Vue.js cheatsheet

Vue.js is an open-source Model–view–viewmodel JavaScript framework for building user interfaces and single-page applications.

Expressions

```
<div id="app">
  <p>I have a {{ product }}</p>
  <p>{{ product + 's' }}</p>
  <p>{{ isWorking ? 'YES' : 'NO' }}</p>
  <p>{{ product.getSalePrice() }}</p>
</div>
```

See: Delimiters

Binding

```
<a v-bind:href="url">...</a>
```

Shorthand syntax

True or false will add or remove attribute

```
<button :disabled="isButtonDisabled">...
```

If isActive is truthy, the class 'active' will appear

```
<div :class="{ active: isActive }">...
```

Style color set to value of activeColor

```
<div :style="{ color: activeColor }">
```

See: v-bind

Directives

Element inserted/removed based on truthiness

```
<p v-if="inStock">{{ product }}</p>
```

```
<p v-else-if="onSale">...</p>
<p v-else>...</p>
```

Toggles the display: none CSS property

```
<p v-show="showProductDetails">...</p>
```

Two-way data binding

```
<input v-model="firstName" >
```

`v-model.lazy="..."`	Syncs input after change event
`v-model.number="..."`	Always returns a number
`v-model.trim="..."`	Strips whitespace

See: Directives

Actions/Events

Calls addToCart method on component

```
<button v-on:click="addToCart">...
```

Shorthand syntax

Arguments can be passed

```
<button @click="addToCart(product)">...
}
```

To prevent default behavior (e.g. page reload)

```
<form @submit.prevent="addProduct">...
```

Only trigger once

```
<img @mouseover.once="showImage">...
```

`.stop`	Stop all event propagation
`.self`	Only trigger if event.target is element itself

Keyboard entry example

```
<input @keyup.enter="submit">
```

Call onCopy when control-c is pressed

```
<input @keyup.ctrl.c="onCopy">
```

See: Events

List rendering

The `:key` is always recommended

```
<li v-for="item in items"

  {{ item }}
</li>
```

To access the position in the array

```
<li v-for="(item, index) in items">...
```

To iterate through objects

```
<li v-for="(value, key) in object">...
```

Using `v-for` with a component

```
<cart-product v-for="item in products"
              :product="item"
              :key="item.id">
```

See: List Rendering

Component

Component anatomy

```
Vue.component('my-component', {
  components: {

    ProductComponent,
    ReviewComponent
  },
  props: {

    message: String,
    product: Object,
    email: {
      type: String,
      required: true,
      default: "none"
      validator: function (value) {
        // should return true if value is valid
      }
    }
  },
  data: function() {
    // `data` must be a function
    return {
      firstName: 'Vue',
      lastName: 'Mastery'
    }
  },
  computed: {
    // Return cached values until dependencies change
    fullName: function () {
      return this.firstName + ' ' + this.lastName
    }
  },
  watch: {
    // Called when firstName changes value
    firstName: function (value, oldValue) { ... }
  },
  methods: { ... },
  template: '<span>{{ message }}</span>',
  // Can also use backticks in `template` for multi-line
})
```

See: Components Basics

Lifecycle hooks

`beforeCreate`	After the instance has been initialized #
`created`	After the instance is created #
`beforeMount`	Before the first render #
`mounted`	After the instance has been mounted #
`beforeUpdate`	When data changes, before the DOM is patched #
`updated`	After a data change #
`beforeDestroy`	Before the instance is destroyed #
`destroyed`	After a Vue instance has been destroyed #

Custom events

See: Lifecycle Hooks

Set listener on component, within its parent

```
<button-counter v-on:incrementBy="incWithVal">
```

Inside parent component

```
methods: {
  incWithVal: function (toAdd) { ... }
}
```

Inside button-counter template

```
this.$emit(
    'incrementBy', // Custom event name
    5 // Data sent up to parent
  )
```

Use props to pass data into child components, custom events to pass data to parent elements.

See: Custom Events

Single file components

Single file

```
<template>
  <p>{{ greeting }} World!</p>
</template>

<script>
module.exports = {
  data: function () {
    return {
      greeting: 'Hello'
    }
  }
}
</script>

<style scoped>
p {
  font-size: 2em;
  text-align: center;
}
</style>
```

See: Single File Components

Separation

```
<template>
  <div>This will be pre-compiled</div>
</template>
<script src="./my-component.js"></script>
<style src="./my-component.css"></style>
```

See: What About Separation of Concerns?

Slots

Using a single slot

Component template

```
<div>
  <h2>I'm a title</h2>



</div>
```

Use of component with data for slot

```
<my-component>

</my-component>
```

See: Slots

Multiple slots

Component template

```
<div class="container">
  <header>

  </header>
  <main>

  </main>
  <footer>

  </footer>
</div>
```

Use of component with data for slots

```
<app-layout>



</app-layout>
```

See: Slots

Also see

- Vue CLI (cli.vuejs.org)
- Vue Router (router.vuejs.org)
- Vue DevTools (chrome.google.com)
- Nuxt.js (nuxtjs.org)
- Vue.js v1.0.28 cheatsheet Legacy version