# Table

Permet d'afficher de nombreuses données possédant un format similaire. Vous pouvez ainsi trier, filtrer et comparer les données de votre tableau.

## Table basique

Un tableau de base pour afficher seulement.

:::demo Après avoir configuré l'attribut `data` de `el-table` avec un tableau d'objets, vous pouvez utiliser `prop` (correspondant à une clé dans chaque objet du tableau `data` ) dans `el-table-column` pour insérer des données dans les colonnes, ainsi que l'attribut `label` pour définir le nom de la colonne. L'attribut `width` définit si besoin la largeur de la colonne.

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

:::

### Table rayée

Ajouter des couleurs différentes pour les lignes paires et impaires permet de lire le tableau plus facilement.

:::demo L'attribut `stripe` accepte un `Boolean`. Si `true`, le tableau sera rayé.

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

:::

### Table avec bordure

:::demo Par défaut, Table n'a pas de bordure verticale. Si vous en avez besoin, mettez l'attribut `border` à `true`.

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

:::

### Table avec statut

Vous pouvez mettre en valeur certaines lignes du tableau suivant leur état, comme "success", "information", "warning", "danger" et d'autres.

:::demo Utilisez `row-class-name` dans `el-table` pour utiliser une fonction qui permettra ajouter des classes à certaines lignes. Vous pouvez ensuite définir les classes en question dans votre CSS.

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

:::

### Table avec un header fixe

Lorsqu'il y a beaucoup de lignes, il peut être utile d'avoir un header fixe afin de ne pas perdre le nom de colonnes à mesure que l'utilisateur défile vers le bas.

:::demo En réglant l'attribut `height` de `el-table`, vous pouvez fixer le header sans avoir besoin de plus de code.

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

:::

### Table avec colonnes fixes

Lorsque qu'il y a beaucoup de colonnes, il peut être utile d'en fixer certaines afin de ne pas perdre de vue leurs informations à mesure que l'utilisateur défile sur les cotés.

:::demo Utilisez l'attribut `fixed` de `el-table-column` qui accepte un `Boolean`. Si `true`, la colonne sera fixée à gauche. Il accepte aussi les valeurs 'left' et 'right', indiquant de quel coté la colonne doit être fixée.

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%">
    <el-table-column
      fixed
      prop="date"
      label="Date"
      width="150">
    </el-table-column>
    <el-table-column
      prop="name"
      label="Nom"
      width="120">
    </el-table-column>
    <el-table-column
      prop="state"
      label="État"
      width="120">
    </el-table-column>
    <el-table-column
      prop="city"
      label="Ville"
      width="120">
    </el-table-column>
    <el-table-column
      prop="address"
      label="Adresse"
      width="300">
    </el-table-column>
    <el-table-column
      prop="zip"
      label="Zip"
```

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

:::

### Table avec header et colonnes fixes

Si vous avez un gros volume de données à afficher, vous pouvez fixer le header et des colonnes en même temps.

:::demo Header et colonnes fixes combinant les deux exemples précédents.

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%"
    height="250">
    <el-table-column
      fixed
      prop="date"
      label="Date"
      width="150">
    </el-table-column>
    <el-table-column
      prop="name"
      label="Nom"
      width="120">
    </el-table-column>
    <el-table-column
      prop="state"
      label="État"
      width="120">
    </el-table-column>
    <el-table-column
      prop="city"
      label="Ville"
      width="120">
    </el-table-column>
    <el-table-column
      prop="address"
      label="Adresse"
      width="300">
    </el-table-column>
    <el-table-column
      prop="zip"
      label="Zip"
      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>
```

:::

### Hauteur fluide dans une Table avec header et colonnes fixes

Quand les données changent dynamiquement, vous pouvez avoir besoin d'une hauteur maximale et d'afficher une barre de défilement si besoin.

:::demo En utilisant l'attribut `max-height` de `el-table`, vous fixez le header. Le tableau ne défilera que si la hauteur des lignes dépasse cette hauteur maximale.

