VUE ESSENTIALS CHEAT SHEET



EXPRESSIONS

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

DIRECTIVES

```
{{ product }}
```

Element inserted/removed based on truthiness

```
..</h1>
<h1 v-else>..</h1>
<h1 v-show="ok">Hello!</h1>
```

Uses element's display CSS property

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

Two-way data binding

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

LIST RENDERING

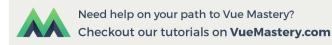
```
...
```

To access the position in the array

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

To iterate through objects

Using v-for with a component



BINDING



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

```
<\!\!\!\text{button :} \textbf{disabled="isButtonDisabled"}\!\!>\!\!\dots
```

True or false will add or remove attribute

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

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

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

Style color set to value of activeColor

ACTIONS/EVENTS



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

Calls addToCart method on component

```
shorthand  <button @click="addToCart">...
```

```
Arguments can be passed
```

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

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

To prevent page reload

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

Only trigger once

Keyboard entry example

```
<input @keyup.ctrl.67="onCopy">
```

Call onCopy when control-c (c is key code 67) is pressed

```
.tab .up .ctrl
.delete .down .alt
.esc .left .shift
.space .right .meta
```

Kev modifiers

.left	.right	.middle	
-------	--------	---------	--

Mouse modifiers

VUE ESSENTIALS CHEAT SHEET



COMPONENT ANATOMY

```
+ -
```

```
Vue.component('my-component', {
  props: { > The parameters the component accepts
    message: String,
    product: Object,
    email: {
      type: String,
      required: true,
      default: "none"
      validator: function (value) {
         Returns true or false
   }
  },
  data: function() { Must be a function
    return {
      firstName: 'Vue',
      lastName: 'Mastery'
  },
  methods: { ... }
  computed: {
                       Return values cached until
    fullName: function () {     dependencies change
    return this.firstName + ' ' + this.lastName
    }
               Called when first Name changes value
  watch: {
   firstName: function (value, oldValue) { .. }
  components: { Components that can be used in the template
    ProductComponent, ReviewComponent
  template: '<span>{{ message }}</span>',
         Can also use backticks for multi-line
```

CUSTOM EVENTS

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

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

Set listener on component, within its parent

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

Inside parent component

Custom event name

this. \$emit('incrementBy', 5) Data sent up to parent

Inside button-counter template



Created by your friends at **VueMastery.com**

LIFECYCLE HOOKS



beforeCreate beforeUpdate created updated beforeMount beforeDestroy mounted destroyed

USING A SINGLE SLOT



Component template

```
<my-component>
This will go in the slot
</my-component>
```

Use of component with data for slot

MULTIPLE SLOTS

```
<div class="container">
    <header>
        <slot name="header"></slot>
        </header>
        <main>
              <slot>Default content</slot>
        </main>
        <footer>
                   <slot name="footer"></slot>
        </footer>
                   <footer>
                   <slot name="footer"></slot>
        </footer>
                   </div>
```

Component template

```
<app-layout>
<h1 slot="header">Page title</h1>
the main content.
Contact info
</app-layout>
```

Use of component with data for slot

NON-PARENT CHILD COMMUNICATION

```
var bus = new Vue()
```

Create global instance

```
bus.$emit('id-selected', 1)
```

Emit event from anywhere

```
bus.$on('id-selected',
    function (id) { ... })
```

Listen for event