Vue-multiselect

(2.0.0-beta.15)

The most complete selecting solution for Vue.js

Stars 1k | license MIT | downloads 15k/month | circleci failing | dependencies none

- Single / multiple select
- Dropdowns
- **Searchable**
- **Tagging**
- Server-side Rendering support
- **Vuex support by default**
- **Ajax support**
- **Fully configurable**
- +99% test coverage
- No dependencies

View on GitHub

Getting started & examples

Getting started

Installation

via npm

```
npm install vue-multiselect --save
```

via CDN

```html
<script src="https://unpkg.com/vue-multiselect@2.0.0-beta.14"></script>
<style src="https://unpkg.com/vue-multiselect@2.0.0-beta.14/dist/vue-multiselect.min.css"></style>
```

Basic usage

```html
<!-- Vue component -->
<template>
```

```
  <div>
    <multiselect v-model="value" :options="options">
  </div>
</template>

<script>
  import Multiselect from 'vue-multiselect'

  // register globally
  Vue.component(Multiselect)

  export default {
    // OR register locally
    components: { Multiselect },
    data () {
      return {
        value: null,
        options: ['list', 'of', 'options']
      }
    }
  }
</script>

<!-- New step!
     Add Multiselect CSS. Can be added as a static asset or inside a component. -->
<style src="vue-multiselect/dist/vue-multiselect.min.css"></style>

<style>
  your styles
</style>
```

Examples

Single select

The basic single select / dropdown doesn't require much configuration.

The `options` prop must be an `Array`.

Optional configuration flags:

- `:searchable="false"` – disables the seach functionality
- `:close-on-select="false"` – the dropdown stays open after selecting an option
- `:show-labels="false"` – the highlighted option doesn't have a label on it

Code sample

```JavaScript
import Multiselect from 'vue-multiselect'

export default {
  components: {
    Multiselect
  },
  data () {
    return {
      value: '',
      options: ['Select option', 'options', 'selected', 'mulitple', 'label', 'searchable', 'clearOnSelect', 'hideSelected', 'maxHeight', 'allowEmpty', 's
    }
  }
}
```

```
Pug
div
  label.typo__label Single select
  multiselect(
    v-model="value",
    :options="options",
    :searchable="false",
    :close-on-select="false",
    :show-labels="false"
    placeholder="Pick a value"
  )
  pre.language-json
    code.
      {{ value  }}
```

```
Html
<div>
  <label class="typo__label">Single select</label>
  <multiselect v-model="value" :options="options" :searchable="false" :close-on-select="false" :show-labels="false" placeholder="Pick a
  <pre class="language-json"><code>{{ value  }}</code></pre>
</div>
```

Single select (object)

When working with objects, you must provide additional props: `label` and `track-by`.

`track-by` is used to identify the option within the options list thus it's value has to be unique. In this example the `name` property is unique across all options, so it can be used as `track-by` value.

`label` is used to display the option.

Optional configuration flags:

- `:searchable="false"` – disables the seach functionality
- `:allow-empty="false"` – once there is a value it can't be deselected
- `deselect-label="Can't remove this value"` – when highlighted, the already selected option will have the Can't remove this value helper label. Useful for single selects that don't allow empty selection.

Code sample

```
JavaScript
import Multiselect from 'vue-multiselect'

export default {
  components: {
    Multiselect
  },
  data () {
    return {
      value: null,
      options: [
        { name: 'Vue.js', language: 'JavaScript' },
        { name: 'Rails', language: 'Ruby' },
        { name: 'Sinatra', language: 'Ruby' },
        { name: 'Laravel', language: 'PHP' },
        { name: 'Phoenix', language: 'Elixir' }
      ]
    }
  }
}
```

Pug

```
div
  label.typo__label Single select / dropdown
  multiselect(
    v-model="value",
    deselect-label="Can't remove this value",
    track-by="name",
    label="name",
    placeholder="Select one",
    :options="options",
    :searchable="false",
    :allow-empty="true"
  )
  pre.language-json
    code.
      {{ value  }}
```

Html

```
<div>
  <label class="typo__label">Single select / dropdown</label>
  <multiselect v-model="value" deselect-label="Can't remove this value" track-by="name" label="name" placeholder="Select one" :optior
  <pre class="language-json"><code>{{ value  }}</code></pre>
</div>
```

Select with search

By default `searchable` is set to true, thus using search doesn't require any prop.