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%"
    max-height="250">
    <el-table-column
      fixed
      prop="date"
      label="Date"
      width="150">
    </el-table-column>
    <el-table-column
      prop="name"
      label="Nom"
      width="120">
    </el-table-column>
    <el-table-column
      prop="state"
      label="État"
      width="120">
    </el-table-column>
    <el-table-column
      prop="city"
      label="Ville"
      width="120">
    </el-table-column>
    <el-table-column
      prop="address"
      label="Adresse"
      width="300">
    </el-table-column>
```

```
    <el-table-column
      prop="zip"
      label="Zip"
      width="120">
    </el-table-column>
    <el-table-column
      fixed="right"
      label="Opérations"
      width="120">
      <template slot-scope="scope">
        <el-button
          @click.native.prevent="deleteRow(scope.$index, tableData)"
          type="text"
          size="small">
          Supprimer
        </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>
```

:::

### Grouper les headers

Quand la structure du tableau est complexe, vous pouvez grouper les headers afin de montrer une hiérarchie.

:::demo Placez simplement `el-table-column` dans une autre `el-table-column`, et vous grouperez ainsi les headers.

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%">
    <el-table-column
      prop="date"
      label="Date"
      width="150">
    </el-table-column>
    <el-table-column label="Informations de livraison">
```

```
      <el-table-column
        prop="name"
        label="Nom"
        width="120">
      </el-table-column>
      <el-table-column label="Informations d'adresse">
        <el-table-column
          prop="state"
          label="État"
          width="120">
        </el-table-column>
        <el-table-column
          prop="city"
          label="Ville"
          width="120">
        </el-table-column>
        <el-table-column
          prop="address"
          label="Adresse"
          width="300">
        </el-table-column>
        <el-table-column
          prop="zip"
          label="Zip"
          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>
```

:::

### Sélecteur simple

Vous pouvez activer la sélection d'une ligne.

:::demo Activez la sélection en ajoutant l'attribut `highlight-current-row`. L'évènement `current-change` se déclenchera à chaque changement de sélection, ses paramètres étant les deux lignes avant et après le changement: `currentRow` et `oldCurrentRow`. Si vous avez besoin d'afficher l'index des lignes, ajoutez une `el-table-column` avec pour `type` la valeur `index`, qui commencera à 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="Date"
      width="120">
    </el-table-column>
    <el-table-column
      property="name"
      label="Nom"
      width="120">
    </el-table-column>
    <el-table-column
      property="address"
      label="Adresse">
    </el-table-column>
  </el-table>
  <div style="margin-top: 20px">
    <el-button @click="setCurrent(tableData[1])">Sélectionner la deuxième ligne</el-
button>
    <el-button @click="setCurrent()">Effacer la sélection</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>
```

:::

### Sélection multiple

Vous pouvez aussi sélectionner plusieurs lignes.

:::demo Ajoutez une `el-table-column` avec son `type` à `selection`. Cet exemple utilise aussi `show-overflow-tooltip` : par défaut, si le contenu est trop long, il se séparera en plusieurs lignes. Si vous souhaitez conserver une seule ligne, utilisez `show-overflow-tooltip`, qui accepte un `Boolean`. Si `true`, le contenu non affiché s'affichera dans le tooltip lorsque la souris passera sur la case.

```
<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="Date"
      width="120">
      <template slot-scope="scope">{{ scope.row.date }}</template>
    </el-table-column>
    <el-table-column
      property="name"
      label="Nom"
      width="120">
    </el-table-column>
    <el-table-column
      property="address"
      label="Adresse"
      show-overflow-tooltip>
    </el-table-column>
  </el-table>
  <div style="margin-top: 20px">
```

