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 4

DIRECTIVES IN VUE 2.0

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

Overall Section Outline

Directives

EXTENDING HTML

Introduction - Recap

Directives are special attributes with the **v-** prefix. Directive attribute values are expected to be **a single JavaScript expression** (with the exception of **v-for**, which will be discussed later on).

A directive's job is to reactively apply side effects to the DOM when the value of its expression changes. For example:

```
<p v-if="seen">Now you see me</p>
```

Here, the **v-if** directive would remove/insert the **<p>** element based on the truthiness of the value of the expression **seen**.

Introduction - Recap

Some directives can take an “argument”, denoted by a colon after the directive name.

For example, the **v-on** directive is used to reactively listen to DOM events:

```
<a v-on:click="doSomething"> ... </a>
```

Here the argument is the event name to listen to. We will talk about event handling in more detail in the next section.

Directives in Depth

If you have not used **AngularJS** before, you probably don’t know what a directive is. Essentially, a **directive** is some special token in the markup that tells the library to do something to a DOM element. In Vue.js, the concept of directive is drastically simpler than that in Angular. A Vue.js directive can only appear in the form of a prefixed HTML attribute that takes the following format:

```
<element prefix-directiveId="[arg:] ( keypath | expression ) [filters...]"></element>
```

For Example:

```
<div v-text="message"></div>
```

Here the prefix is **v** which is the default. The directive ID is **text** and the the keypath is **message**. This directive instructs Vue.js to update the div’s **textContent** whenever the **message** property on the ViewModel changes.

It’s similar to using **{{message}}** inside the **<div>**

Inline Expressions

```
<div v-text="'hello ' + user.firstName + ' ' + user.lastName"></div>
```

Here we are using a computed expression instead of a single property key. Vue.js automatically tracks the properties an expression depends on and refreshes the directive whenever a dependency changes. Thanks to async batch updates, even when multiple dependencies change, an expression will only be updated once every event loop.

You should use expressions wisely and avoid putting too much logic in your templates, especially statements with side effects (with the exception of event listener expressions). To discourage the overuse of logic inside templates, Vue.js inline expressions are limited to **one statement only**. For bindings that require more complicated operations, use Computed Properties instead.

Arguments

Some directives require an argument before the keypath or expression.

In this example the **click** argument indicates we want the **v-on** directive to listen for a click event and then call the **clickHandler** method of the ViewModel instance.

```
<div v-on="click : clickHandler"></div>
```

or shorthand , we could say

```
<div @click="clickHandler"></div>
```

Vue.js

Filters

Filters can be appended to directive keypaths or expressions to further process the value before updating the DOM. Filters are denoted by a single pipe (**|**) as in shell scripts and can be followed by one or more arguments:

```
<element directive="expression | filterId [args...]"></element>
```

For Example:

```
<span v-text="message | capitalize"></span>
```

```
<span {{ message | uppercase }}></span>
```

```
<span {{ message | uppercase | reverse }}></span>
```

For more on Filters (including writing your own Custom Filters, see

http://optimizely.github.io/vuejs.org/guide/filters.html

Multiple Clauses

You can create multiple bindings of the same directive in a single attribute, separated by commas:

```
<div v-on="
    click   : onClick,
    keyup   : onKeyup,
    keydown : onKeydown
">
</div>
```

Literal Directives

Some directives don’t create data bindings - they simply take the attribute value as a literal string. For example the **v-component** directive:

```
<div v-component="my-component"></div>
```

Here "**my-component**" is not a data property - it’s simply a string ID that Vue.js uses to lookup the corresponding Component constructor.

Since Vue.js 0.10, you can also use mustache expressions inside literal directives. This allows you to *dynamically resolve* the type of component you want to use:

```
<div v-component="{{ isOwner ? 'owner-panel' : 'guest-panel' }}"></div>
```

However, note that mustache expressions inside literal directives are evaluated **only once**. After the directive has been compiled, it will no longer react to value changes. To dynamically instantiate different components at run time, use the **v-view** directive.

A full list of literal directives can be found in the **API reference**.

Empty Directives (speeds up load times)

Some directives don’t even expect an attribute value - they simply do something to the element once and only once. For example the **v-pre** directive:

```
<div v-pre>
    <!-- markup in here will not be compiled -->
</div>
```

Again, a full list of empty directives can be found in the **API reference**.