The internal search engine is based on the `label` prop. In other words – when searching, vue-multiselect only compares the option labels with the current search query. If you want to search inside other object properties look at the **ajax search example**.

`custom-label` accepts a function with the `option` object as the first param. It should return a string which is then used to display a custom label.

Code sample

JavaScript

```
import Multiselect from 'vue-multiselect'

export default {
  components: {
    Multiselect
  },
  data () {
    return {
      value: { name: 'Vue.js', language: 'JavaScript' },
      options: [
        { name: 'Vue.js', language: 'JavaScript' },
        { name: 'Rails', language: 'Ruby' },
        { name: 'Sinatra', language: 'Ruby' },
        { name: 'Laravel', language: 'PHP' },
        { name: 'Phoenix', language: 'Elixir' }
      ]
    }
  },
  methods: {
    nameWithLang ({ name, language }) {
      return `${name} — [${language}]`
    }
  }
}
```

Pug

```
div
  label.typo__label Select with search
  multiselect(
    v-model="value",
    :options="options",
    :custom-label="nameWithLang"
    placeholder="Select one",
    label="name",
    track-by="name"
  )
  pre.language-json
    code.
      {{ value  }}
```

Html

```
<div>
  <label class="typo__label">Select with search</label>
  <multiselect v-model="value" :options="options" :custom-label="nameWithLang" placeholder="Select one" label="name" track-by="nar
  <pre class="language-json"><code>{{ value  }}</code></pre>
</div>
```

Multiple select

To allow multiple selections pass the `:multiple="true"` prop.

Optional configuration flags:

- `:close-on-select="false"` – the dropdown stays open after selecting an option
- `:hide-selected="true"` – already selected options will not be displayed in the dropdown
- `:clear-on-select="false"` – the search query stays the same after selecting an option

Code sample

JavaScript

```
import Multiselect from 'vue-multiselect'

export default {
  components: {
    Multiselect
  },
  data () {
    return {
      value: [
        { name: 'Vue.js', language: 'JavaScript' }
      ],
      options: [
        { name: 'Vue.js', language: 'JavaScript' },
        { name: 'Adonis', language: 'JavaScript' },
        { name: 'Rails', language: 'Ruby' },
        { name: 'Sinatra', language: 'Ruby' },
        { name: 'Laravel', language: 'PHP' },
        { name: 'Phoenix', language: 'Elixir' }
      ]
    }
  }
}
```

```pug
div
  label.typo__label Simple select / dropdown
  multiselect(
    v-model="value",
    :options="options",
    :multiple="true",
    :close-on-select="false",
    :clear-on-select="false",
    :hide-selected="true",
    placeholder="Pick some"
    label="name",
    track-by="name"
  )
  pre.language-json
    code.
      {{ value  }}
```

```html
<div>
  <label class="typo__label">Simple select / dropdown</label>
  <multiselect v-model="value" :options="options" :multiple="true" :close-on-select="false" :clear-on-select="false" :hide-selected="true
  <pre class="language-json"><code>{{ value  }}</code></pre>
</div>
```

Asynchronous select

Vue-Multiselect supports changing the option list on the fly, thus can be also used a type-a-head search box.

To react to the seach query changes, set a handler function on the `@search-change` event. It receives the `searchQuery` as the first param, which can be used to make an asynchronous API call.

It is convenient to set the `:loading` prop to `true`, whenever a request is in progress. Look at the provided `asyncFind` method for an example usage.

Optional configuration flags:

- `:internal-search="false"` – disables the multiselect's internal search engine. If you do that, you have to manually update the available `:options`.
- `id="ajax"` – every event is emitted with this as the second param. Useful for identification which component instance triggered the method (in loops for example)
- `:limit="3"` – limits the visible results to 3.
- `:limit-text="limitText"` – function that receives the current selected options count and should return a string to show when the `:limit` count is exceed
- `:options-limit="300"` – limits the displayed options to `300`. Useful for optimisations purposes.