```
    <el-button @click="toggleSelection([tableData[1], tableData[2]])">Sélectionner
les deuxième et troisième lignes</el-button>
    <el-button @click="toggleSelection()">Effacer la sélection</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>
```

:::

### Trier

Il est possible de trier les données afin de trouver plus facilement ce qu'on cherche.

:::demo Ajoutez l'attribut `sortable` a une colonne pour pouvoir trier cette colonne. Il accepte un `Boolean` et à `false` par défaut. Ajoutez `default-sort` pour déterminer les propriétés par défaut du tri. Pour appliquer vos propres règles, utilisez `sort-method` ou `sort-by`. Si vous avez besoin d'un tri dynamique depuis le serveur, mettez `sortable` à `custom`, et écoutez l'évènement `sort-change` de `Table`. Depuis cet évènement, vous aurez accès à la colonne et l'ordre de tri. cet exemple utilise aussi `formatter` pour formater les valeurs de colonnes. Il prend une fonction avec deux paramètres: `row` et `column`. Vous pouvez ainsi transformer les valeurs.

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

:::

### Filtrer

Vous pouvez filtrer la table pour obtenir rapidement les lignes désirées.

:::demo Réglez `filters` et `filter-method` dans `el-table-column` pour rendre la colonne filtrable. `filters` prends un tableau, et `filter-method` est une fonction déterminant comment les lignes s'affichent. Elle prend trois paramètres: `value`, `row` et `column`.

```
<template>
  <el-button @click="resetDateFilter">Effacer le filtre date</el-button>
  <el-button @click="clearFilter">Effacer tout les filtres</el-button>
  <el-table
    ref="filterTable"
    :data="tableData"
    style="width: 100%">
    <el-table-column
      prop="date"
      label="Date"
      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="Nom"
      width="180">
    </el-table-column>
    <el-table-column
      prop="address"
      label="Adresse"
      :formatter="formatter">
    </el-table-column>
    <el-table-column
      prop="tag"
      label="Tag"
      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>
```

:::

### Template de colonne personnalisé

Vous pouvez customiser le contenu des colonnes afin de pouvoir utiliser d'autres composants.

:::demo Vous avez acccès aux données suivantes: `row`, `column`, `$index` et `store` (gestionnaire d'état de Table) grâce aux Scoped slot.

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%">
    <el-table-column
      label="Date"
      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="Nom"
      width="180">
      <template slot-scope="scope">
        <el-popover trigger="hover" placement="top">
          <p>Nom: {{ 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="Opérations">
      <template slot-scope="scope">
        <el-button
          size="mini"
          @click="handleEdit(scope.$index, scope.row)">Editer</el-button>
        <el-button
          size="mini"
          type="danger"
          @click="handleDelete(scope.$index, scope.row)">Supprimer</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 avec header personnalisé

Vous pouvez également personnaliser le header de la table.

:::demo Vous pouvez personnaliser le header grâce aux slots avec portée.

```
<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.$index, scope.row)">Editer</el-button>
        <el-button
          size="mini"
          type="danger"
          @click="handleDelete(scope.$index, scope.row)">Supprimer</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: 'John',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-04',
          name: 'Morgan',
          address: 'No. 189, Grove St, Los Angeles'
        }, {
          date: '2016-05-01',
          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>
```

:::

### Ligne extensible

Lorsque le contenu d'une ligne est trop long et que vous ne souhaitez pas afficher de scrollbar, vous pouvez utiliser une ligne extensible.

:::demo Activer la ligne extensible en mettant `type` à `expand` et en utilisant un slot. Le contenu de `el-table-column` sera généré quand l'extension aura lieu, et vous avez aux même attributs que pour le slot des templates de colonnes.

```
<template>
  <el-table
    :data="tableData"
    style="width: 100%">
    <el-table-column type="expand">
      <template slot-scope="props">
        <p>État: {{ props.row.state }}</p>
        <p>Ville: {{ props.row.city }}</p>
        <p>Adresse: {{ props.row.address }}</p>
        <p>Zip: {{ props.row.zip }}</p>
      </template>
    </el-table-column>
```

```
    <el-table-column
      label="Date"
      prop="date">
    </el-table-column>
    <el-table-column
      label="Nom"
      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>
```

:::

### Arborescence et lazy loading

:::demo You can display tree structure data. When row contains the `children` field, it is treated as nested data. For rendering nested data, the prop `row-key` is required。Also, child row data can be loaded asynchronously. Set `lazy` property of Table to true and the function `load`. Specify `hasChildren` attribute in row to determine which row contains children. Both `children` and `hasChildren` can be configured via `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="date"
      sortable
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="Nom"
      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="Nom"
      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>
```

:::

### Ligne de somme

Pour les tableaux de nombres, vous pouvez ajouter une ligne en plus pour afficher la somme de chaque colonne.

