Web Application Development

Produced by

David Drohan (ddrohan@wit.ie)

Department of Computing & Mathematics
Waterford Institute of Technology

http://www.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Vue.js

PART 6

FILTERS IN VUE

Vue.js

Overall Section Outline

1. **Introduction** – Why you should be using VueJS
2. **Terminology & Overview** – The critical foundation for understanding
3. **Declarative Rendering & Reactivity** – Keeping track of changes (Data Binding)
4. **Components** – Reusable functionality (Templates, Props & Slots)
5. **Routing** – Navigating the view (Router)
6. **Directives**– Extending HTML
7. **Event Handling** – Dealing with User Interaction
8. **Filters** – Changing the way we see things
9. **Computed Properties & Watchers** – Reacting to Data Change
10. **Transitioning Effects** – I like your <style>
11. **Case Study** – Labs in action

Vue.js

Overall Section Outline

1. **Introduction** – Why you should be using VueJS
2. **Terminology & Overview** – The critical foundation for understanding
3. **Declarative Rendering & Reactivity** – Keeping track of changes (Data Binding)
4. **Components** – Reusable functionality (Templates, Props & Slots)
5. **Routing** – Navigating the view (Router)
6. **Directives**– Extending HTML
7. **Event Handling** – Dealing with User Interaction
8. **Filters – Changing the way we see things**
9. **Computed Properties & Watchers** – Reacting to Data Change
10. **Transitioning Effects** – I like your <style>
11. **Case Study – Labs in action**

Filters

CHANGING THE WAY WE SEE THINGS

Vue.js

Introduction - Recap

As previously mentioned, 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**

(the latter supported in 2.1.0+). 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:id="rawId | formatId"></div>
```

Filters in Depth

Similar to AngularJS, Vue.js has its way of transforming data and applying filters to it, but you must keep in mind

that filters don't transform the original data, they only change the output and return a filtered version of it.

Filters can be useful in a lot of different situations like keeping your API responses as clean as possible and handling the formatting of your data on your frontend.

They can also be efficient in cases where you want to avoid repetition and concatenation by encapsulating all that logic behind reusable code blocks.

What follows is an overview on how to create and use your own Filters, and a few other relevant pieces of info.

Default Filters

If this isn't your first time reading about Vue.js filters and you've done a little research, then you know that the older versions shipped with **built-in** filters, but they got removed from Vue 2.0 and this is Evan You 's (the creator of Vue.js)' reasoning behind that:

> Built-in filters can be useful, but they lack the flexibility of pure JavaScript. When a built-in function doesn't suit your needs, you either end up re-implementing something similar (and shipping both in your final code, where the built-in becomes useless, dead code) or have to wait for Vue to update them and release a new version.

With that in mind, **be careful reading or watching old tutorials**.

Defining & Using Filters

Remember, Filters are simple JavaScript functions, they take the value to be transformed as the first parameter, but you can also pass in as many other arguments as you will need to return the formatted version of that value (as we'll see later).

With Vue, you can register your filters in two different ways: **Globally** and **Locally**. The former gives you access to your filter across all your components, while the latter which allows you to contain and only use your filter inside the component it was defined in.

Define a filter globally **before** creating the Vue instance: *(otherwise you'll get a 'Failed to resolve filter' error)*

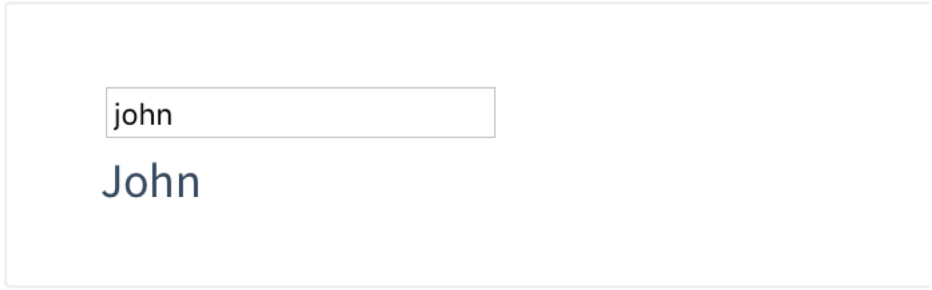
```
Vue.filter('capitalize', function (value) {
  if (!value) return ''
  value = value.toString()
  return value.charAt(0).toUpperCase() + value.slice(1)
})

new Vue({
  // ...
})
```

Defining & Using Filters

Or you can define local filters in a component's options:

