Transfer 穿梭框

基础用法

:::demo Transfer 的数据通过 `data` 属性传入。数据需要是一个对象数组，每个对象有以下属性：`key` 为数据的唯一性标识，`label` 为显示文本，`disabled` 表示该项数据是否禁止转移。目标列表中的数据项会同步到绑定至 `v-model` 的变量，值为数据项的 `key` 所组成的数组。当然，如果希望在初始状态时目标列表不为空，可以像本例一样为 `v-model` 绑定的变量赋予一个初始值。

```
<template>
  <el-transfer v-model="value" :data="data"></el-transfer>
</template>

<script>
  export default {
    data() {
      const generateData = _ => {
        const data = [];
        for (let i = 1; i <= 15; i++) {
          data.push({
            key: i,
            label: `备选项 ${ i }`,
            disabled: i % 4 === 0
          });
        }
        return data;
      };
      return {
        data: generateData(),
        value: [1, 4]
      };
    }
  };
</script>
```

:::

可搜索

在数据很多的情况下，可以对数据进行搜索和过滤。

:::demo 设置 `filterable` 为 `true` 即可开启搜索模式。默认情况下，若数据项的 `label` 属性包含搜索关键字，则会在搜索结果中显示。你也可以使用 `filter-method` 定义自己的搜索逻辑。`filter-method` 接收一个方法，当搜索关键字变化时，会将当前的关键字和每个数据项传给该方法。若方法返回 `true`，则会在搜索结果中显示对应的数据项。

```
<template>
  <el-transfer
    filterable
    :filter-method="filterMethod"
    filter-placeholder="请输入城市拼音"
```

```
    v-model="value"
    :data="data">
  </el-transfer>
</template>

<script>
  export default {
    data() {
      const generateData = _ => {
        const data = [];
        const cities = ['上海', '北京', '广州', '深圳', '南京', '西安', '成都'];
        const pinyin = ['shanghai', 'beijing', 'guangzhou', 'shenzhen', 'nanjing',
'xian', 'chengdu'];
        cities.forEach((city, index) => {
          data.push({
            label: city,
            key: index,
            pinyin: pinyin[index]
          });
        });
        return data;
      };
      return {
        data: generateData(),
        value: [],
        filterMethod(query, item) {
          return item.pinyin.indexOf(query) > -1;
        }
      };
    }
  };
</script>
```

:::

可自定义

可以对列表标题文案、按钮文案、数据项的渲染函数、列表底部的勾选状态文案、列表底部的内容区等进行自定义。

:::demo 可以使用 `titles`、`button-texts`、`render-content` 和 `format` 属性分别对列表标题文案、按钮文案、数据项的渲染函数和列表顶部的勾选状态文案进行自定义。数据项的渲染还可以使用 `scoped-slot` 进行自定义。对于列表底部的内容区，提供了两个具名 slot：`left-footer` 和 `right-footer`。此外，如果希望某些数据项在初始化时就被勾选，可以使用 `left-default-checked` 和 `right-default-checked` 属性。最后，本例还展示了 `change` 事件的用法。注意：由于 jsfiddle 不支持 JSX 语法，所以使用 `render-content` 自定义数据项的例子在 jsfiddle 中无法运行。但是在实际的项目中，只要正确地配置了相关依赖，就可以正常运行。

```
<template>
  <p style="text-align: center; margin: 0 0 20px">使用 render-content 自定义数据项</p>
  <div style="text-align: center">
    <el-transfer
      style="text-align: left; display: inline-block"
```

```
      v-model="value"
      filterable
      :left-default-checked="[2, 3]"
      :right-default-checked="[1]"
      :render-content="renderFunc"
      :titles="['Source', 'Target']"
      :button-texts="['到左边', '到右边']"
      :format="{
        noChecked: '${total}',
        hasChecked: '${checked}/${total}'
      }"
      @change="handleChange"
      :data="data">
      <el-button class="transfer-footer" slot="left-footer" size="small">操作</el-
button>
      <el-button class="transfer-footer" slot="right-footer" size="small">操作</el-
button>
    </el-transfer>
  </div>
  <p style="text-align: center; margin: 50px 0 20px">使用 scoped-slot 自定义数据项</p>
  <div style="text-align: center">
    <el-transfer
      style="text-align: left; display: inline-block"
      v-model="value4"
      filterable
      :left-default-checked="[2, 3]"
      :right-default-checked="[1]"
      :titles="['Source', 'Target']"
      :button-texts="['到左边', '到右边']"
      :format="{
        noChecked: '${total}',
        hasChecked: '${checked}/${total}'
      }"
      @change="handleChange"
      :data="data">
      <span slot-scope="{ option }">{{ option.key }} - {{ option.label }}</span>
      <el-button class="transfer-footer" slot="left-footer" size="small">操作</el-
button>
      <el-button class="transfer-footer" slot="right-footer" size="small">操作</el-
button>
    </el-transfer>
  </div>
</template>

<style>
  .transfer-footer {
    margin-left: 20px;
    padding: 6px 5px;
  }
</style>

<script>
```

```
  export default {
    data() {
      const generateData = _ => {
        const data = [];
        for (let i = 1; i <= 15; i++) {
          data.push({
            key: i,
            label: `备选项 ${ i }`,
            disabled: i % 4 === 0
          });
        }
        return data;
      };
      return {
        data: generateData(),
        value: [1],
        value4: [1],
        renderFunc(h, option) {
          return <span>{ option.key } - { option.label }</span>;
        }
      };
    },

    methods: {
      handleChange(value, direction, movedKeys) {
        console.log(value, direction, movedKeys);
      }
    }
  };
</script>
```