Custom Directives

You can register a global custom directive with the Vue.directive() method, passing in a **directiveID** followed by a **definition object**. A definition object can provide several hook functions (all optional):

- **bind**: called only once, when the directive is first bound to the element.
- **update**: called when the binding value changes. The new value is provided as the argument.
- **unbind**: called only once, when the directive is unbound from the element.

For Example :

Custom Directives

```
Vue.directive('my-directive', {
    bind: function (value) {
        // do preparation work
        // e.g. add event listeners or expensive stuff
        // that needs to be run only once
        // `value` is the initial value
    },
    update: function (value) {
        // do something based on the updated value
        // this will also be called for the initial value
    },
    unbind: function () {
        // do clean up work
        // e.g. remove event listeners added in bind()
    }
})
```

Custom Directives

Once registered, you can use it in Vue.js templates like this (you need to add the Vue.js prefix to it) :

```
<div v-my-directive="someValue"></div>
```

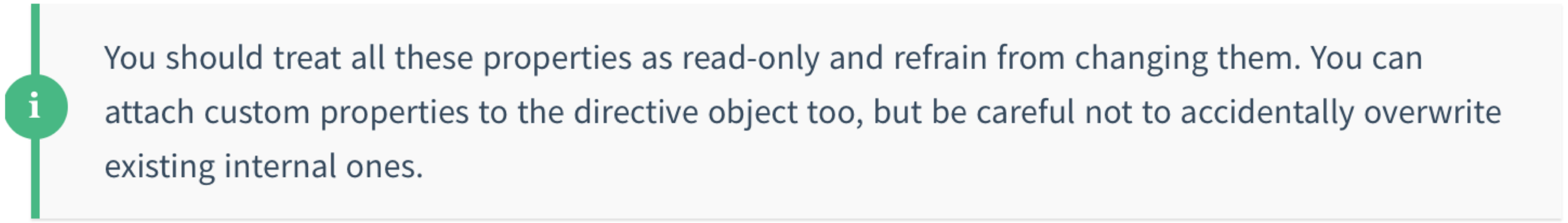
When you only need the update function, you can pass in a single function instead of the definition object:

```
Vue.directive('my-directive', function (value) {
    // this function will be used as update()
})
```

Custom Directives

All the hook functions will be copied into the actual **directive object**, which you can access inside these functions as their this context. The directive object exposes some useful properties:

- **el**: the element the directive is bound to.
- **key**: the keypath of the binding, excluding arguments and filters.
- **arg**: the argument, if present.
- **expression**: the raw, unparsed expression.
- **vm**: the context ViewModel that owns this directive.
- **value**: the current binding value.



You should treat all these properties as read-only and refrain from changing them. You can attach custom properties to the directive object too, but be careful not to accidentally overwrite existing internal ones.

Custom Directives - Example

```html
<div id="demo" v-demo="LightSlateGray : msg"></div>
```

```javascript
Vue.directive('demo', {
    bind: function () {
        this.el.style.color = '#fff'
        this.el.style.backgroundColor = this.arg
    },
    update: function (value) {
        this.el.innerHTML =
            'argument - ' + this.arg + '<br>' +
            'key - ' + this.key + '<br>' +
            'value - ' + value
    }
})
var demo = new Vue({
    el: '#demo',
    data: {
        msg: 'hello!'
    }
})
```

For more on Custom Directives, see **`http://optimizely.github.io/vuejs.org/guide/directives.html`**

Most Commonly Used Directives

Name	Shortcut	Purpose	Example
v-model	none	Creates two-way binding	<textarea rows="5" **v-model**="message" maxlength="72"></textarea>
v-bind	:	Bind attributes dynamically, or pass props	<div **:**style="{ background: color }"></div>
v-if, v-else-if, v-else	none	Conditional Rendering	<p **v-if**="add === 'Y'">Super ☺</p> <p **v-else-if**="add === 'N'">Boo ☹</p> <p **v-else**>Make Up Your Mind!!</p>
v-on	@	Attaches an event listener to the element	<button **@**click="fnName"></button>
v-pre	none	Skip compiling for raw content, can boost performance	<div v-pre> {{ raw content with no methods }} </div>

Vue.js

Most Commonly Used Directives

Name	Shortcut	Purpose	Example
v-show	none	Will show or hide a component/element based on state, but will leave it in the DOM without unmounting (unlike **v-if**)	<child **v-show**="showComponent"></child> (toggles visibility when showComponent is true)
v-once	none	Don't rerender	<div class="**v-once**">Keep me from rerendering</div>
v-for	none	Render a list of items based on an array	<ul id="example-1"> <li **v-for**="item in items"> {{ item.message }} </li></ul>

For Full Reference

https://vuejs.org/v2/api/#Directives

Case Study

LABS IN ACTION

Analysing our Case Study

So now that we’ve covered some more detail about Directives let’s take a closer look at how we us them in **DonationVue**.

