Transfer

Basic usage

:::demo Data is passed to Transfer via the `data` attribute. The data needs to be an object array, and each object should have these attributes: `key` being the identification of the data item, `label` being the displayed text, and `disabled` indicating if the data item is disabled. Items inside the target list are in sync with the variable binding to `v-model`, and the value of that variable is an array of target item keys. So, if you don't want the target list be initially empty, you can initialize the `v-model` with an array.

```
<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: `Option ${ i }`,
            disabled: i % 4 === 0
          });
        }
        return data;
      };
      return {
        data: generateData(),
        value: [1, 4]
      };
    }
  };
</script>
```

:::

Filterable

You can search and filter data items.

:::demo Set the `filterable` attribute to `true` to enable filter mode. By default, if the data item `label` contains the search keyword, it will be included in the search result. Also, you can implement you own filter method with the `filter-method` attribute. It takes a method and passes search keyword and each data item to it whenever the keyword changes. For a certain data item, if the method returns true, it will be included in the result list.

```
<template>
  <el-transfer
    filterable
    :filter-method="filterMethod"
    filter-placeholder="State Abbreviations"
    v-model="value"
    :data="data">
  </el-transfer>
</template>

<script>
  export default {
    data() {
      const generateData = _ => {
        const data = [];
        const states = ['California', 'Illinois', 'Maryland', 'Texas', 'Florida',
'Colorado', 'Connecticut '];
        const initials = ['CA', 'IL', 'MD', 'TX', 'FL', 'CO', 'CT'];
        states.forEach((city, index) => {
          data.push({
            label: city,
            key: index,
            initial: initials[index]
          });
        });
        return data;
      };
      return {
        data: generateData(),
        value: [],
        filterMethod(query, item) {
          return item.initial.toLowerCase().indexOf(query.toLowerCase()) > -1;
        }
      };
    }
  };
</script>
```

:::

Customizable

You can customize list titles, button texts, render function for data items, checking status texts in list footer and list footer contents.

:::demo Use `titles`, `button-texts`, `render-content` and `format` to respectively customize list titles, button texts, render function for data items, checking status texts in list header. Plus, you can also use scoped slot to customize data items. For list footer contents, two named slots are provided: `left-footer` and `right-footer`. Plus, if you want some items initially checked, you can use `left-default-checked` and `right-default-checked`. Finally, this example demonstrate the `change` event. Note that this demo can't run in jsfiddle because it doesn't support JSX syntax. In a real project, `render-content` will work if relevant dependencies are correctly configured.

```
<template>
  <p style="text-align: center; margin: 0 0 20px">Customize data items using render-
content</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="['To left', 'To right']"
      :format="{
        noChecked: '${total}',
        hasChecked: '${checked}/${total}'
      }"
      @change="handleChange"
      :data="data">
      <el-button class="transfer-footer" slot="left-footer"
size="small">Operation</el-button>
      <el-button class="transfer-footer" slot="right-footer"
size="small">Operation</el-button>
    </el-transfer>
    <p style="text-align: center; margin: 50px 0 20px">Customize data items using
scoped slot</p>
    <div style="text-align: center">
      <el-transfer
        style="text-align: left; display: inline-block"
        v-model="value4"
        filterable
        :left-default-checked="[2, 3]"
        :right-default-checked="[1]"
        :titles="['Source', 'Target']"
        :button-texts="['To left', 'To right']"
        :format="{
          noChecked: '${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">Operation</el-button>
        <el-button class="transfer-footer" slot="right-footer"
size="small">Operation</el-button>
      </el-transfer>
    </div>
  </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: `Option ${ 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>
```

:::

Prop aliases

By default, Transfer looks for `key` , `label` and `disabled` in a data item. If your data items have different key names, you can use the `props` attribute to define aliases. :::demo The data items in this example do not have `key` s or `label` s, instead they have `value` s and `desc` s. So you need to set aliases for `key` and `label` .

```
<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: `Option ${ i }`,
            disabled: i % 4 === 0
          });
        }
        return data;
      };
      return {
        data: generateData(),
        value: []
      };
    }
  };
</script>
```

:::

Attributes

Attribute	Description	Type	Accepted Values	Default
value / v-model	binding value	array	—	—
data	data source	array[{ key, label, disabled }]	—	[ ]
filterable	whether Transfer is filterable	boolean	—	false
filter-placeholder	placeholder for the filter input	string	—	Enter keyword
filter-method	custom filter method	function	—	—

target-order	order strategy for elements in the target list. If set to `original`, the elements will keep the same order as the data source. If set to `push`, the newly added elements will be pushed to the bottom. If set to `unshift`, the newly added elements will be inserted on the top	string	original / push / unshift	original
titles	custom list titles	array	—	['List 1', 'List 2']
button-texts	custom button texts	array	—	[ ]
render-content	custom render function for data items	function(h, option)	—	—
format	texts for checking status in list header	object{noChecked, hasChecked}	—	{ noChecked: '${checked}/${total}', hasChecked: '${checked}/${total}' }
props	prop aliases for data source	object{key, label, disabled}	—	—
left-default-checked	key array of initially checked data items of the left list	array	—	[ ]
right-default-checked	key array of initially checked data items of the right list	array	—	[ ]

Slot

Name	Description
left-footer	content of left list footer
right-footer	content of right list footer

Scoped Slot

Name	Description
—	Custom content for data items. The scope parameter is { option }

Methods

Method	Description	Parameters
clearQuery	clear the filter keyword of a certain panel	'left' / 'right'

Events

Event Name	Description	Parameters
change	triggers when data items change in the right list	key array of current data items in the right list, transfer direction (left or right), moved item keys
left-check-change	triggers when end user changes the checked state of any data item in the left list	key array of currently checked items, key array of items whose checked state have changed
right-check-change	triggers when end user changes the checked state of any data item in the right list	key array of currently checked items, key array of items whose checked state have changed