Code sample

```javascript
import Multiselect from 'vue-multiselect'
import { ajaxFindCountry } from './countriesApi'

export default {
  components: {
    Multiselect
  },
  data () {
    return {
      selectedCountries: [],
      countries: [],
      isLoading: false
    }
  },
  methods: {
    limitText (count) {
      return `and ${count} other countries`
    },
    asyncFind (query) {
      this.isLoading = true
      ajaxFindCountry(query).then(response => {
        this.countries = response
        this.isLoading = false
      })
    }
  }
}
```

Pug

```
div
  label.typo__label Async multiselect
  multiselect(
    v-model="selectedCountries",
    id="ajax",
    label="name",
    track-by="code",
    placeholder="Type to search",
    :options="countries",
    :multiple="true",
    :searchable="true",
    :loading="isLoading",
    :internal-search="false",
    :clear-on-select="false",
    :close-on-select="false",
    :options-limit="300",
    :limit="3",
    :limit-text="limitText",
    @search-change="asyncFind"
  )
    span(slot="noResult").
      Oops! No elements found. Consider changing the search query.
  pre.language-json
    code.
      {{ selectedCountries  }}
```

Html

```
<div>
  <label class="typo__label">Async multiselect</label>
  <multiselect v-model="selectedCountries" id="ajax" label="name" track-by="code" placeholder="Type to search" :options="countries" :
  <pre class="language-json"><code>{{ selectedCountries  }}</code></pre>
</div>
```

Tagging

To add tagging functionality to single/multiple selects, set the `:taggable` prop to `true`. This will add an additional option at the beginning of the options list whenever you type a phrase that doesn't have an exact match in the available options. Selecting this temporary option will emit the `@tag` event with the current typed search query as the first param. The event handler should add the received **tag** to both the options list and the value.

Remember that when working with objects as options, you have to transform the received tag string to an object that matches the objects structure of the options list. In this example, the `addTag` method generates an object with a unique `code` property.

Optional configuration flags:

- `tag-placeholder="Add this as new tag"` – A helper label that will be displayed when highlighting the just typed tag suggestion.

Code sample

JavaScript

```javascript
import Multiselect from 'vue-multiselect'

export default {
  components: {
    Multiselect
  },
  data () {
    return {
      value: [
        { name: 'Javascript', code: 'js' }
      ],
      options: [
        { name: 'Vue.js', code: 'vu' },
        { name: 'Javascript', code: 'js' },
        { name: 'Monterail', code: 'pl' },
        { name: 'Open Source', code: 'os' }
      ]
    }
  },
  methods: {
    addTag (newTag) {
      const tag = {
        name: newTag,
        code: newTag.substring(0, 2) + Math.floor((Math.random() * 10000000))
      }
      this.options.push(tag)
      this.value.push(tag)
    }
  }
}
```

Pug

```pug
div
  label.typo__label Tagging
  multiselect(
    v-model="value",
    tag-placeholder="Add this as new tag",
    placeholder="Search or add a tag",
    label="name",
    track-by="code",
    :options="options",
    :multiple="true",
    :taggable="true",
    @tag="addTag"
  )
  pre.language-json
    code.
      {{ value  }}
```

Html

```html
<div>
  <label class="typo__label">Tagging</label>
  <multiselect v-model="value" tag-placeholder="Add this as new tag" placeholder="Search or add a tag" label="name" track-by="code" :
  <pre class="language-json"><code>{{ value  }}</code></pre>
</div>
```

Custom option template

You can use `option` **scoped slot** to provide a custom option template. The available `props` include `props.option` and `props.search`. Look at the provided example for more details.

To ensure the keyboard navigation works properly, remember to set the `:option-height` so it equals the height of the option template. By default, the component assumes an option height of 40px.

Optional configuration flags:

- `:option-height="104"` – The height of the custom option template.

