Transfer

Uso básico

:::demo Los datos se pasan a Transfer a través del atributo `data`. Los datos tienen que ser un array de objetos, y cada objeto debe tener estos atributos: `key` que será el identificador del ítem, `label` que será el texto a mostrar, y `disabled` que indicará si el elemento esta desactivado. Los items dentro de la lista destino están sincronizados con la variable asociada a `v-model`, y el valor de esa variable es un array de claves de los elementos de la lista destino. Así que si no quiere que la lista destino esté vacía inicialmente puede inicializar el `v-model` con un 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>
```

:::

Filtrar

Puede buscar y filtrar los items.

:::demo Ponga el atributo `filterable` a true para permitir el filtrado. Por defecto si el `label` del ítem contiene el término buscado será incluido en el resultado. También puede implementar su propio método de filtrado con el atributo `filter-method`, que recibe un método y le pasa la búsqueda y cada ítem. Los items para los que devuelva true serán incluidos en el resultado de la búsqueda.

```
<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>
```

:::

Personalizable

Puede personalizar los títulos, botones, la función de renderizado de los items, el texto de estatus de la cabecera y el contenido del pie.

:::demo Use `titles`, `button-texts`, `render-content` y `format` respectivamente para personalizar los títulos de las listas, el texto de los botones, la función de renderizado para los items y el texto de la cabecera que muestra el estado de los items. Además, también puede utilizar el scoped-slot para personalizar los elementos de datos. Para el pie de la lista hay dos slots: `left-footer` y `right-footer`. Además, si quiere algunos items marcados inicialmente puede usar `left-default-checked` y `right-default-checked`. Finalmente este ejemplo muestra el evento `change`. Tenga en cuenta que este ejemplo no se puede ejecutar en jsfiddle porque no

soporta sintaxis JSX. En un proyecto real `render-content` funcionará si las dependencias son configuradas correctamente.

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

Por defecto Transfer busca los atributos `key`, `label`, y `disabled` en cada elemento. Si sus datos tienen un nombre diferente para la clave puede usar el atributo `props` para añadir alias.

:::demo En este ejemplo los elementos no tienen `key` y `label`, en vez de eso tienen `value` y `desc`. Así que tiene que añadir alias para `key` y `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>
```

:::

Atributos

Atributo	Descripción	Tipo	Valores aceptados	Por defecto
value / v-model	valor enlazado	array	—	—
data	Origen de datos	array[{ key, label, disabled }]	—	[ ]
filterable	Si se puede filtrar	boolean	—	false
filter-placeholder	Placeholder para el input del filtro	string	—	Enter keyword
filter-method	Método de filtrado	function	—	—

target-order	estrategia de órdenes para elementos de la lista destino. Si está configurado en `original`, los elementos mantendrán el mismo orden que la fuente de datos. Si está configurado para `push`, los nuevos elementos añadidos se insertaran al final. Si se ajusta a `unshift`, los nuevos elementos añadidos se insertarán en la parte superior.	string	original / push / unshift	original
titles	Títulos de las listas	array	—	['List 1', 'List 2']
button-texts	Texto de los botones	array	—	[ ]
render-content	Función de renderizado	function(h, option)	—	—
format	Texto para el status en el header	object{noChecked, hasChecked}	—	{ noChecked: '${checked}/${total}', hasChecked: '${checked}/${total}' }
props	prop alias para el origen de datos	object{key, label, disabled}	—	—
left-default-checked	Array de claves de los elementos marcados inicialmente en la lista de la izquierda	array	—	[ ]
right-default-checked	Array de claves de los elementos marcados inicialmente en la lista de la derecha	array	—	[ ]

Slot

Nombre	Descripción
left-footer	Contenido del footer de la lista de la izquierda
right-footer	Contenido del footer de la lista de la derecha

Scoped Slot

Name	Descripción
—	Contenido personalizado para los datos de los items. El parámetro del scope es { option }

Métodos

Método	Descripción	Parámetros
clearQuery	borra la palabra clave del filtro de un determinado panel	'left' / 'right'

Eventos

Nombre	Descripción	Parámetros
change	se lanza cuando los elementos cambian en la lista de la derecha	array con las claves de los elementos de la lista de la derecha
left-check-change	se dispara cuando el usuario final cambia el estado verificado de cualquier elemento de datos en la lista izquierda	array clave de ítems actualmente verificados, array clave de ítems cuyo estado verificado ha cambiado
right-check-change	se dispara cuando el usuario final cambia el estado verificado de cualquier elemento de datos en la lista derecha.	array clave de ítems actualmente verificados, array clave de ítems cuyo estado verificado ha cambiado