:::demo Mettez l'attribut `show-summary` à `true` dans `el-table`. Par défaut, la première colonne n'affiche que 'Sum' (vous pouvez configurer ce label avec `sum-text`), alors que les autres affichent la somme de chaque colonne. Vous pouvez choisir comment effectuer l'opération grâce à `summary-method`, qui doit retourner un tableau, cahque élément correspondant à la somme de chaque colonne. La deuxième table de cet exemple montre tout cela en pratique.

```
<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="Nom">
    </el-table-column>
    <el-table-column
      prop="amount1"
      sortable
      label="Quantité 1">
    </el-table-column>
    <el-table-column
      prop="amount2"
      sortable
      label="Quantité 2">
    </el-table-column>
    <el-table-column
      prop="amount3"
      sortable
      label="Quantité 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="Nom">
    </el-table-column>
    <el-table-column
      prop="amount1"
      label="Coût 1 ($)">
    </el-table-column>
    <el-table-column
      prop="amount2"
      label="Coût 2 ($)">
```

```
    </el-table-column>
    <el-table-column
      prop="amount3"
      label="Coût 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] = 'Coût 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>
```

:::

### Étendue des lignes et colonnes

Vous pouvez configurer l'étendue des lignes et colonnes afin de fusionner des cellules.

:::demo Utilisez `span-method` pour configurer chaque étendue. Il accepte une fonction, et lui passe un objet incluant la ligne actuelle `row`, la colonne actuelle `column`, l'index de la ligne `rowIndex` et l'index de la colonne `columnIndex`. La fonction doit retourner un tableau contenant deux nombres, le premier étant `rowspan` et le second `colspan`. Elle peut aussi retourner un objet avec les propriétés `rowspan` et `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="Nom">
      </el-table-column>
      <el-table-column
        prop="amount1"
```