Code sample

```javascript
import Multiselect from 'vue-multiselect'

export default {
  components: {
    Multiselect
  },
  data () {
    return {
      value: { title: 'Explorer', desc: 'Discovering new species!', img: 'static/posters/creatures.png' },
      options: [
        { title: 'Space Pirate', desc: 'More space battles!', img: 'static/posters/fleet.png' },
        { title: 'Merchant', desc: 'PROFIT!', img: 'static/posters/trading_post.png' },
        { title: 'Explorer', desc: 'Discovering new species!', img: 'static/posters/creatures.png' },
        { title: 'Miner', desc: 'We need to go deeper!', img: 'static/posters/resource_lab.png' }
      ]
    }
  },
  methods: {
    customLabel ({ title, desc }) {
      return `${title} – ${desc}`
    }
  }
}
```

```pug
div
  label.typo__label Custom option template
  multiselect(
    v-model="value",
    placeholder="Fav No Man’s Sky path",
    label="title",
    track-by="title",
    :options="options",
    :option-height="104",
    :custom-label="customLabel",
    :show-labels="false"
  )
    template(slot="option", scope="props")
      img.option__image(:src="props.option.img", alt="No Man’s Sky")
      .option__desc
        span.option__title {{ props.option.title }}
        span.option__small {{ props.option.desc }}
  pre.language-json
    code.
      {{ value  }}
```

```html
<div>
```

```
  <label class="typo__label">Custom option template</label>
  <multiselect v-model="value" placeholder="Fav No Man’s Sky path" label="title" track-by="title" :options="options" :option-height="104
    <template slot="option" scope="props"><img class="option__image" :src="props.option.img" alt="No Man’s Sky">
      <div class="option__desc"><span class="option__title">{{ props.option.title }}</span><span class="option__small">{{ props.op
    </template>
  </multiselect>
  <pre class="language-json"><code>{{ value  }}</code></pre>
</div>
```

Option groups

The options list can also contain groups. It requires passing 2 additional props: `group-label` and `group-values`. `group-label` is used to locate the group label. `group-values` should point to the group’s option list.

Despite that the available options are grouped, the selected options are stored as a flat array of objects.

Please look at the provided example for a example options list structure.

Code sample

JavaScript

```javascript
import Multiselect from 'vue-multiselect'

export default {
  components: {
    Multiselect
  },
  data () {
    return {
      options: [
        {
          language: 'Javascript',
          libs: [
            { name: 'Vue.js', category: 'Front-end' },
            { name: 'Adonis', category: 'Backend' }
          ]
        },
        {
          language: 'Ruby',
          libs: [
            { name: 'Rails', category: 'Backend' },
            { name: 'Sinatra', category: 'Backend' }
          ]
        },
        {
          language: 'Other',
          libs: [
            { name: 'Laravel', category: 'Backend' },
            { name: 'Phoenix', category: 'Backend' }
          ]
        }
      ],
      value: []
    }
  }
}
```

Pug

```pug
div
  label.typo__label Groups
  multiselect(
    v-model="value",
    :options="options",
    :multiple="true",
    group-values="libs",
    group-label="language",
    placeholder="Type to search",
    track-by="name",
    label="name",
  )
    span(slot="noResult").
      Oops! No elements found. Consider changing the search query.
  pre.language-json
    code.
      {{ value  }}
```

Html

```html
<div>
  <label class="typo__label">Groups</label>
  <multiselect v-model="value" :options="options" :multiple="true" group-values="libs" group-label="language" placeholder="Type to sea
  <pre class="language-json"><code>{{ value  }}</code></pre>
</div>
```

Vuex support

Due to the one-way data-flow enforced by Vuex you should not be using `v-model` for manipulating the currently selected value. Because Vue-Multiselect always uses it's own internal copy of the value it never mutates the `:value` by itself, which means it can can safely used with Vuex or even Redux.

In Vue 2.0 `v-model` is just a syntax sugar for `:value` and `@input`. Because of this we can use the `@input` event to trigger Vuex actions or mutations. Whenever we mutate the `:value` in Vuex, Multiselect's internal value will update.

