Conditional Rendering and Rendering of Lists

Chad Campbell
INDEPENDENT SOFTWARE CONSULTANT
@chadcampbell https://www.ecofic.com

Overview

Rendering elements conditionally

Rendering lists of items

Reacting to list changes

Rendering Elements Conditionally

Conditional Directives

Used when an app is loaded

Used at runtime

Rendering Conditionally on Load

For slower network connections

For additional JavaScript object initialization

`v-cloak` Directive

Hides elements until a view is compiled

```
<h2 v-cloak>{{ appName }}</h2>
```

v-cloak in Action

Hide an element until a view is compiled

```
[v-cloak] {
  display: none;
}
```

v-cloak Required CSS

Hides DOM elements until a view is compiled

Defined in /public/css/app.css

`v-if … v-else`

`v-show`

Rendering Conditionally During Runtime

`v-if … v-else`

Let you conditionally render content based on an expression

Using `if...else` Directives

```
<label>Query Results</label>
<p v-cloak>
  <div v-if="beers.length === 0">No beers were returned</div>
  <div v-else-if="beers.length === 1">
    1 beer was returned
  </div>
  <div v-else>{{ beers.length }} beers were returned</div>
</p>
```

v-show

Use to conditionally show content at runtime

Using v-show

```
<!-- The search results area -->
<div>
  <div v-show="beers.length === 0">
    No beers were returned
  </div>

  <div v-show="beers.length > 0">
    [Beers will be rendered here]
  </div>
</div>
```

v-show

Sets the CSS display property

Always renders the element

Shows or hides an HTML block

Comparing v-if and v-show

`v-if`	`v-show`
Better for complex logic	Better for basic logic
Conditionally renders	Always renders
Higher toggling costs at runtime	Lower toggling costs at runtime

`v-show`

`v-if`

`v-cloak`

Rendering Elements Conditionally

Rendering Lists of Items

Looping in Vue

v-for Looping

Specific number of times

Through an object

Through a list

Looping a Specific Number of Times

```
<ul>
  <li v-for="page in pageCount">
    <a href="#" v-on:click="search(page);">{{ page }}</a>
  </li>
</ul>
```

Looping a Specific Number of Times

Using v-for to create a pager

```
<ul>
  <li v-for="page in pageCount">
    <a href="#" v-on:click="search(page);">{{ page }}</a>
  </li>
</ul>
```

Alias

Represents the current item that's being iterated on

Value starts at 1, not 0

```
<ul>
  <li v-for="page in pageCount">
    <a href="#" v-on:click="search(page);">{{ page }}</a>
  </li>
</ul>
```

The “in” Delimiter

Helps with readability

```
<ul>
  <li v-for="page of pageCount">
    <a href="#" v-on:click="search(page);">{{ page }}</a>
  </li>
</ul>
```

The “of” Delimiter

A personal preference

✓ More consistent with JavaScript’s iterator syntax

```
<ul>
  <li v-for="page in pageCount">
    <a href="#" v-on:click="search(page);">{{ page }}</a>
  </li>
</ul>
```

Source Data

What's being iterated over

STOP You cannot iterate over a range of floating-point numbers

Traversing Object Properties

```javascript
var growler = new Vue({
  el: '#growler',
  data: {
   currentUser: {
    firstName: 'Chad',
    fingers: 10,
    tags: ['male', 'scorpio']
    socialMedia: {
      twitter: '@ecofic‘,
      youtube:'ecofic'
    }
   }
  }});
```

```html
<div id="growler">
  <ul>
    <li v-for="v in currentUser">
      {{ v }}
    </li>
  </ul>
</div>
```

```
var growler = new Vue({
  el: '#growler',
  data: {
   currentUser: {
    firstName: 'Chad',
    fingers: 10,
    tags: ['male', 'scorpio']
    socialMedia: {
      twitter: '@ecofic‘,
      youtube:'ecofic'
    }
   }
  }});
```

- Chad
- Campbell
- 10
- { "twitter": "@ecofic", "youtube": "ecofic" }
- ["male", "scorpio"]

```
var growler = new Vue({
  el: '#growler',
  data: {
   currentUser: {
    firstName: 'Chad',
    fingers: 10,
    tags: ['male', 'scorpio']
    socialMedia: {
      twitter: '@ecofic‘,
      youtube:'ecofic'
    }
   }
  }});
```

```
<div id="growler">
 <ul>
  <li v-for="(v, k) in currentUser">
   {{ k }} - {{ v }}
  </li>
 </ul>
</div>
```

```
var growler = new Vue({
  el: '#growler',
  data: {
   currentUser: {
    firstName: 'Chad',
    fingers: 10,
    tags: ['male', 'scorpio']
    socialMedia: {
      twitter: '@ecofic‘,
      youtube:'ecofic'
    }
   }
  }});
```

