Web Application Development

2020
Vue.js

Part III
Components

Form Input Bindings
v-model
v-model

You can use the v-model directive to create two-way data bindings on form input, textarea, and select elements.

It automatically picks the correct way to update the element based on the input type.
v-model

v-model internally uses different properties and emits different events for different input elements:

- text and textarea elements use value property and input event

<span>Multiline message is:</span>
<p style="white-space: pre-line;">{{ message }}</p>
<textarea v-model="message" placeholder="add multiple lines"></textarea>

Multiline message is:
Line One
Line Two

Line One
Line Two
v-model internally uses different properties and emits different events for different input elements:

- checkboxes and radio buttons use checked property and change event

```html
<input type="checkbox" id="checkbox" v-model="checked">
<label for="checkbox">{{ checked }}</label>
```

v-model will ignore the initial value, checked or selected attributes found on any form elements. It will always treat the Vue instance data as the source of truth. You should declare the initial value on the JavaScript side, inside the data option of your component.
v-model

v-model internally uses different properties and emits different events for different input elements:

- checkboxes and radio buttons use checked property and change event

```javascript
new Vue({
  el: '#example-3',
  data: {
    checkedNames: []
  }
});
```

```html
<div id='example-3'>
  <input type="checkbox" id="jack" value="Jack" v-model="checkedNames">
  <label for="jack">Jack</label>
  
  <input type="checkbox" id="john" value="John" v-model="checkedNames">
  <label for="john">John</label>
  
  <input type="checkbox" id="mike" value="Mike" v-model="checkedNames">
  <label for="mike">Mike</label>
  
  <span>CHECKED NAMES: {{ checkedNames }}</span>
</div>
```
v-model

v-model internally uses different properties and emits different events for different input elements:

- checkboxes and radio buttons use `checked` property and `change` event
v-model

v-model internally uses different properties and emits different events for different input elements:

- select fields use value as a prop and change as an event.

```javascript
new Vue({
  el: '...',
  data: {
    selected: ''
  }
})
```
Filters
Vue.js allows you to define filters that can be used to apply common text formatting.

Filters are usable in two places: mustache interpolations and v-bind expressions.

Filters should be appended to the end of the JavaScript expression, denoted by the “pipe” symbol:

<!-- in mustaches -->
{% message | capitalize %}

<!-- in v-bind -->
<div v-bind:message="message | capitalize"></div>
filters: {
capitalize: function (value) {
  if (!value) return ''
  value = value.toString()
  return value.charAt(0).toUpperCase() + value.slice(1)
}
}
The filter's function always receives the expression's value (the result of the former chain) as its first argument. In the above example, the `capitalize` filter function will receive the value of `message` as its argument.

Filters can be chained:

```javascript
{{ message | filterA | filterB }}
```

Filters are JavaScript functions, therefore they can take arguments:

```javascript
{{ message | filterA('arg1', arg2) }}
```
computed
var vm = new Vue({
  el: '#example',
  data: {
    message: 'Hello'
  },
  computed: {
    // a computed getter
    reversedMessage: function () {
      // 'this' points to the vm instance
      return this.message.split('').reverse().join('')
    }
  }
})

Original message: "Hello"
Reversed message: "olleH"
var vm = new Vue({
  el: '#example',
  data: {
    message: 'Hello'
  },
  methods: {
    reverseMessage: function () {
      return this.message.split('').reverse().join('')
    }
  }
});

Original message: "Hello"
Reversed message: "olleH"
References

https://vuejs.org/v2/api/
Questions?

Next: Backend - Node.js I