Vue.js cheatsheet

Table of Contents

1 Events

```
<button v-on:click="method">Click me!</button> <button
@click="method">Click me!</button>
```

2 Styling

```
<p v-bind:style="{ property: value }">...</p>
```

3 Conditionals

```
<p v-if="show">This element is shown/hidden in DOM</p> <p v-else-
if="show2">v-else-if is available in Vue 2.1+</p> <p v-else>...</p>
```

```
<p v-show="false">This element is in DOM, but hidden with CSS</p>
```

4 Looping

```
<ul> <li v-for="item in collection">{{ item }}</li> </ul>
```

```
<template v-for="item in collections"> <h2>item.title</h2>
<p>item.text</p> <hr/> </template>
```

```
<ul><li v-for="(value, key) in obj">...</li></ul> <ul><li v-for="
(value, key, index) in obj">...</li></ul> <ul><li v-for="i in 10">{{ i
}}</li></ul> => 123456789
```

5 Components

```
<script> /* global registration of a component */
Vue.component("hello", { template: '<h1>Hello</h1>' /* data must be a
function returning a fresh object! */ data: function () { return { ...
}; } }); </script> <div id="app"> <hello></hello> <hello></hello>
</div>
```

```
/* component description */ var comp = { template: ..., data: function
() { return { ... }; }, ... }; new Vue({ el: '#app', /* Local
registration of a component */ components: { 'my-comp': comp } });
```

6 Vue lifecycle

Vue ←watch→ **Virtual DOM** ←update→ **DOM**

- `new Vue();`
- `.beforeCreate();`
- `.created();`
- `.beforeMount();`
- `.updated();`
- `.beforeUpdate();`
- `.beforeDestroy();`
- `.destroyed();`

```
new Vue({ el: '#app', beforeCreate: function () { ...; }, created:
function () { ...; } ... })
```

7 Development workflow

Code → Preprocessing → Production Server → Users

8 Vue CLI

```
$ npm install -g vue-cli
```

Different templates: `simple`, `webpack-simple`, `webpack`, `browserify` / `browserify-simple`

```
$ vue init webpack-simple my-project $ npm run dev
```

8.1 Example: `webpack-simple` layout

```
$ find . ./.babelrc ./.gitignore ./dist ./dist/build.js
./dist/build.js.map ./index.html ./package.json ./README.md ./src
./src/App.vue ./src/assets ./src/assets/logo.png ./src/main.js
./webpack.config.js $ cat src/App.vue <template> /* there must be one
root element inside a template! */ <div id="app"> <h1>hello world!</h1>
</div> </template> <script> /* a root "component" */ export default {
data: function () { return { data: ... } } } </script> /* scoped limits
the application of this style */ <style scoped> </style>
```

9 Slots

Passing some content (i.e. DOM trees) from outside into a component?

`Parent.vue`:

```
<template> <div class="container"> <app-quote> <h2>The Quote</h2> <p>A
wonderful quote</p> </app-quote> </div> </template> <script> import
appQuote from './components/appQuote.vue'; export default { components:
{ appQuote } } </script>
```

Child component:

```
<template> <div> <slot></slot> </div> </template> <script> export
default {} </script>
```

9.1 Multiple slots: `<slot name="">`

In `child.vue`:

```
<slot name="title"></slot> <slot name="content"></slot>
```

In `parent.vue`:

```
<h2 slot="title">{{ quoteTitle }}</h2> <p slot="content">A wonderful
quote</p>
```

An unnamed slot is the default slot!

9.2 Optional `<slot>`: slots with defaults

```
<slot name="subtitle">The default subtitle content</slot>
```

10 Dynamic components: `<component>`

```
<script> import appQuote from './components/Quote.vue'; import
appAuthor from './components/Author.vue'; import appNew from
'./components/New.vue'; export default { components: { appQuote,
appAuthor, appNew }, data: { selectedComponent: 'appQoute' } }
</script> <template> <div class="container"> <button
@click="selectedComponent = 'appQuote'">Quote</button> <button
@click="selectedComponent = 'appAuthor'">Author</button> <button
@click="selectedComponent = 'appNew">New</button> <hr> <component
:is="selectedComponent"></component> </div> </template>
```

An old component is destroyed on change! Although:

```
<keep-alive> <component :is="selector"></component> </keep-alive>
```

Hooks into `<keep-alive>` lifecycle:

```
export default { ..., destroyed: function() { ...; }, deactivated:
function() { ...; }, activated: function() { ...; } }
```

