Vue

Introduction - Vue.js

Vue.js - The Progressive JavaScript Framework

https://vuejs.org/v2/guide/

Documentation

Expressions

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

Binding

Shorthand syntax	`<a :href="url">...</a>`
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 }">`

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
```

Actions / Events

Calls addToCart method on component

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

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">
```

List rendering

The :key is always recommended

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

Component anatomy

```
Vue.component('my-component', {
components: {
// Components that can be used in the template
ProductComponent,
ReviewComponent
},

props: {
// The parameters the component accepts
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
})
```

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

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.

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>
```

Separation

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

Single slot

Component template

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

Use of component with data for slot

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

Multiple slots

Component template

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

Use of component with data for slot

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