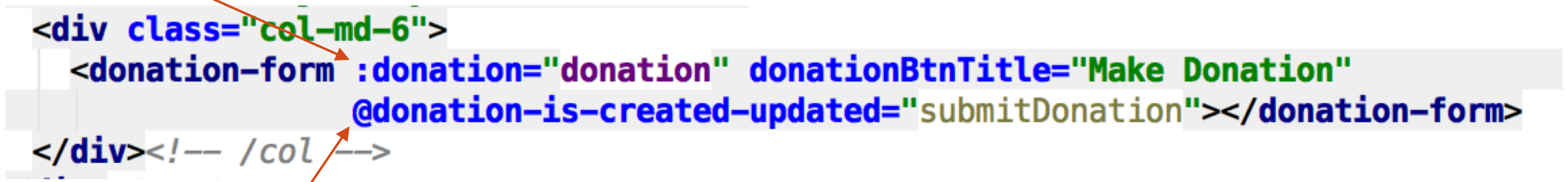
The main files of note are

- **`Donate.vue`**
- **`Donations.vue`**
- **`Edit.vue`**
- **`DonationForm.vue`**

so basically everywhere ☺

Vue.js

Donate.vue

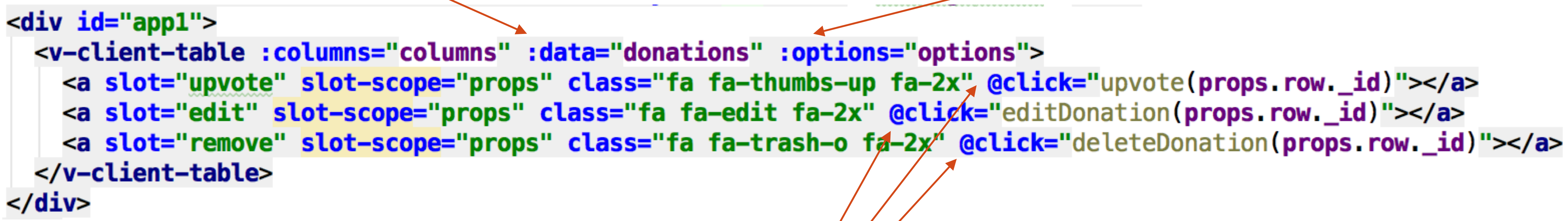


```
<div class="col-md-6">
  <donation-form :donation="donation" donationBtnTitle="Make Donation"
                 @donation-is-created-updated="submitDonation"></donation-form>
</div><!-- /col -->
```

Donations.vue

v-bind shorthand

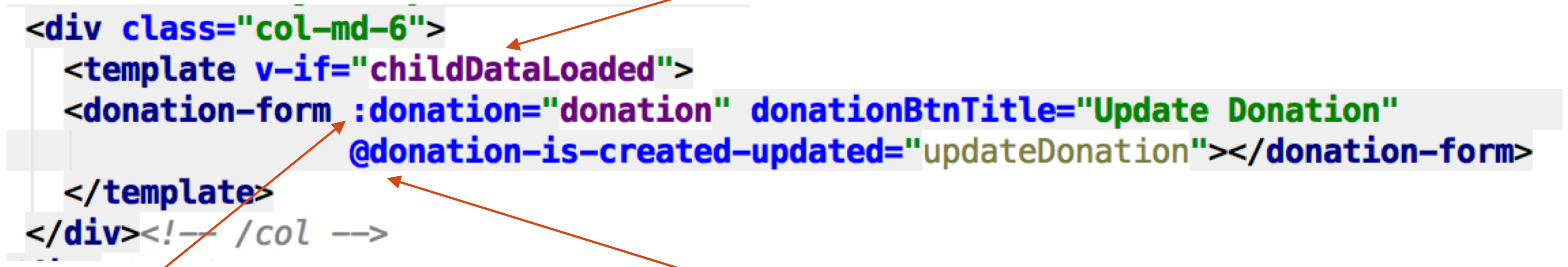
v-bind shorthand



```
<div id="app1">
  <v-client-table :columns="columns" :data="donations" :options="options">
    <a slot="upvote" slot-scope="props" class="fa fa-thumbs-up fa-2x" @click="upvote(props.row._id)"></a>
    <a slot="edit" slot-scope="props" class="fa fa-edit fa-2x" @click="editDonation(props.row._id)"></a>
    <a slot="remove" slot-scope="props" class="fa fa-trash-o fa-2x" @click="deleteDonation(props.row._id)"></a>
  </v-client-table>
</div>
```