```
filters: {
  capitalize: function (value) {
    if (!value) return ''
    value = value.toString()
    return value.charAt(0).toUpperCase() + value.slice(1)
  }
}
```

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



Defining & Using Filters

The filter's function always receives the expression's value (the result of the former chain) as its first argument. In the previous example, the **capitalize** filter function will receive the value of **message** as its argument.

Filters can also be chained, using the pipe (**|**) symbol and you can run a single value through a series of transformers. :

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

In this case, filterA, defined with a single argument, will receive the value of message, and then the filterB function will be called with the result of filterA passed into filterB's single argument.

Vue.js

Defining & Using Filters

As mentioned already, Filters are JavaScript functions, therefore they can take arguments:

```
{{ message | filterA('arg1', arg2) }}
```

Here filterA is defined as a function taking three arguments. The value of message will be passed into the first argument. The plain string 'arg1' will be passed into the filterA as its second argument, and the value of expression arg2 will be evaluated and passed in as the third argument.

Examples

Convert a JavaScript value to a JSON string:

```
Vue.filter('json', function (value) {
    return JSON.stringify(value);
});

new Vue({
    el: '#app',

    data: {
        user: {
            username: 'johndoe',
            email: 'john@example.com',
            countryCode: 'U.K.'
        }
    }
});
```

```
<div id="app">
  <span>{{ user | json }}</span>
</div>
```

Vue.js

Examples

Extracting a list of property values from an array of objects:

```
Vue.filter('pluck', function (objects, key) {
    return objects.map(function(object) {
        return object[key];
    });
});
```

```
new Vue({
    el: '#app',

    data: {
        users: [
        {
            "id": 4,
            "first_name": "Eve",
            "last_name": "Holt"
        },
        {
            "id": 5,
            "first_name": "Charles",
            "last_name": "Morris"
        },
        {
            "id": 6,
            "first_name": "Tracey",
            "last_name": "Ramos"
        }
        ]
    }
});
```

```
<div id="app">
  <span>{{ users | pluck('last_name') }}</span>
</div>
```

Examples

Return the element at the given index:

```
Vue.filter('at', function (value, index) {
    return value[index];
});

new Vue({
    el: '#app',

    data: {
        videos: ['Zi_XLOBDo_Y', 'sOnqjkJTMaA', 'sOnqjkJTMaA']
    }
});
```

```
<div id="app">
  <span>{{ videos | at(1) }}</span>
</div>
```

Examples

Return the minimum value in a given list:

```
Vue.filter('min', function (values) {
    return Math.min(...values);
});

