Select

When there are plenty of options, use a drop-down menu to display and select desired ones.

Basic usage

:::demo `v-model` is the value of `el-option` that is currently selected.

```
<template>
  <el-select v-model="value" placeholder="Select">
    <el-option
      v-for="item in options"
      :key="item.value"
      :label="item.label"
      :value="item.value">
    </el-option>
  </el-select>
</template>

<script>
  export default {
    data() {
      return {
        options: [{
          value: 'Option1',
          label: 'Option1'
        }, {
          value: 'Option2',
          label: 'Option2'
        }, {
          value: 'Option3',
          label: 'Option3'
        }, {
          value: 'Option4',
          label: 'Option4'
        }, {
          value: 'Option5',
          label: 'Option5'
        }],
        value: ''
      }
    }
  }
</script>
```

:::

Disabled option

:::demo Set the value of `disabled` in `el-option` to `true` to disable this option.

```
<template>
  <el-select v-model="value" placeholder="Select">
    <el-option
      v-for="item in options"
      :key="item.value"
      :label="item.label"
      :value="item.value"
      :disabled="item.disabled">
    </el-option>
  </el-select>
</template>

<script>
  export default {
    data() {
      return {
        options: [{
          value: 'Option1',
          label: 'Option1'
        }, {
          value: 'Option2',
          label: 'Option2',
          disabled: true
        }, {
          value: 'Option3',
          label: 'Option3'
        }, {
          value: 'Option4',
          label: 'Option4'
        }, {
          value: 'Option5',
          label: 'Option5'
        }],
        value: ''
      }
    }
  }
</script>
```

:::

Disabled select

Disable the whole component.

:::demo Set `disabled` of `el-select` to make it disabled.

```
<template>
  <el-select v-model="value" disabled placeholder="Select">
    <el-option
      v-for="item in options"
      :key="item.value"
      :label="item.label"
      :value="item.value">
    </el-option>
  </el-select>
</template>

<script>
  export default {
    data() {
      return {
        options: [{
          value: 'Option1',
          label: 'Option1'
        }, {
          value: 'Option2',
          label: 'Option2'
        }, {
          value: 'Option3',
          label: 'Option3'
        }, {
          value: 'Option4',
          label: 'Option4'
        }, {
          value: 'Option5',
          label: 'Option5'
        }],
        value: ''
      }
    }
  }
</script>
```

:::

Clearable single select

You can clear Select using a clear icon.

:::demo Set `clearable` attribute for `el-select` and a clear icon will appear. Note that `clearable` is only for single select.

```
<template>
  <el-select v-model="value" clearable placeholder="Select">
    <el-option
      v-for="item in options"
      :key="item.value"
      :label="item.label"
      :value="item.value">
    </el-option>
  </el-select>
</template>

<script>
  export default {
    data() {
      return {
        options: [{
          value: 'Option1',
          label: 'Option1'
        }, {
          value: 'Option2',
          label: 'Option2'
        }, {
          value: 'Option3',
          label: 'Option3'
        }, {
          value: 'Option4',
          label: 'Option4'
        }, {
          value: 'Option5',
          label: 'Option5'
        }],
        value: ''
      }
    }
  }
</script>
```

:::

Basic multiple select

Multiple select uses tags to display selected options.

:::demo Set `multiple` attribute for `el-select` to enable multiple mode. In this case, the value of `v-model` will be an array of selected options. By default the selected options will be displayed as Tags. You can collapse them to a text by using `collapse-tags` attribute.

```
<template>
  <el-select v-model="value1" multiple placeholder="Select">
    <el-option
      v-for="item in options"
      :key="item.value"
      :label="item.label"
      :value="item.value">
    </el-option>
  </el-select>

  <el-select
    v-model="value2"
    multiple
    collapse-tags
    style="margin-left: 20px;"
    placeholder="Select">
    <el-option
      v-for="item in options"
      :key="item.value"
      :label="item.label"
      :value="item.value">
    </el-option>
  </el-select>
</template>

<script>
  export default {
    data() {
      return {
        options: [{
          value: 'Option1',
          label: 'Option1'
        }, {
          value: 'Option2',
          label: 'Option2'
        }, {
          value: 'Option3',
          label: 'Option3'
        }, {
          value: 'Option4',
          label: 'Option4'
        }, {
          value: 'Option5',
          label: 'Option5'
        }],
        value1: [],
        value2: []
      }
    }
  }
</script>
```

:::

Custom template

You can customize HTML templates for options.

