PROGRAM WITH ERIK 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) {
        Return 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
    }
  }
  watch: {       Called when firstName changes value
    firstName: function (value, oldValue) { .. }
  },
  components: { components that can be used in the template
    ProductComponenet, 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

PWE

Lifecycle Hooks

beforeCreate	beforeUpdate
created	updated
beforeMount	beforeDestroy
mounted	destroyed

Using a Single Slot

```
<div>
  <h2>I'm a title</h2>
    <slot>
      Only displayed if no content
    </slot>
  </div>
```

Component template

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

Component template

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

PROGRAM WITH ERIK CHEAT SHEET

Expressions

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

Directives

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

Element inserted/removed based on truthiness

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

```
<li v-for="item in items" :key="item.id">
  {{ item }}
</li>
```

key always recommended

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

To access the position in the array

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

To iterate through objects

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

Using v-for with a component

PWE

Binding

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

shorthand

```
<a :href="url">..</a>
```

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

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">...
```

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

Arguments can be passed

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

To prevent page reload

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

Only trigger once

`.stop` Stop all event propagation

`.self` Only trigger if event.target is element itself

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

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

Key modifiers

.left	.right	.middle

Mouse modifiers