```
        sortable
        label="Quantité 1">
      </el-table-column>
      <el-table-column
        prop="amount2"
        sortable
        label="Quantité 2">
      </el-table-column>
      <el-table-column
        prop="amount3"
        sortable
        label="Quantité 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="Nom">
      </el-table-column>
      <el-table-column
        prop="amount1"
        label="Quantité 1">
      </el-table-column>
      <el-table-column
        prop="amount2"
        label="Quantité 2">
      </el-table-column>
      <el-table-column
        prop="amount3"
        label="Quantité 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>
```

:::

### Indices personnalisés

Vous pouvez personnaliser les indices des colonnes de type `index`.

:::demo Utilisez l'attribut `index` sur une `el-table-column` avec `type=index`. Si un nombre est assigné, tout les indices auront un décalage égal à ce nombre. Il peut aussi prendre une fonction avec chaque indice (commençant à `0`) comme paramètre, et la valeur de retour sera affichée en tant qu'indice.

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

:::

### Attributs de Table

| Attribut | Description | Type | Valeurs acceptées | Défaut |
|---|---|---|---|---|
| data | Les données de la table. | array | — | — |
| height | La hauteur de la table. Par défaut la hauteur est `auto`. Si sa valeur est un nombre, la hauteur est en px; si c'est un string, la valeur est assigné au | string/number | — | — |

| | | | | |
|---|---|---|---|---|
| | style.height de l'élement. La hauteur est affectée par les styles externes. | | | |
| max-height | Table's max-height. The legal value is a number or the height in px. | string/number | — | — |
| stripe | Si la table est rayée. | boolean | — | false |
| border | Si la table à une bordure verticale. | boolean | — | false |
| size | Taille de la table. | string | medium / small / mini | — |
| fit | Si la largeur des colonnes s'adapte au conteneur. | boolean | — | true |
| show-header | Si le header de la table est visible. | boolean | — | true |
| highlight-current-row | Si la ligne courante est mise en valeur. | boolean | — | false |
| current-row-key | Clé de la ligne actuelle. Propriété set-only. | string,number | — | — |
| row-class-name | Fonction qui retourne un nom de classe pour chaque ligne. Peut aussi être une simple chaîne de caractères assignant une classe à chaque ligne. | Function({row, rowIndex})/String | — | — |
| row-style | Fonction qui retourne un style pour chaque ligne. Peut aussi être un objet assignant un style à chaque ligne. | Function({row, rowIndex})/Object | — | — |
| cell-class-name | Fonction qui retourne un nom de classe pour chaque cellule. Peut aussi être une simple chaîne de caractères assignant une classe à chaque cellule. | Function({row, column, rowIndex, columnIndex})/String | — | — |
| cell-style | Fonction qui retourne un style pour chaque cellule. Peut aussi être un objet assignant un style à chaque cellule. | Function({row, column, rowIndex, columnIndex})/Object | — | — |
| header-row-class-name | Fonction qui retourne un nom de classe pour chaque ligne de header. Peut aussi être une simple chaîne de caractères | Function({row, rowIndex})/String | — | — |

| | | | | |
|---|---|---|---|---|
| | assignant une classe à chaque ligne de header. | | | |
| header-row-style | Fonction qui retourne un style pour chaque ligne de header. Peut aussi être un objet assignant un style à chaque ligne de header. | Function({row, rowIndex})/Object | — | — |
| header-cell-class-name | Fonction qui retourne un nom de classe pour chaque cellule de header. Peut aussi être une simple chaîne de caractères assignant une classe à chaque cellule de header. | Function({row, column, rowIndex, columnIndex})/String | — | — |
| header-cell-style | Fonction qui retourne un style pour chaque cellule de header. Peut aussi être un objet assignant un style à chaque cellule de header. | Function({row, column, rowIndex, columnIndex})/Object | — | — |
| row-key | Clé de chaque ligne, utilisée pour optimiser le rendu. Requise si `reserve-selection` est activé. Quand c'est un `String`, l'accès multi-niveaux est supporté, e.g. `user.info.id`, mais `user.info[0].id` n'est pas supporté. Dans ce dernier cas une `Function` devrait être utilisée. | Function(row)/String | — | — |
| empty-text | Texte à afficher quand il n'y a pas de données. Vous pouvez changer cette zone grâce à `slot="empty"`. | String | — | No Data |
| default-expand-all | whether expand all rows by default, works when the table has a column type="expand" or contains tree structure data | Boolean | — | false |