Code sample

JavaScript

```javascript
import Vue from 'vue'
import Vuex from 'vuex'
import Multiselect from 'vue-multiselect'

const { mapActions, mapState } = Vuex

Vue.use(Vuex)

const store = new Vuex.Store({
  state: {
    value: 'Vuex',
    options: ['Vuex', 'Vue', 'Vuelidate', 'Vue-Multiselect', 'Vue-Router']
  },
  mutations: {
    updateValue (state, value) {
      state.value = value
    }
  },
  actions: {
    updateValueAction ({ commit }, value) {
      commit('updateValue', value)
    }
  }
})

export default {
  store,
  components: {
    Multiselect
  },
  computed: {
    ...mapState(['value', 'options'])
    ...mapState(['value', 'options'])
  },
  methods: {
    ...mapActions(['updateValueAction'])
  }
}
```

```pug
div
  label.typo__label Vuex example.
  multiselect(
    placeholder="Pick action",
    :value="value",
    :options="options",
    :searchable="false",
    @input="updateValueAction",
  )
```

```html
<div>
  <label class="typo__label">Vuex example.</label>
  <multiselect placeholder="Pick action" :value="value" :options="options" :searchable="false" @input="updateValueAction"></multisele
</div>
```

Action dispatcher

The component may also act as dispatcher for different actions/methods. In this case there is no need for the `:value` prop. Instead of `@input` you can listen on the `@select` event. The difference between the two is that `@select` only receives the currently selected value instead of the whole list of selected values (if select is multiple).

Optional configuration flags:

- `:reset-after="true"` – Resets the internal value after each select action inside the component.

Code sample

```javascript
import Multiselect from 'vue-multiselect'

export default {
  components: {
    Multiselect
  },
  data () {
    return {
      actions: ['alert', 'console.log', 'scrollTop']
    }
  },
  methods: {
    dispatchAction (actionName) {
      switch (actionName) {
        case 'alert':
          window.alert('You just dispatched "alert" action!')
          break
        case 'console.log':
          console.log('You just dispatched "console.log" action!')
          break
        case 'scrollTop':
          window.scrollTo(0, 0)
          break
      }
    }
  }
}
```

```pug
div
 label.typo__label Open console to see logs.
 multiselect(
   placeholder="Pick action",
   :options="actions",
   :searchable="false",
   :reset-after="true",
   @select="dispatchAction"
 )
```

```html
<div>
  <label class="typo__label">Open console to see logs.</label>
  <multiselect placeholder="Pick action" :options="actions" :searchable="false" :reset-after="true" @select="dispatchAction"></multise
</div>
```

Custom configuration

Shows error when touched, but nothing is selected.

Optional configuration flags:

- `:max-height="150"` – Set the dropdown height to 150px
- `:max="3"` – Set the maximal number of selections
- `:allow-empty="false"` – Doesn't allow to remove the last option if it exists
- `:block-keys="['Tab', 'Enter']"` – Block the `Tab` and `Enter` keys from triggering their default behaviour
- `@close="onTouch"` – Event emitted when closing the dropdown

Code sample

```javascript
import Multiselect from 'vue-multiselect'

export default {
  components: {
    Multiselect
  },
  data () {
    return {
      isDisabled: false,
      isTouched: false,
      value: [],
      options: ['Select option', 'Disable me!', 'Reset me!', 'mulitple', 'label', 'searchable']
    }
  },
  computed: {
    isInvalid () {
      return this.isTouched && this.value.length === 0
    }
  },
  methods: {
    onChange (value) {
      this.value = value
      if (value.indexOf('Reset me!') !== -1) this.value = []
    },
    onSelect (option) {
      if (option === 'Disable me!') this.isDisabled = true
    },
    onTouch () {
      this.isTouched = true
    }
  }
}
```

Pug

```
div(
  :class="{ 'invalid': isInvalid }"
)
  label.typo__label Customized multiselect
  multiselect(
    placeholder="Pick at least one",
    select-label="Enter doesn't work here!",
    :value="value",
    :options="options",
    :multiple="true",
    :searchable="true",
    :allow-empty="false",
    :hide-selected="true",
    :max-height="150",
    :max="3",
    :disabled="isDisabled",
    :block-keys="['Tab', 'Enter']",
    @input="onChange",
    @close="onTouch",
    @select="onSelect"
  )
  label.typo__label.form__label(v-show="isInvalid") Must have at least one value
```