v-on shorthand

Edit.vue



```
<div class="col-md-6">
  <template v-if="childDataLoaded">
  <donation-form :donation="donation" donationBtnTitle="Update Donation"
                 @donation-is-created-updated="updateDonation"></donation-form>
  </template>
</div><!-- /col -->
```

v-if

v-on shorthand

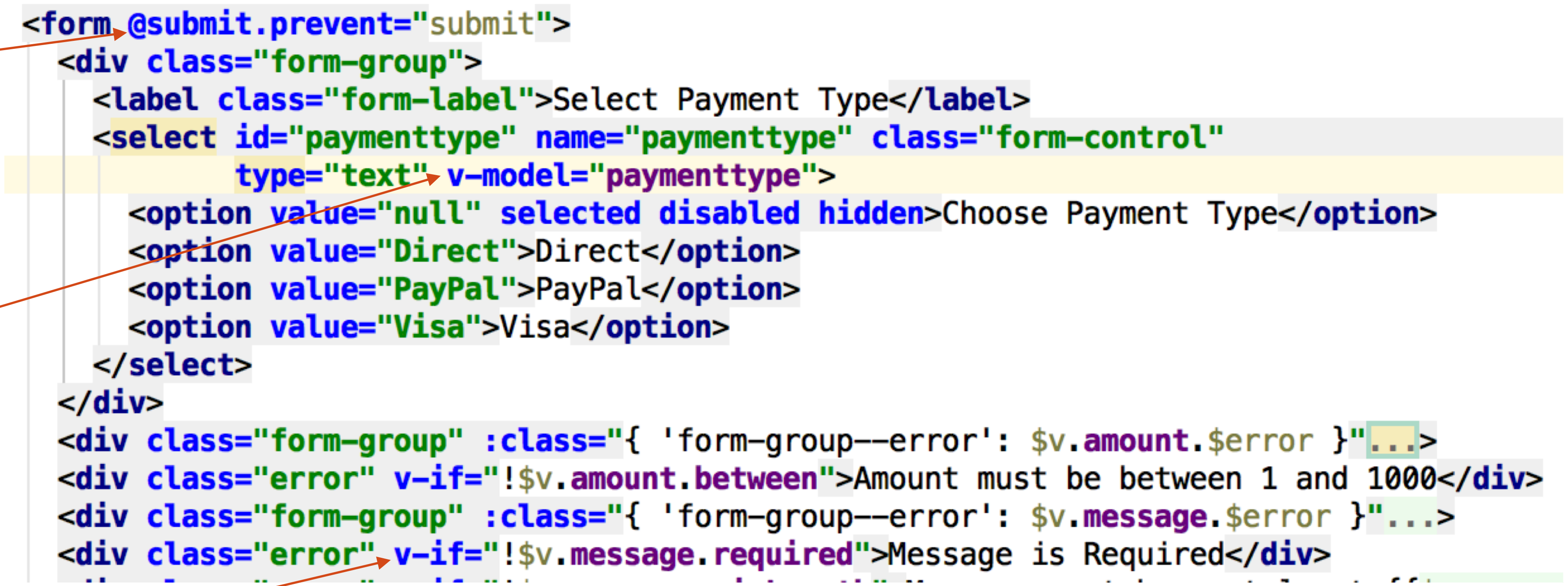
v-bind shorthand

DonationForm.vue

v-on shorthand

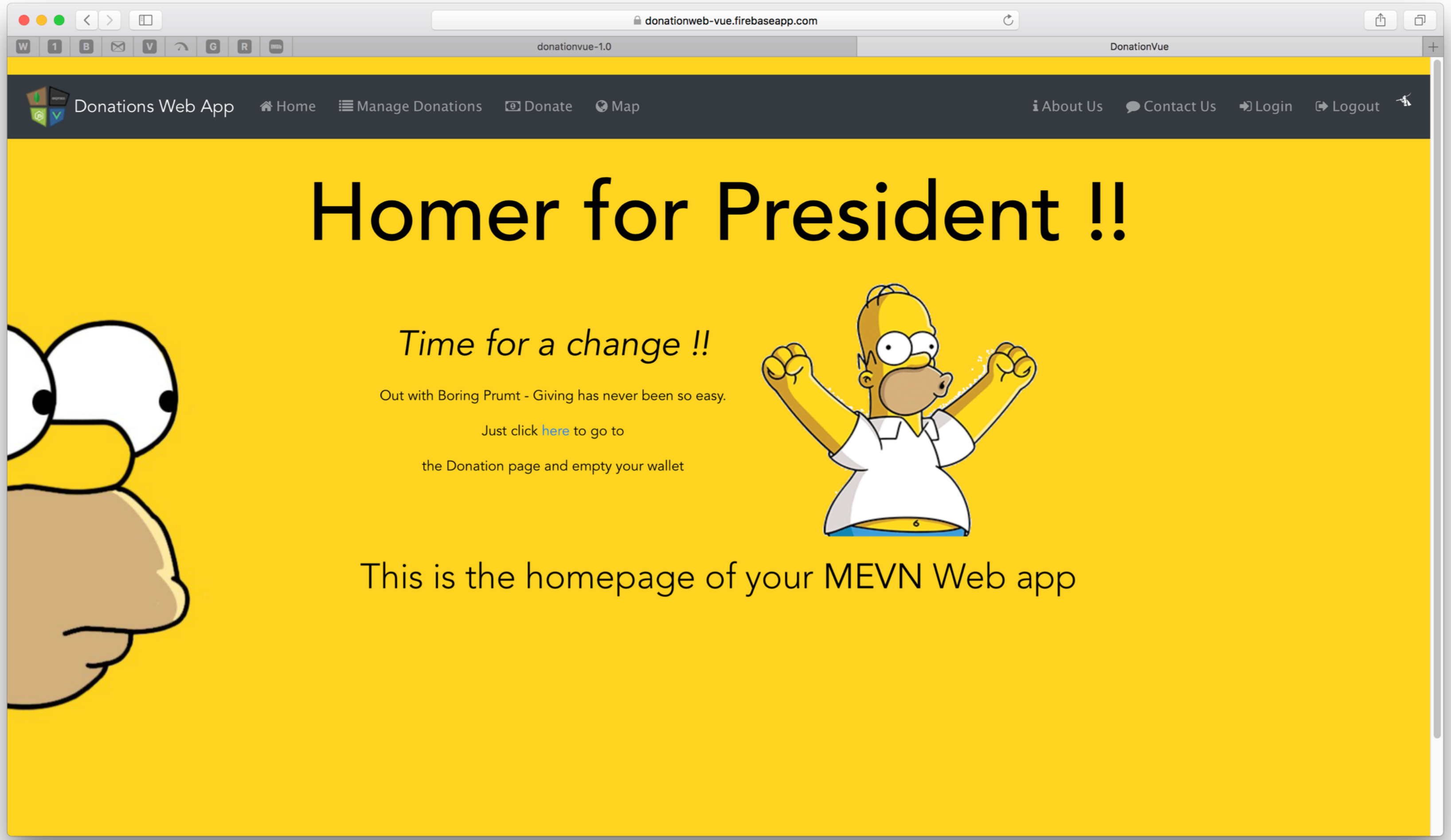
v-model

v-if



```
<form @submit.prevent="submit">
  <div class="form-group">
    <label class="form-label">Select Payment Type</label>
    <select id="paymenttype" name="paymenttype" class="form-control"
            type="text" v-model="paymenttype">
      <option value="null" selected disabled hidden>Choose Payment Type</option>
      <option value="Direct">Direct</option>
      <option value="PayPal">PayPal</option>
      <option value="Visa">Visa</option>
    </select>
  </div>
  <div class="form-group" :class="{ 'form-group--error': $v.amount.$error }"...>
  <div class="error" v-if="!$v.amount.between">Amount must be between 1 and 1000</div>
  <div class="form-group" :class="{ 'form-group--error': $v.message.$error }"...>
  <div class="error" v-if="!$v.message.required">Message is Required</div>
```

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



Homer for President !!

Time for a change !!

Out with Boring Prumt - Giving has never been so easy.

Just click here to go to

the Donation page and empty your wallet

This is the homepage of your MEVN Web app

References

- https://vuejs.org
- https://vuejs.org/v2/api/#Directives
- https://css-tricks.com/intro-to-vue-1-rendering-directives-events/
- https://flaviocopes.com/vue-directives/
- http://optimizely.github.io/vuejs.org/guide/directives.html

Questions?