:::

数据项属性别名

默认情况下，Transfer 仅能识别数据项中的 `key`、`label` 和 `disabled` 字段。如果你的数据的字段名不同，可以使用 `props` 属性为它们设置别名。 :::demo 本例中的数据源没有 `key` 和 `label` 字段，在功能上与它们相同的字段名为 `value` 和 `desc`。因此可以使用 `props` 属性为 `key` 和 `label` 设置别名。

```
<template>
  <el-transfer
    v-model="value"
    :props="{
      key: 'value',
      label: 'desc'
    }"
    :data="data">
  </el-transfer>
</template>

<script>
```

```
  export default {
    data() {
      const generateData = _ => {
        const data = [];
        for (let i = 1; i <= 15; i++) {
          data.push({
            value: i,
            desc: `备选项 ${ i }`,
            disabled: i % 4 === 0
          });
        }
        return data;
      };
      return {
        data: generateData(),
        value: []
      };
    }
  };
</script>
```

:::

Attributes

参数	说明	类型	可选值	默认值
value / v-model	绑定值	array	—	—
data	Transfer 的数据源	array[{ key, label, disabled }]	—	[ ]
filterable	是否可搜索	boolean	—	false
filter-placeholder	搜索框占位符	string	—	请输入搜索内容
filter-method	自定义搜索方法	function	—	—
target-order	右侧列表元素的排序策略：若为 `original`，则保持与数据源相同的顺序；若为 `push`，则新加入的元素排在最后；若为 `unshift`，则新加入的元素排在最前	string	original / push / unshift	original
titles	自定义列表标题	array	—	['列表 1', '列表 2']
button-texts	自定义按钮文案	array	—	[ ]
render-content	自定义数据项渲染函数	function(h, option)	—	—

format	列表顶部勾选状态文案	object{noChecked, hasChecked}	—	{ noChecked: '${checked}/${total}', hasChecked: '${checked}/${total}' }
props	数据源的字段别名	object{key, label, disabled}	—	—
left-default-checked	初始状态下左侧列表的已勾选项的 key 数组	array	—	[ ]
right-default-checked	初始状态下右侧列表的已勾选项的 key 数组	array	—	[ ]

Slot

name	说明
left-footer	左侧列表底部的内容
right-footer	右侧列表底部的内容

Scoped Slot

name	说明
—	自定义数据项的内容，参数为 { option }

Methods

方法名	说明	参数
clearQuery	清空某个面板的搜索关键词	'left' / 'right'，指定需要清空的面板

Events

事件名称	说明	回调参数
change	右侧列表元素变化时触发	当前值、数据移动的方向（'left' / 'right'）、发生移动的数据 key 数组
left-check-change	左侧列表元素被用户选中 / 取消选中时触发	当前被选中的元素的 key 数组、选中状态发生变化的元素的 key 数组
right-check-change	右侧列表元素被用户选中 / 取消选中时触发	当前被选中的元素的 key 数组、选中状态发生变化的元素的 key 数组