| expand-row-keys | Détermine les lignes qui sont étendues, contient les clés des lignes correspondantes. Vous devriez configurer `row-key` avant celle-ci. | Array | — | |
| default-sort | Détermine l'ordre de tri par défaut. La propriété `prop` détermine la colonne par | Object | `order`: ascending, descending | Si `order` est absent, son défaut sera `ascending`. |

| | | | | |
|---|---|---|---|---|
| | défaut, `order` détermine l'ordre par défaut. | | | |
| tooltip-effect | Propriété `effect` de Tooltip. | String | dark/light | |
| show-summary | Si une ligne de somme doit apparaître. | Boolean | — | false |
| sum-text | Le label de la première cellule de la ligne de somme. | String | — | Sum |
| summary-method | La méthode pour calculer la somme. | Function({ columns, data }) | — | — |
| span-method | Méthode qui retourne les valeurs de colspan et rowspan. | Function({ row, column, rowIndex, columnIndex }) | — | — |
| select-on-indeterminate | Contrôle le comportement de la checkbox globale dans les tables avec sélection multiple lorsque seulement certaines lignes sont sélectionnées. Si `true`, toutes les lignes sont sélectionnées. | Boolean | — | true |
| indent | horizontal indentation of tree data | Number | — | 16 |
| lazy | whether to lazy loading data | Boolean | — | — |
| load | method for loading child row data, only works when `lazy` is true | Function({ row, treeNode, resolve }) | — | — |
| tree-props | configuration for rendering nested data | Object | — | { hasChildren: 'hasChildren', children: 'children' } |

### Évènements de Table

| Nom | Description | Paramètres |
|---|---|---|
| select | Se déclenche quand l'utilisateur clique sur la checkbox d'une ligne. | selection, row |
| select-all | Se déclenche quand l'utilisateur clique sur la checkbox du header. | selection |
| selection-change | Se déclenche quand la selection change. | selection |
| cell-mouse-enter | Se déclenche quand la souris passe sur une cellule. | row, column, cell, event |
| cell-mouse- | Se déclenche quand la souris sort d'une cellule. | row, column, |

| | | |
|---|---|---|
| leave | | cell, event |
| cell-click | Se déclenche quand l'utilisateur clique sur une cellule. | row, column, cell, event |
| cell-dblclick | Se déclenche quand l'utilisateur double-clique sur une cellule. | row, column, cell, event |
| row-click | Se déclenche quand l'utilisateur clique sur une ligne. | row, column, event |
| row-contextmenu | Se déclenche quand l'utilisateur fait un clic droit sur une ligne. | row, column, event |
| row-dblclick | Se déclenche quand l'utilisateur double-clique sur une ligne. | row, column, event |
| header-click | Se déclenche quand l'utilisateur clique sur une colonne du header. | column, event |
| header-contextmenu | Se déclenche quand l'utilisateur fait un clic droit sur une colonne du header. | column, event |
| sort-change | Se déclenche quand l'ordre de tri change. | { column, prop, order } |
| filter-change | column's key. If you need to use the filter-change event, this attribute is mandatory to identify which column is being filtered. | filters |
| current-change | Se déclenche quand la ligne sélectionnée change. | currentRow, oldCurrentRow |
| header-dragend | Se déclenche après un changement de taille de colonne en déplaçant la ligne verticale du header. | newWidth, oldWidth, column, event |
| expand-change | triggers when user expands or collapses a row (for expandable table, second param is expandedRows; for tree Table, second param is expanded) | row, (expandedRows \| expanded) |

### Méthodes de Table

| Méthode | Description | Paramètres |
|---|---|---|
| clearSelection | Dans les tables avec sélection multiple, efface la sélection. | — |
| toggleRowSelection | Dans les tables avec sélection multiple, change la sélection d'une ligne. Grâce au deuxième paramètre vous pouvez directement décider si cette ligne est sélectionnée. | row, selected |
| toggleAllSelection | Utilisé dans les tables à sélection multiples, sélectionne toutes les lignes. | - |
| toggleRowExpansion | used in expandable Table or tree Table, toggle if a certain row is expanded. With the second parameter, you can directly set if this row is expanded or collapsed | row, expanded |
| setCurrentRow | Dans les tables à sélection simple, sélectionne une ligne. Sans | row |

| | | |
|---|---|---|
| | paramètre la sélection est effacé. | |
| clearSort | Efface le tri. | — |
| clearFilter | Efface les filtres des colonnes dont les `columnKey` sont passées. Si aucun paramètre, efface tout les filtres. | columnKeys |
| doLayout | Rafraîchi le layout de la table. Quand la visibilité de la table change vous aurez peut-être besoin de cette méthode pour corriger le layout. | — |
| sort | Tri la table manuellement. La proriété `prop` détermine la colonne, `order` détermine l'ordre de tri. | prop: string, order: string |