- firstName - Chad
- lastName - Campbell
- fingers - 10
- socialMedia - { "twitter": "@ecofic", "youtube": "ecofic" }
- tags - ["male", "scorpio"]

```
var growler = new Vue({
  el: '#growler',
  data: {
   user: {
    firstName: 'Chad',
    fingers: 10,
    tags: ['male', 'scorpio']
    socialMedia: {
      twitter: '@ecofic‘,
      youtube:'ecofic'
    }
   }
  }});
```

```
<div id="growler">
 <ul>
  <li v-for="(v, k, i) in user">
    {{ i }}: {{ k }} - {{ v }}
  </li>
 </ul>
</div>
```

```
var growler = new Vue({
  el: '#growler',
  data: {
   user: {
    firstName: 'Chad',
    fingers: 10,
    tags: ['male', 'scorpio']
    socialMedia: {
      twitter: '@ecofic‘,
      youtube:'ecofic'
    }
   }
  }});
```

- 0: firstName - Chad
- 1: lastName - Campbell
- 2: fingers - 10
- 3: socialMedia - { "twitter": "@ecofic", "youtube": "ecofic" }
- 4: tags - ["male", "scorpio"]

In the context of properties, the alias value starts at 0, not 1.

Iterating Over an Object

Gets keys via `Object.keys()`

Different JavaScript engines return properties in different orders

Iterating Through Arrays

```
var growler = new Vue({
  el: '#growler',
  data: {
    pages: [1, 2, 3, 4, 5]
  }
});
```

```
<div id="growler">
 <ul>
  <li v-for="page in pages">
   <a href="#">{{ page }}</a>
  </li>
 </ul>
</div>
```

```
var growler = new Vue({
  el: '#growler',
  data: {
    pages: [
     { number: 1, url: '?page=1' },
     { number: 2, url: '?page=2' },
     { number: 3, url: '?page=3' }
    ]
  }
});
```

```
<div id="growler">
  <ul>
    <li v-for="page in pages">
      <a v-bind:href="page.url">
        {{ page.number }}
      </a>
    </li>
  </ul>
</div>
```

```
var growler = new Vue({
  el: '#growler',
  data: {
    pages: [
     { number: 1, url: '?page=1' },
     { number: 2, url: '?page=2' },
     { number: 3, url: '?page=3' }
    ]
  }
});
```

```
<div id="growler">
  <ul>
    <li v-for="(page, i) in pages">
      <a v-bind:href="page.url">
        {{ page.number }}
      </a>
      <small>i: {{ i }}</small>
    </li>
  </ul>
</div>
```

`v-for`

Iterating Through Nested Arrays

```
var growler = new Vue({
  el: '#growler',
  data: {
    pages: [
     { number: 1, url: '?page=1‘,
       sections: [ ‘A’, ‘B’, ‘C’]
     },
     { number: 2, url: '?page=2‘,
       sections: ‘Y’, ‘Z’
     }
    ]
  }
});
```

```
<div id="growler">
  <div v-for="page in pages">
    <b>Page: {{ page.number }}</b>
    <div
      v-for="sec in page.sections">
      {{page.number}}-{{ sec }}
    </div>
    <br />
  </div>
</div>
```

Using `v-for` and `v-if` Together

```
var growler = new Vue({
 el: '#growler',
 data: {
  beers: [
   {name:'Tikibalang Barley Wine',
abv:9.6},
   {name:'Hyote Chocolate Stout',
abv:7.4},
   {name:'Pope Lick Porter',abv:6.5},
   {name:'Ahool Ale',abv:5.4},
   {name:'North Adjule Lager',abv:3.7}
  ],
  maxAbv:7.0
 }
});
```

```
<div id="growler">
  <div>
    <label><strong>Max Alcohol by Volume
(a.b.v.)</strong></label>
    <input type="range" v-model="maxAbv"
      min="0" max="100" />
      <small>{{ maxAbv }}%</small>
  </div>

  <ul>
    <li><strong>Beers</strong></li>
    <li
      v-for="beer in beers"
      v-if="beer.abv < maxAbv">
      {{ beer }}
    </li>
  </ul>
</div>
```

Detecting Array Changes

Mutation Functions

```
sort()
reverse()
push()
pop()
shift()
unshift()
splice()
```

sort Function

Puts elements in an `Array` in order.

Sorting Beer Names

```
beers: [
  'Tikibalang Barley Wine',
  'Hyote Chocolate Stout',
  'Pope Lick Porter',
  'Ahool Ale',
  'North Adjule Lager'
]
```

```
this.beers.sort();
```

```
beers: [
  'Ahool Ale',
  'Hyote Chocolate Stout',
  'North Adjule Lager',
  'Pope Lick Porter',
  'Tikibalang Barley Wine'
]
```

Sorting by Alcohol by Volume Values

```
abv: [
  9.7,
  12.2,
  5.7,
  11.1,
  2.9
]
```

```
this.abv.sort();
```

```
abv: [
  11.1,
  12.2,
  2.9,
  5.7,
  9.7
]
```

```
this.abv.sort(function(v1, v2) {
  return v1 - v2;
});
```

Customizing the sort Function

Sorting by numeric values

`reverse` Function

Rearranges an `Array` into the opposite direction.

