

How to read these codes snippets

Vue components contain in the same place `<script>`, `<template>` and `<style>` data contents. In order to distinguish them: *Italic code is for script*. Regular code is for template.

Directives

| | | |
|----------------------------------|------------------------------|--|
| v-if / v-else-if / v-else | Condition | <code><section v-show='var == true'>show me</section></code> |
| v-show | Condition (no alternative) | <code><section v-if='var == true'>show me</section></code> |
| v-for | Loop (iterate dataStore elt) | <code><li v-for='elt in dataStoreElt'>...</code> |
| v-on:event (@event) | Event listener | <code><button v-on:click='function'>Click!</button></code> |
| v-bind:elt (:elt) | Bind | <code><li v-bind:href='dataStoreElt.prop'>...</code> |
| v-model | Bind forms | <code><input type='text' v-model='dataStoreElt' /></code> |

Share data between parent/child components

Direct data sharing between 2 linked components is possible in 3 ways: **props & slot** for *parent* communicating to *child*, and **\$emit** for *child* communicating to *parent*.

props (Parent > Child)

| Parent / | Child / |
|---|---|
| <code>import Child from './url'</code> | <code>props : ['url'] or props : {url : 'url'}</code> |
| <code>components : {Child}</code> | |
| <code><Child url= '/contact' ></code> | <code><a :href= 'url'></code> |

slot (Parent > Child)

| Parent / | Child / |
|---|--|
| <code>import Child from './url'</code> | |
| <code>components : {Child}</code> | |
| <code><Child v-slot:slotName >anything to replace default content </Child></code> | <code><button> fixed data</code> |
| | <code><slot name='slotName'>default free content </slot>{{[n]]} </button></code> |

Slots are **free areas** that can be named to use more than one. Default content of slots set in *child component* can be replaced as needed.

\$emit (Child > Parent)

| Parent / | Child / |
|---|---|
| <code>import Child from './url'</code> | <code>methods : { eventPerso(){ this.\$emit('event-perso',{msg : 'hello'}) } }</code> |
| <code>components : {Child}</code> | |
| <code>data() { return{ msg : "" } }</code> | |
| <code>methods : { setMsg(event) { this.msg = event.msg } }</code> | |
| <code><Child @event-perso = 'setMsg' /></code> | <code><button @click = 'eventPerso'></code> |



Router

Install: `vue add router` // *App.vue* will be erased!

`route = { path: '/', name: 'Name', component: ImportedName OR () => import('./path/Component') }`

`<router-view />` - Display point

`<router-link to="/path"></router-link>` - Link using routes (no reload page)

Lifecycle

`beforeCreate()/created() beforeMount()/mounted() beforeDestroy()/destroyed()`

Function used in `export default {...}` to define the moment some operations must be initiated.

Vuex - General dataStore

State (dataStore)

Paths: `$store.state`

pluggin: `{ mapState }` - use spread operator ...

Pluggin must be used in **computed**

Vuex store/ `state: {key: 'value'}`

* **objects** can be used in place of **strings** to rename variables

Component/ `Import { mapState } from 'vuex'`

`computed: { ...mapState(['var1', 'var2']) }` *

Getters (computed)

Path: `$store.getters`

pluggin: `{ mapGetters }` - use spread operator ...

Pluggin must be used in **computed**

Vuex store/ `getters: { setDate(state) { return `${state.eday} :${state.month} ${state.year} } }`

Component/

direct/ `computed: { date() {this.$store.getters('setDate')}} }`

pluggin/ `import { mapGetters } from 'vuex'`

`computed: { ...mapGetters(['setDate']) }`

Mutations (change State)

Path: - no access from components -

Pluggin: - none -

Isolated from components

Vuex store/ `state: { count: 0 }`

CAPITALIZED_NAME

`mutations: { INCREASE_COUNT(state, amount) {state.count += Number(amount)} }`

args: `state, payload`

Actions (commit changes to State)

Path: `$store.dispatch`

Pluggin: `{ mapActions }` - use spread operator ...



Vuex - General dataStore (cont)

Pluggin must be used in **methods**

Apply mutators to State

args: *context, payload*

```
Vuex store/ actions: { updateCount(context, amount) {context.commit('INCREASE_COUNT', amount)} }
```

Component/

```
direct/ methods: { update() {this.$store.dispatch('updateCount')} }
```

```
pluggin/ import { mapActions } from 'vuex'
```

```
methods: { ...mapActions(['updateCount']) }
```

All data contained in this dataStore is accessible and changeable to any component.

C

By **Peanuts-83**

cheatography.com/peanuts-83/

Published 19th November, 2021.

Last updated 20th November, 2021.

Page 3 of 3.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>