tooltip* debe ocultarse o mostrarse al hacer *hover* en la celda	boolean	—	false
align	alineación	string	left/center/right	left
header-align	alineación de la cabecera de la tabla. Si se omite, se aplicará el valor del atributo anterior.	String	left/center/right	—

class-name	nombre de clase de la celda en la columna	string	—	—
label-class-name	nombre de clase de la etiqueta de esta columna	string	—	—
selectable	función que determina si una cierta fila puede ser seleccionada, funciona cuando `type` esta en `selection`	Function(row, index)	—	—
reserve-selection	especifica si se reserva la selección después de actualizar los datos, funciona cuando `type` está en `selection`. Note que `row-key` es requerido para que esto funcione	boolean	—	false
filters	un arreglo de opciones para filtrado de datos. Para cada elemento en este arreglo, `text` y `value` son obligatorios	Array[{ text, value }]	—	—
filter-placement	colocación para el menú desplegable del filtro	String	same as Tooltip's `placement`	—
filter-multiple	especifica si el filtrado de datos soporta múltiples opciones	Boolean	—	true
filter-method	método para filtrado de datos. Si `filter-multiple` está habilitado, este método se invocará varias veces para cada fila, y una fila puede mostrar si una de las llamadas devuelve `true`	Function(value, row, column)	—	—
filtered-value	el valor del filtro para los datos seleccionados, puede ser útil cuando el encabezado de la tabla se representará con `render-header`	Array	—	—

Table-column Scoped Slot

Name	Description

—	Contenido personalizado para las columnas de la tabla. El parámetro del scope es { row, column, $index }
header	Contenido personalizado para el encabezado de la tabla. El parámetro del scope es { column, $index }