Html

```
<div :class="{ 'invalid': isInvalid }">
  <label class="typo__label">Customized multiselect</label>
  <multiselect placeholder="Pick at least one" select-label="Enter doesn't work here!" :value="value" :options="options" :multiple="true"
  <label class="typo__label form__label" v-show="isInvalid">Must have at least one value</label>
</div>
```

Props

Name	Type	Default	Description
		multiselectMixin.js	
Id	Integer||String		Used to identify the component in events.
Options	Array		Array of available options: Objects, Strings or Integers. If array of objects, visible label will default to option.label.
Value	Object||Array||String||Integer		Presets the selected options.
Multiple	Boolean	`false`	Equivalent to the `multiple` attribute on a <select> input.
TrackBy	String		Used to compare objects. **Only use if options are objects.**

Label	String		Label from option Object, that will be visible in the dropdown.
Searchable	Boolean	`true`	Add / removes search input.
ClearOnSelect	Boolean	`true`	Clear the search input after `select()`. Use only when multiple is true.
HideSelected	Boolean	`false`	Hide already selected options
Placeholder	String	`'Select option'`	Equivalent to the `placeholder` attribute on a <select> input.
AllowEmpty	Boolean	`true`	Allows to remove all selected values. Otherwise one must be left selected.
ResetAfter	Boolean	`false`	Reset `this.value`, `this.search`, `this.selected` after `this.value` changes.
CloseOnSelect	Boolean	`true`	Enable/disable closing after selecting an option
CustomLabel	Function => String		Function used to create a custom label
Taggable	Boolean	`false`	Disable / Enable tagging
TagPlaceholder	String	`'Press enter to create a tag'`	String to show when highlighting a potential tag
Max	Number		Number of allowed selected options.
optionsLimit	Number	1000	Limits the options displayed in the dropdown to the first X options.
groupValues	String		ame of the property containing the group values
groupLabel	String		Name of the property containing the group label
blockKeys	Array	[]	Array of keyboard key aliases to block when selecting
internalSearch	Boolean	true	Decide whether to filter the results internally based on search query. Useful for async filtering, where we search through more complex data.
	Multiselect.vue		
SelectLabel	String	`'Press enter to select'`	String to show when pointing to an option
SelectedLabel	String	`'Selected'`	String to show next to selected option
			String to show when pointing to an alredy

DeselectLabel	String	`'Press enter to remove'`	String to show when pointing to an already selected option
ShowLabels	Boolean	`true`	Decide whether to show labels on highlighted options
Limit	Number	`99999`	Limit the display of selected options. The rest will be hidden within the limitText string.
LimitText	Function => String	``count => `and ${count} more` ``	Function that process the message shown when selected elements pass the defined limit.
Loading	Boolean	`false`	Show/hide the loading spinner.
Disabled	Boolean	`false`	Enable/disable the multiselect.
MaxHeight	Integer	`300`	Sets max-height style value of the dropdown
		pointerMixin.js	
ShowPointer	Boolean	`true`	Enable/disable highlighting of the pointed value.

Events

Name	Attributes	Listen to	Description
Input	`(value, id)`	`@input`	Emitted after `this.value` changes
Select	`(selectedOption, id)`	`@select`	Emitted after selecting an option
Remove	`(removedOption, id)`	`@remove`	Emitted after removing an option
SearchChange	`(searchQuery, id)`	`@search-change`	Emitted after the search query changes
Tag	`(searchQuery, id)`	`@tag`	Emitted after user attemts to add a tag
Open	`(id)`	`@open`	Emitted when the dropdown opens. Useful for detecting when touched.
Close	`(value, id)`	`@close`	Emitted when the dropdown closes

Slots

Name	Description
option	Slot for custom option template. See **example.** **Default: Shows option label**
maxElements	Shows when the maximum options have been selected. Defaults to string: **Default: Maximum of <max> options selected. First remove a selected option to select another.**
	Shows when no elements match the search query. Defaults to string:

noResult	**Default: No elements found. Consider changing the search query.**
beforeList	Shows before the list, when dropdown is open.
afterList	Shows after the list, when dropdown is open.

Created by Damian Dulisz @DamianDulisz

With love from Monterail