### Slots de Table

| Nom | Description |
|---|---|
| append | Contenu à insérer après la dernière ligne. Vous aurez sans doute besoin de ce slot si vous implémentez un scroll infini. Il sera affiché au-dessus de la ligne de somme s'il y en a une. |

### Attributs de Table-column

| Attribut | Description | Type | Valeurs acceptées | Défaut |
|---|---|---|---|---|
| type | Type de la colonne. Si mis à `selection`, la colonne affichera des checkbox. Si mis à `index`, la colonne affichera l'indice de la ligne (début à 1). Si mis à `expand`, affichera l'icône d'extension. | string | selection/index/expand | — |
| index | Personnalise les indices de chaque ligne, marche avec les colonnes `type=index`. | number, Function(index) | - | - |
| label | Label de la colonne. | string | — | — |
| column-key | La clé de la colonne. Si vous avez besoin d'utiliser l'évènement filter-change, vous aurez besoin de cet attribut pour savoir quelle colonne est filtrée. | string | string | — |
| prop | Nom du champ de l'objet de données. Alias: `property`. | string | — | — |
| width | Largeur de la colonne. | string | — | — |
| min-width | Largeur minimale de la | string | — | — |

| | | | | |
|---|---|---|---|---|
| | colonne. Les colonnes avec `width` ont une largeur fixe, alors que celles avec `min-width` ont une largeur proportionnellement distribuée. | | | |
| fixed | Si la colonne est fixée à droite ou à gauche. Fixée à gauche si `true`. | string/boolean | true/left/right | — |
| render-header | Fonction de rendu pour le header de cette colonne. | Function(h, { column, $index }) | — | — |
| sortable | Si la colonne peut être triée. Tri dynamique possible en mettant à 'custom' et en écoutant l'évènement `sort-change` de Table. | boolean, string | true, false, custom | false |
| sort-method | Méthode de tri, marche quand `sortable` est `true`. Doit retourner un nombre, tout comme Array.sort. | Function(a, b) | — | — |
| sort-by | Détermine par quelle propriété effectuer le tri, marche quand `sortable` est `true` et `sort-method` est `undefined`. Si c'est un Array, sera triée par la propriété suivante si la précédente est équivalente. | Function(row, index)/String/Array | — | — |
| sort-orders | Liste des stratégies de tri, marche quand `sortable` est `true`. Accepte un tableau. Lorsque l'utilisateur clique plusieurs fois sur le header, la colonne est triée dans l'ordre des stratégies indiquée. | array | Les élements du tableau doivent être parmi: `ascending`, `descending` et `null` (restaure l'état originel du tableau). | ['ascending', 'descending', null] |
| resizable | Si la largeur de la colonne peut être modifiée, marche quand `border` de `el-table` est `true`. | boolean | — | false |
| formatter | Fonction pour formater le contenu des cellules. | Function(row, column, cellValue, | — | — |

| | | | | |
|---|---|---|---|---|
| | | index) | | |
| show-overflow-tooltip | Si du contenu trop long doit être caché et affiché dans une tooltip quand la souris passe sur la cellule. | boolean | — | false |
| align | Alignement. | string | left/center/right | left |
| header-align | Alignement du header. Si omis, la valeur du `align` ci-dessus est appliqué. | String | left/center/right | — |
| class-name | Classe des cellules dans cette colonne. | string | — | — |
| label-class-name | Classe du label de cette colonne. | string | — | — |
| selectable | Détermine si certaines colonnes peuvent être sélectionnées, marche quand `type` est 'selection'. | Function(row, index) | — | — |
| reserve-selection | Si la sélection doit être conservée après rafraîchissement, marche quand `type` est 'selection'. Notez que `row-key` est requis. | boolean | — | false |
| filters | Un tableau d'options de filtrage. Pour chaque élément, `text` et `value` sont requis. | Array[{ text, value }] | — | — |
| filter-placement | Emplacement du menu du filtre. | String | Voir `placement` de Tooltip. | — |
| filter-multiple | Si le filtrage supporte plusieurs options. | Boolean | — | true |
| filter-method | Méthode de filtrage. Si `filter-multiple` est activé, cette méthode sera appelé plusieurs fois pour chaque ligne, qui sera affichée si dès qu'un `true` sera renvoyé. | Function(value, row, column) | — | — |
| filtered-value | Valeur de filtre pour les colonnes sélectionnées, peut être utile quand le | Array | — | — |

| | | | | |
|---|---|---|---|---|
| | header est rendu avec `render-header`. | | | |

### Slot de Table-column

| Nom | Description |
|---|---|
| — | Contenu personnalisé pour les colonnes. Les paramètres sont { row, column, $index } |
| header | Contenu personnalisé pour le header. Le paramètre de scope est { column, $index } |