Transfer

Permet de transférer des options d'une liste à une autre de manière ergonomique.

Usage

:::demo Les données sont passée via l'attribut `data`. Ce doit être un tableau d'objets, chaque objet ayant les propriétés suivantes: `key` étant l'identifiant de l'objet, `label` étant le texte à afficher et `disabled` indiquant si l'objet est désactivé. Ces objets sont synchronisés avec `v-model`, sa valeur étant un tableau d'identifiants des objets. Si vous ne souhaitez pas avoir une liste vide par défaut, vous pouvez donc initialiser `v-model` avec un tableau.

```
<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>
```

:::

Filtrable

Vous pouvez filtrer les options.

:::demo Ajoutez l'attribut `filterable` activer le filtrage. Par défaut, si la propriété `label` de l'objet contient le mot-clé, il sera inclus dans les résultats. Vous pouvez aussi implémenter votre propre filtre grâce à `filter-method`. Cette fonction est lancée à chaque changement de mot-clé. Si elle retourne `true` l'objet en question sera dans les résultats.

```
<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>
```

:::

Personnalisable

Vous pouvez personnaliser les titres, les textes des boutons, les fonctions de rendu pour les objets et le contenu du footer.

:::demo Utilisez `titles`, `button-texts`, `render-content` et `format` pour personnaliser respectivement les titres des listes, les textes des boutons, les fonctions de rendus des objets et le texte des statuts du header. Vous pouvez aussi utiliser des slots pour les objets. Pour le contenu du footer, deux slots sont fournis: `left-footer` et `right-footer`. Si vous souhaitez que certains items soient sélectionnés par défaut, utilisez `left-default-checked` et `right-default-checked`. Enfin, cet exemple utilise aussi l'évènement `change`. Notez que cet

exemple ne fonctionne pas dans jsfiddle car il ne supporte pas JSX. Dans un vrai projet, `render-content` fonctionnera si les dépendances sont satisfaites.

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

:::

Alias des propriétés

Par défaut, Transfer utilise `key`, `label` et `disabled` de vos objets. Si vos objets ont des propriétés différentes, vous pouvez utiliser l'attribut `props` pour définir des alias.

:::demo Les objets de cet exemple n'ont pas de `key` ni `label`, à la place ils ont `value` et `desc`. Vous devez donc configurer les alias de `key` et `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>
```

:::

Attributs

Attribut	Description	Type	Valeurs acceptées	Défaut
value / v-model	La valeur liée.	array	—	—
data	Données principales.	array[{ key, label, disabled }]	—	[ ]
filterable	Si Transfer est filtrable.	boolean	—	false
filter-placeholder	Placeholder du champ de filtrage.	string	—	Enter keyword
filter-method	Méthode de filtrage.	function	—	—
target-order	Ordre de tri des éléments de la liste d'arrivée. S'il est à	string	original / push /	original

	`original`, les éléments garderont le même ordre que la liste d'origine. Si à `push`, les nouveaux éléments seront mis à la suite des anciens. Si mis à `unshift`, les nouveaux éléments seront mis en haut de la liste.		unshift	
titles	Titres des listes.	array	—	['List 1', 'List 2']
button-texts	Textes des boutons.	array	—	[ ]
render-content	Fonction de rendu pour les objets.	function(h, option)	—	—
format	Textes de statut dans le header.	object{noChecked, hasChecked}	—	{ noChecked: '${checked}/${total}', hasChecked: '${checked}/${total}' }
props	Alias des propriétés des objets source.	object{key, label, disabled}	—	—
left-default-checked	Tableau des objets initialement sélectionnés dans la liste de gauche.	array	—	[ ]
right-default-checked	Tableau des objets initialement sélectionnés dans la liste de droite.	array	—	[ ]

Slot

Nom	Description
left-footer	Contenu du footer de la liste de gauche.
right-footer	Contenu du footer de la liste de droite.

Slot avec portée

Nom	Description
—	Contenu personnalisé pour les objets. Le paramètre est { option }.

Méthodes

Méthode	Description	Paramètres
clearQuery	Efface le filtre d'une liste.	'left' / 'right'

Events

Event nom	Description	Paramètres
change	Se déclenche quand un objet change dans la liste de droite.	Tableau des objets de la liste de droite, direction du transfer (`left` ou `right`), les clés des objets bougés.
left-check-change	Se déclenche quand l'utilisateur change le statut d'un objet dans la liste de gauche.	Liste des objets actuellement sélectionnées, Liste des objets dont le statut a changé.
right-check-change	Se déclenche quand l'utilisateur change le statut d'un objet dans la liste de droite.	Liste des objets actuellement sélectionnées, Liste des objets dont le statut a changé.