11 Forms

The `.lazy` modifier: don't react on every little change, just on enter/blur:

```
<input type="password" v-model.lazy="passwd"></input>
```

Multiple bindings to the same name in `data` make an array in `data` automatically (useful for checkboxes, `<input type="checkbox"></input>`).

```
<input type="radio" v-model="choice"></input> <input type="radio" v-
model="choice"></input> <input type="radio" v-model="choice"></input>
```

– automatically switches the `choice` value.

```
<select id="priority" class="form-control" v-model="selectedPriority">
<!-- default value is 'Medium': --> <option v-for="priority in
priorities" :selected="priority = 'Medium'"> {{ priority }} </option>
</select>
```

12 Using `v-model` with components

Component:

```
export default { props: ['value'], methods: { switched(isOn) {
this.$emit('input', isOn); } } }
```

```
<template> ... <app-component v-model="isOn"></app-component> ...
</template>
```

13 Directives: v-...

```
Some existing directives: <p v-text="'some text'"></p> <p v-
text="'<strong>text</strong>'"></p> A custom directive example: <p v-
hightlight="'green'">A custom directive</p>
```

Hooks:

- `bind(el, binding, vnode)` – Once directive is attached;
- `inserted(el, binding, vnode)` – Inserted in parent node;
- `update(el, binding, vnode, oldVnode)` – once component is updated (without children);
- `componentUpdated(el, binding, vnode, oldVnode)` – (with children);
- `unbind(el, binding, vnode)`

```
Vue.directive('hightlight', { bind(el, binding, vnode) {
el.style.backgroundColor = binding.value; }, });
```

Arguments for a directive:

```
<div v-directive:arg="value"></div> e.g. <div v-
highlight:background="green"></div>
```

```
Vue.directive('hightlight', { bind(el, binding, vnode) { if
(binding.arg == 'background') { ... } }, ... }
```

Modifiers:

```
<div v-highlight.delayed="green"></div>
```

```
Vue.directive('highlight', { bind(el, binding, vnode) { if
(binding.modifiers['delayed']) { ... } }, ... }
```

Registering directives locally:

```
export default { directives: { 'local-highlight': { bind(el, binding,
vnode) { ... }, ... } } }
```

Multiple modifiers values: pass an object

14 Filters

```
export default { data() { return { text: "Hello there", fruits:
["Apple", "Banana", "Cherry"], } }, filters: { toLowercase(value) {
return transformed(value); } }, computed: { /* computed are sometimes
better than filters: */ filteredFruits: function() { return
this.fruits.filter((element) => { return
element.match(this.filterText); }); } } }
```

```
<p>{{ text | toLowercase }}</p>
```

15 Mixins

```
export const fruitMixin = { data() { return { fruits: ["Apple",
"Banana", "Cherry"], filterText: "" }; }, computed: { filteredFruits()
{ return this.fruits.filter((elem) => elem.match(this.filterText)); } }
};
```

```
import { fruitMixin } from './fruitMixin'; export default { mixins:
[fruitMixin] data: function () { /* mixins are merged with this: */
return { .. }; } }
```

16 Transitions

`<transition name="*">` elements. CSS classes:

- `*-enter` when forward animation starts;
- `*-enter-active` when forward animation ends;
- `*-leave` when reverse animation starts;
- `*-leave-active` when reverse animation ends;

```
<template> <div class="container"> <div class="row"> <div class="col-
xs-12 col-sm-8 col-sm-offset-2 col-md-6 col-md-offset-3">
<h1>Animations</h1> <hr> <button class="btn btn-primary" @click="show =
!show">Show Alert</button> <br/> <!-- must contain a single element: --
> <transition name="animation-name"> <div class="alert alert-info" v-
if="show">This is some info</div> </transition> </div> </div> </div>
</template> <script> export default { data() { return { show: false };
} } </script> <style> .animation-name-enter { opacity: 0; } .animation-
name-enter-active { transition: opacity 1s; } .animation-name-leave {
/* opacity: 1; */ } .animation-name-leave-active { transition: opacity
1s; opacity: 0; } </style>
```

```
.slide-enter { /* transform: translateY(20px); */ } .slide-enter-active
{ animation: slide-in 1s ease-out forwards; } .slide-leave { } .slide-
leave-active { animation: slide-out 1s ease-out forwards; } @keyframes
slide-in { from: { transform: translateY(20px); } to: { transform:
translateY(0); } } @keyframes slide-out { from: { transform:
translateY(0); } to: { transform: translateY(20px); } }
```

16.1 `appear` attribute

```
<transition name="animation-name" appear>...</transition>
```

triggers animations on load.