new Vue({
    el: '#app',

    data: {
        ages: [23, 19, 45, 12, 32]
    }
});
```

```
<div id="app">
  <span>{{ ages | min }}</span>
</div>
```

Examples

Shuffle a list of elements:

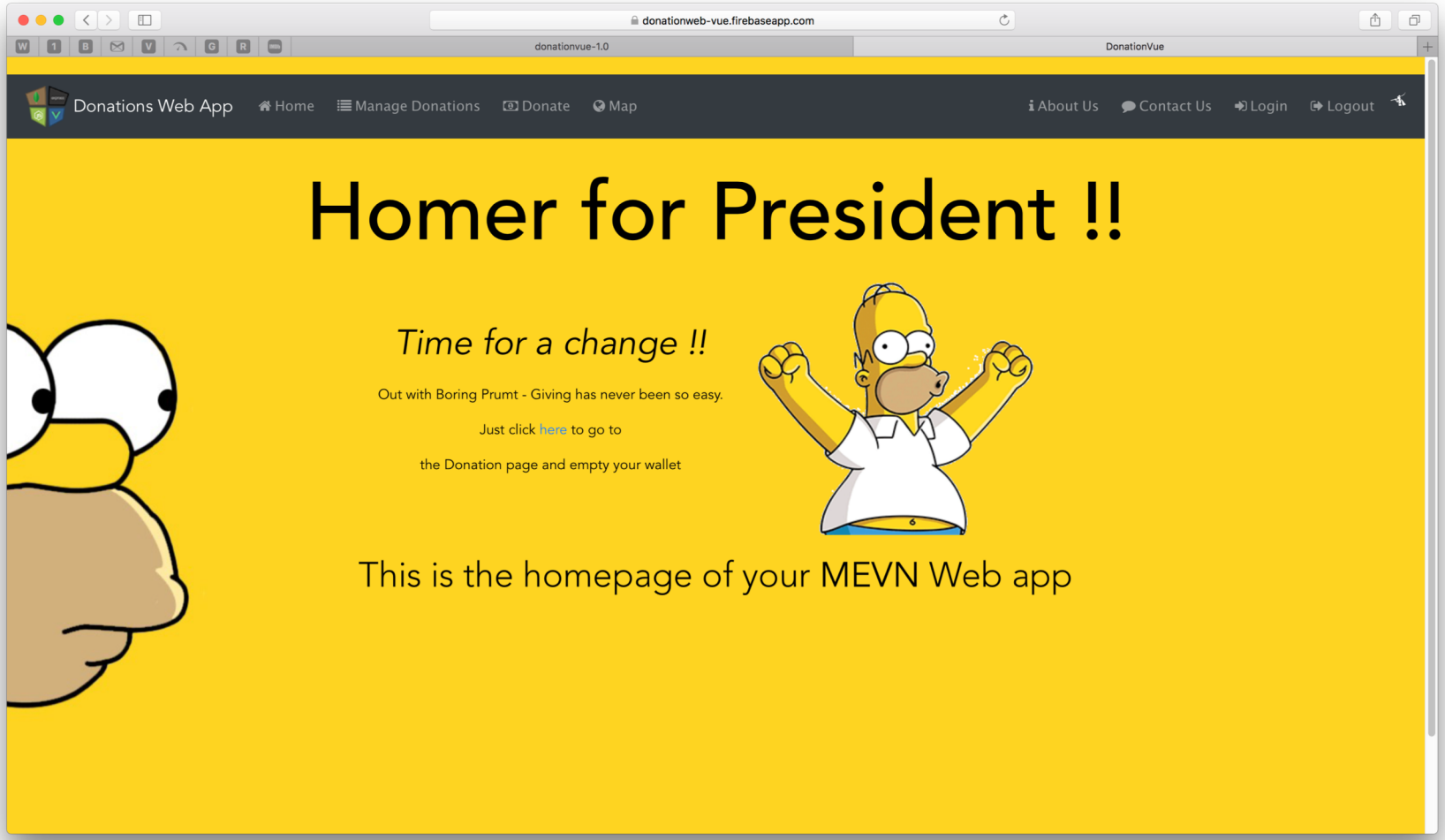
```
Vue.filter('shuffle', function (values) {
    for (var i = values.length - 1; i > 0; i--) {
        var j = Math.floor(Math.random() * (i + 1));
        var temp = values[i];
        values[i] = values[j];
        values[j] = temp;
    }
    return values;
});

new Vue({
    el: '#app',

    data: {
        cards: ['Lahire', 'Judith', 'Lancelot', 'Alexandre']
    }
});
```

```
<div id="app">
  <span>{{ cards | shuffle }}</span>
</div>
```

Case Study

LABS IN ACTION

Demo Application https://donationweb-vue.firebaseapp.com



References

- **https://vuejs.org**
- https://scotch.io/tutorials/how-to-create-filters-in-vuejs-with-examples

Questions?