Reversing an Array

```
beers: [
  'Ahool Ale',
  'Hyote Chocolate Stout',
  'North Adjule Lager',
  'Pope Lick Porter',
  'Tikibalang Barley Wine'
]
```

```
this.beers.reverse();
```

```
beers: [
  'Tikibalang Barley Wine',
  'Pope Lick Porter',
  'North Adjule Lager',
  'Hyote Chocolate Stout',
  'Ahool Ale'
]
```

Reversing an Array

```
beers: [
  'Hyote Chocolate Stout',
  'Ahool Ale',
  'Tikibalang Barley Wine'
]
```

```
this.beers.reverse();
```

```
beers: [
  'Tikibalang Barley Wine',
  'Ahool Ale',
  'Hyote Chocolate Stout'
]
```

push Function

Adds items to the end of an `Array`

Pushing onto an Array

```
beers: [
  'Hyote Chocolate Stout',
  'Ahool Ale',
  'Tikibalang Barley Wine'
]
```

```
var newBeer =
 'Pope Lick Porter';
this.beers
  .push(newbeer);
```

```
beers: [
  'Tikibalang Barley Wine',
  'Ahool Ale',
  'Hyote Chocolate Stout',
  'Pope Lick Porter'
]
```

pop Function

Removes the last element from an `Array`

Popping an Array Item

```
beers: [
   'Hyote Chocolate Stout',
   'Ahool Ale',
   'Tikibalang Barley Wine',
   'Pope Lick Porter'
]
```

```
this.beers.pop();
```

```
beers: [
   'Hyote Chocolate Stout',
   'Ahool Ale',
   'Tikibalang Barley Wine'
]
```

```
'Pope Lick Porter'
```

`unshift` Function

Adds items to the beginning of an `Array`.

Adding Items to the Beginning of an Array

```
beers: [
  'Hyote Chocolate Stout',
  'Ahool Ale',
  'Tikibalang Barley Wine'
]
```

```
var newBeer = 'Pope Lick Porter';
  this.beers.unshift(newBeer);
```

```
beers: [
  'Pope Lick Porter’,
  'Tikibalang Barley Wine',
  'Ahool Ale',
  'Hyote Chocolate Stout'
]
```

`shift` Function

Removes the first item of an `Array`.

Removing Items from the Start of an Array

```
beers: [
  'Pope Lick Porter',
  'Hyote Chocolate Stout',
  'Ahool Ale',
  'Tikibalang Barley Wine'
]
```

```
this.beers.shift();
```

```
beers: [
  'Tikibalang Barley Wine',
  'Ahool Ale',
  'Hyote Chocolate Stout'
]
```

`splice` Function

Add and remove items to and from an `Array`.

Removing Items from the Middle of an Array

```
beers: [
  'Pope Lick Porter',
  'Hyote Chocolate Stout',
  'Ahool Ale',
  'Tikibalang Barley Wine'
]
```

```
this.beers.splice(1, 2);
```

```
beers: [
  'Pope Lick Porter’,
  'Tikibalang Barley Wine'
]
```

Adding Items to the Middle of an Array

```
beers: [
  'Pope Lick Porter',
  'Tikibalang Barley Wine'
]
```

```
var removed = [
  'Hyote Chocolate Stout',
  'Ahool Ale'
];
for (var i=0; i<removed.length; i++) {
  this.beers.splice(1, 0, removed[i]);
}
```

```
beers: [
  'Pope Lick Porter',
  'Hyote Chocolate Stout',
  'Ahool Ale',
  'Tikibalang Barley Wine'
]
```

Updating an Array Element

```
beers: [
  'Pope Lick Porter',
  'Hyote Chocolate Stout',
  'Ahool Ale',
  'Tikibalang Barley Wine'
]
```

```
Vue.set(this.beers, 2, 'Ahool Pale Ale');
```

```
beers: [
  'Pope Lick Porter',
  'Hyote Chocolate Stout'
```

STOP Vue cannot detect updates to individual Array elements

Updating an Array Element

```
beers: [
  'Pope Lick Porter',
  'Hyote Chocolate Stout',
  'Ahool Ale',
  'Tikibalang Barley Wine'
]
```

```
this.beers.splice(2, 1, 'Ahool Pale Ale');
```

```
beers: [
  'Pope Lick Porter',
  'Hyote Chocolate Stout',
  'Ahool Pale Ale',
  'Tikibalang Barley Wine'
]
```

Summary

Handle array changes

Render lists of items

Conditionally render elements