Introduction to VUE.js

Stanley Seow

List Rendering

- We can use the v-for directive to render a list of items based on an array. The v-for directive requires a special syntax in the form of **item in items**, where items is the source data array and item is an **alias** for the array element being iterated on

v-for

- Render the element or template block multiple times based on the source data.

```
<div v-for="item in items">
    {{ item.text }}
</div>

<div v-for="(item, index) in items"></div>
```

Conditional Rendering

- Render an element or group of elements based on the condition of a variable
- For group, use <template> for v-if
- v-if
- v-show (CSS display property)

v-show

- Toggles the element’s display CSS property based on the truthy-ness of the expression value.
- This directive triggers transitions when its condition changes.
- There are **NO v-else** for v-show

```
<h2 v-show="ok">It is OK</h2>
```

v-if

- The directive v-if is used to conditionally render a block. The block will only be rendered if the directive’s expression returns a truthy value.

```
<h1 v-if="awesome">Vue is awesome!</h1>
```

v-else-if

- Render this when the first v-if condition false and this condition is true
- Can be chained together

```
<div v-if="type === ‘A’ "> A </div>
<div v-else-if="type === ‘B’ "> B </div>
<div v-else-if="type === ‘C’ "> C </div>
<div v-else> Not A/B/C </div>
```

v-else

- Render this if above v-if conditions is false

Vue Options

- data
- props
- methods
- computed
- watch

data

- Object / Functions
- Must be a function in components

```
data() {
        message: “hello”,
        counter: 0,
}
```

data in components

- In components MUST return an object

```
data() {
        return {
            Number: 123,
            myData: ‘hello’
        }
```

methods

- Methods to be mixed into the Vue instance. You can access these methods directly on the VM instance, or use them in directive expressions.
- All methods will have their **this** context automatically bound to the Vue instance.

computed

- Computed properties are called to return a computed results.
- This is done to reduced putting too much logic in your templates can make them bloated and hard to maintain
- **Don’t accept arguments**

```
<div>
<p>Original message: "{{ message }}"</p>
<p>Computed reversed message: "{{ reversedMessage }}"</p>
</div>

computed: {
        reversedMessage: function () {
                return this.message.split('').reverse().join('')
        }
}
```

watch

- Watch for a changes in a data value and invoke the function
- Watch for a

```
watch: {
        a: function (val, oldVal) {
                console.log( val, oldVal)
 },
```

Methods vs Computed

Methods

- To call a function when an event happen in the DOM
- To call a function from the computed or watchers when something happens in your component.
- You need to pass parameters

Computed

- You need to compose new data from existing data sources
- You need to reference a value directly in your template
- You call the same function more than once in your template

Computed Exercise

- Add computed calculate total
- Add computed calculate discounts based on total sales
- Add computed calculate shipping fees based on total sales

Method Exercise

- + / - buttons called methods of increase / decrease
- +5 / -5 buttons
- Other functions or methods
- Generate orderID