:::demo Insert customized HTML templates into the slot of `el-option`.

```
<template>
  <el-select v-model="value" placeholder="Select">
    <el-option
      v-for="item in cities"
      :key="item.value"
      :label="item.label"
      :value="item.value">
      <span style="float: left">{{ item.label }}</span>
      <span style="float: right; color: #8492a6; font-size: 13px">{{ item.value }}</span>
    </el-option>
  </el-select>
</template>

<script>
  export default {
    data() {
      return {
        cities: [{
          value: 'Beijing',
          label: 'Beijing'
        }, {
          value: 'Shanghai',
          label: 'Shanghai'
        }, {
          value: 'Nanjing',
          label: 'Nanjing'
        }, {
          value: 'Chengdu',
          label: 'Chengdu'
        }, {
          value: 'Shenzhen',
          label: 'Shenzhen'
        }, {
          value: 'Guangzhou',
          label: 'Guangzhou'
        }],
        value: ''
      }
    }
  }
</script>
```

:::

Grouping

Display options in groups.

:::demo Use `el-option-group` to group the options, and its `label` attribute stands for the name of the group.

```
<template>
  <el-select v-model="value" placeholder="Select">
    <el-option-group
      v-for="group in options"
      :key="group.label"
      :label="group.label">
      <el-option
        v-for="item in group.options"
        :key="item.value"
        :label="item.label"
        :value="item.value">
      </el-option>
    </el-option-group>
  </el-select>
</template>

<script>
  export default {
    data() {
      return {
        options: [{
          label: 'Popular cities',
          options: [{
            value: 'Shanghai',
            label: 'Shanghai'
          }, {
            value: 'Beijing',
            label: 'Beijing'
          }]
        }, {
          label: 'City name',
          options: [{
            value: 'Chengdu',
            label: 'Chengdu'
          }, {
            value: 'Shenzhen',
            label: 'Shenzhen'
          }, {
            value: 'Guangzhou',
            label: 'Guangzhou'
          }, {
            value: 'Dalian',
            label: 'Dalian'
          }]
        }],
        value: ''
      }
    }
  }
</script>
```

:::

Option filtering

You can filter options for your desired ones.

:::demo Adding `filterable` to `el-select` enables filtering. By default, Select will find all the options whose `label` attribute contains the input value. If you prefer other filtering strategies, you can pass the `filter-method`. `filter-method` is a `Function` that gets called when the input value changes, and its parameter is the current input value.

```
<template>
  <el-select v-model="value" filterable placeholder="Select">
    <el-option
      v-for="item in options"
      :key="item.value"
      :label="item.label"
      :value="item.value">
    </el-option>
  </el-select>
```

```
</template>

<script>
  export default {
    data() {
      return {
        options: [{
          value: 'Option1',
          label: 'Option1'
        }, {
          value: 'Option2',
          label: 'Option2'
        }, {
          value: 'Option3',
          label: 'Option3'
        }, {
          value: 'Option4',
          label: 'Option4'
        }, {
          value: 'Option5',
          label: 'Option5'
        }],
        value: ''
      }
    }
  }
</script>
```

:::

Remote Search

Enter keywords and search data from server.

:::demo Set the value of `filterable` and `remote` with `true` to enable remote search, and you should pass the `remote-method`. `remote-method` is a `Function` that gets called when the input value changes, and its parameter is the current input value. Note that if `el-option` is rendered with the `v-for` directive, you should add the `key` attribute for `el-option`. Its value needs to be unique, such as `item.value` in the following example.

```
<template>
  <el-select
    v-model="value"
    multiple
    filterable
    remote
    reserve-keyword
    placeholder="Please enter a keyword"
    :remote-method="remoteMethod"
    :loading="loading">
    <el-option
      v-for="item in options"
      :key="item.value"
      :label="item.label"
      :value="item.value">
    </el-option>
  </el-select>
</template>

<script>
  export default {
    data() {
      return {
        options: [],
        value: [],
        list: [],
        loading: false,
        states: ["Alabama", "Alaska", "Arizona",
        "Arkansas", "California", "Colorado",
        "Connecticut", "Delaware", "Florida",
        "Georgia", "Hawaii", "Idaho", "Illinois",
        "Indiana", "Iowa", "Kansas", "Kentucky",
        "Louisiana", "Maine", "Maryland",
        "Massachusetts", "Michigan", "Minnesota",
        "Mississippi", "Missouri", "Montana",
        "Nebraska", "Nevada", "New Hampshire",
        "New Jersey", "New Mexico", "New York",
        "North Carolina", "North Dakota", "Ohio",
        "Oklahoma", "Oregon", "Pennsylvania",
        "Rhode Island", "South Carolina",
        "South Dakota", "Tennessee", "Texas",
        "Utah", "Vermont", "Virginia",
        "Washington", "West Virginia", "Wisconsin",
        "Wyoming"]
      }
    },
    mounted() {
      this.list = this.states.map(item => {
        return { value: `value:${item}`, label: `label:${item}` };
      });
    },
    methods: {
      remoteMethod(query) {
        if (query !== '') {
          this.loading = true;
          setTimeout(() => {
            this.loading = false;
            this.options = this.list.filter(item => {
              return item.label.toLowerCase()
                .indexOf(query.toLowerCase()) > -1;
            });
          }, 200);
        } else {
          this.options = [];
        }
      }
    }
  }
</script>
```