16.2 Manual leave-enter classes:

```
<transition appear <!-- enter-class="" --> enter-active-class="animated
bounce" <!-- leave-class="" : don't specify empty fields --> leave-
```

```
active-class="animated shake" > .. </transition>
```

16.3 Dynamic transition name

```
<transition :name="property">
```

16.4 Multiple elements transitions

```
<transition :name="animation" mode="in-out"> <div v-if="show"
key="info">..</div> <div v-else key="warning">..</div> </transition>
```

16.5 Transition JS hooks

```
<transition <!-- all these methods take `el` --> @before-
enter="beforeEnterMethod" @enter="enterMethod" <!-- function
enterMethod(el, done) { ... ; done(); } --> @after-
ender="afterEnterMethod" @enter-cancelled="enterCancelledMethod"
@before-leave="beforeLeaveMethod" @leave="leaveMethod" @after-
leave="afterLeaveMethod" @leave-cancelled="leaveCancelledMethod" >
<div>..</div> </transition>
```

17 HTTP: vue-resource

17.1 Setup

```
$ npm install vue-resource # or $ bower install vue-resource
```

or

```
<script src="//cdn.jsdelivr.net/vue.resource/1.0.3/vue-
resource.min.js"></script>
```

17.2 Importing vue-resource

```
import VueResource from 'vue-resource'; Vue.use(VueResource); new Vue({
});
```

17.3 Usage: simple example

```
methods: { submit() { this.$http.post('/data.json', this.user)
.then(response => { console.log(response); }, error => {
console.log(error); }); }, fetchData() { this.$http.get('data.json')
.then(response => { return response.json(); }).then(data => {
console.log(data); }); } }
```

```
Vue.use(vueResource); Vue.http.options.root =
'https://my.server.org/end/point'; Vue.http.options.headers = { .. };
/* see more in the official vue-resource documentation */
```

17.4 `Vue.http.interceptors` hooks

```
Vue.http.interceptors.push((request, next) => { console.log(request);
... next(response => { ... }); });
```

17.5 Setting custom resources

```
export default { data() { return { resource: {} }; }, methods: {
submit() { /* corresponds to POST */ this.resource.save({}, this.user);
}, fetchData() { /* corresponds to GET */ this.resource. } }, created()
{ this.resource = this.$resource('data.json'); const customActions = {
saveAlt: {method: 'POST', url: 'alternative.json'} }
this.anotherResource = this.$resource('data.json', {}, customActions);
} };
```

17.6 URL templating

```
export default { data() { return { node: 'data' }; }, created() {
this.res = this.$resouce('{node}.json'); ... } };
```

18 Routing

18.1 Setting up

```
$ npm install --save vue-router
```

18.2 First route

`routes.js`:

```
import User from './components/user/User.vue'; import Home from
'./components/Home.vue'; export const routes = [ { path: '', component:
Home }, { path: '/user', component: User }, ];
```

`main.js`:

```
import VueRouter from 'vue-router'; import App from './App.vue'; import
{ routes } from './routes.js'; Vue.use(VueRouter); const router = new
VueRouter({ routes }); new Vue({ el: '#app', render: h => h(App),
router });
```

`App.vue`:

```
<template> ... <router-view></router-view> </template>
```

18.3 Routing modes (hash vs history)

"Seamless" mode: the server always returns `index.html` on any request

```
... const router = { routes, mode: 'history' // 'hash' by default };
```

18.4 Links and navigation

```
<template> <ul class="nav nav-pills" <li role="presentation"><router-
link to="/">Home</router-link></li> <li role="presentation"><router-
link :to="variable">User</router-link></li> </ul> </template>
```

18.5 Styling active links

```
<template> <ul class="nav nav-pills" /* `exact` only matches the whole
URL */ <router-link to="/" tag="li" active-class="active" exact>
<a>Home</a> </router-link> <router-link :to="variable" tag="li" active-
class="active"> <a>User</a> </router-link> </ul> </template>
```

18.6 Using navigation from Javascript

```
export default { ... methods: { navigateHome() {
this.$router.push('/'); } }, ... }
```

18.7 Dynamic paths

```
export const routes = [ { path: '', component: Home }, { path:
'/user/:id', component: User }, ]; export default { data() { return {
/* how to get :id value: */ id: this.$route.params.id } }, ... }
```

Author: Dmytro S.
Created: 2017-02-16 Thu 00:47
Validate