:::

Create new items

Create and select new items that are not included in select options :::demo By using the `allow-create` attribute, users can create new items by typing in the input box. Note that for `allow-create` to work, `filterable` must be `true`. This example also demonstrates `default-first-option`. When this attribute is set to `true`, you can select the first option in the current option list by hitting enter without having to navigate with mouse or arrow keys.

```
<template>
  <el-select
    v-model="value"
    multiple
    filterable
    allow-create
    default-first-option
    placeholder="Choose tags for your article">
    <el-option
      v-for="item in options"
      :key="item.value"
      :label="item.label"
      :value="item.value">
    </el-option>
  </el-select>
</template>

<script>
```

```
  export default {
    data() {
      return {
        options: [{
          value: 'HTML',
          label: 'HTML'
        }, {
          value: 'CSS',
          label: 'CSS'
        }, {
          value: 'JavaScript',
          label: 'JavaScript'
        }],
        value: []
      }
    }
  }
</script>
```

:::

If the binding value of Select is an object, make sure to assign `value-key` as its unique identity key name.

Select Attributes

Attribute	Description	Type	Accepted Values	Default
value / v-model	binding value	boolean / string / number	—	—
multiple	whether multiple-select is activated	boolean	—	false
disabled	whether Select is disabled	boolean	—	false
value-key	unique identity key name for value, required when value is an object	string	—	value
size	size of Input	string	large/small/mini	—

Attribute	Description	Type	Accepted Values	Default
clearable	whether select can be cleared	boolean	—	false
collapse-tags	whether to collapse tags to a text when multiple selecting	boolean	—	false
multiple-limit	maximum number of options user can select when `multiple` is `true`. No limit when set to 0	number	—	0
name	the name attribute of select input	string	—	—
autocomplete	the autocomplete attribute of select input	string	—	off
auto-complete	@DEPRECATED in next major version	string	—	off
placeholder	placeholder	string	—	Select
filterable	whether Select is filterable	boolean	—	false
allow-create	whether creating new items is allowed. To use this, `filterable` must be true	boolean	—	false
filter-method	custom filter method	function	—	—

Attribute	Description	Type	Accepted Values	Default
remote	whether options are loaded from server	boolean	—	false
remote-method	custom remote search method	function	—	—
loading	whether Select is loading data from server	boolean	—	false
loading-text	displayed text while loading data from server	string	—	Loading
no-match-text	displayed text when no data matches the filtering query, you can also use slot `empty`	string	—	No matching data
no-data-text	displayed text when there is no options, you can also use slot `empty`	string	—	No data
popper-class	custom class name for Select's dropdown	string	—	—

Attribute	Description	Type	Accepted Values	Default
reserve-keyword	when `multiple` and `filter` is true, whether to reserve current keyword after selecting an option	boolean	—	false
default-first-option	select first matching option on enter key. Use with `filterable` or `remote`	boolean	-	false
popper-append-to-body	whether to append the popper menu to body. If the positioning of the popper is wrong, you can try to set this prop to false	boolean	-	true
automatic-dropdown	for non-filterable Select, this prop decides if the option menu pops up when the input is focused	boolean	-	false

Select Events

Event Name	Description	Parameters
change	triggers when the selected value changes	current selected value
visible-change	triggers when the dropdown appears/disappears	true when it appears, and false otherwise
remove-tag	triggers when a tag is removed in multiple mode	removed tag value
clear	triggers when the clear icon is clicked in a clearable Select	—
blur	triggers when Input blurs	(event: Event)
focus	triggers when Input focuses	(event: Event)

Select Slots

Name	Description
—	Option component list
prefix	content as Select prefix
empty	content when there is no options

Option Group Attributes

Attribute	Description	Type	Accepted Values	Default
label	name of the group	string	—	—
disabled	whether to disable all options in this group	boolean	—	false

Option Attributes

Attribute	Description	Type	Accepted Values	Default
value	value of option	string/number/object		—

Attribute	Description	Type	Accepted Values	Default
label	label of option, same as `value` if omitted	string/number	—	—
disabled	whether option is disabled	boolean	—	false

Methods

Method	Description	Parameters
focus	focus the Input component	-
blur	blur the Input component, and hide the dropdown	-