VUE.JS EVENT HANDLING
imgflip.com

Module 3-12

VUE Event Handling

Objectives

- Use the methods property to define methods
- Implement event handling using the v-on directive
- Utilize component methods inside of event handlers
- Use event modifiers, like propagation and default

VUE Methods

Before tackling handlers we will introduce one more tool to our repertoire, the VUE method.

- A VUE method is similar to a function or method in other languages - they are called when needed, optionally taking in parameters and providing some kind of output.

- Just like with the computed section, the methods section is comprised of JavaScript, thus should be part of the script section in a VUE component.

VUE Methods vs Computed Properties

Methods and Computed properties were designed for different purposes.

- You use a computed property, to generate “derived data” in which your output is based on the data in your JSON data model.
    - Computed values are cached once encountered.

- You use a method when you want a tool that resembles a function in other languages.
    - Methods are executed only when called.

Defining VUE Methods

VUE methods go into their own section, they are a peer of the data and computed section.

```
<script>
export default {
    name: "product-review",
    data() {
            ...
    },
    computed: {
            ...
    },
    methods: {
            //your methods go here
    }
}
</script>
```

Defining VUE Methods

VUE methods are defined in a similar fashion as computed properties, with successive methods split by a comma:

```
methods: {
    numberOfReviews(reviews, starType) {
        return reviews.reduce( (currentCount, review ) => {
            return currentCount + ( review.rating === starType ? 1 : 0);
        }, 0);
    },

    addNewReview() {
        this.reviews.unshift(this.newReview);
        this.resetForm();
    },

    resetForm() {
        this.showForm = false;
        this.newReview = {};
    }
}
```

- Here we have three distinct methods being defined.
- The first method shows that a method can take on parameters and return a value.

Calling VUE Methods

VUE methods work flexibly and can be called in the following contexts:

- Within a v-on directive in the template section (more on this later)
- By a computed property: When we do this, the computed property needs to use this i.e. **this.myMethod();**
- By another method.

Let’s Create Some Methods

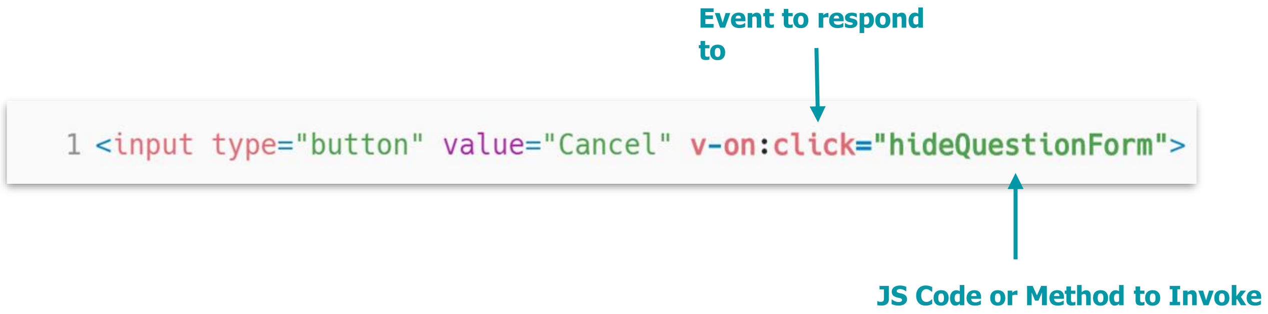
YOU KNOW WHY THE OLD LADY FELL DOWN THE WELL?
SHE DIDN'T SEE THAT WELL.

Event Handling Review

- Recall that a few lectures ago we added event listeners to DOM elements so that certain actions might be taken in response to events that take place on the web page.

- The VUE framework provides a directive to facilitate this.

The **v-on** directive



```
<input type="button" value="Cancel" v-on:click="hideQuestionForm">
```

The **v-on** directive

- The v-on directive takes on the following pattern:

  v-on: **<<event>>**= ‘**<<action to take>>**’

- Here are some examples:

Here we say: when the user clicks on the span, set the JSON data property to counter + 1.

```
<span class="amount" v-on:click="counter += 1">Button has been clicked {{ counter }} times </span>
```

Here we say: when the user click on an anchor, call the method **addNewReivew**

```
<a v-on:click="addNewReview">Add new Review</a>
```

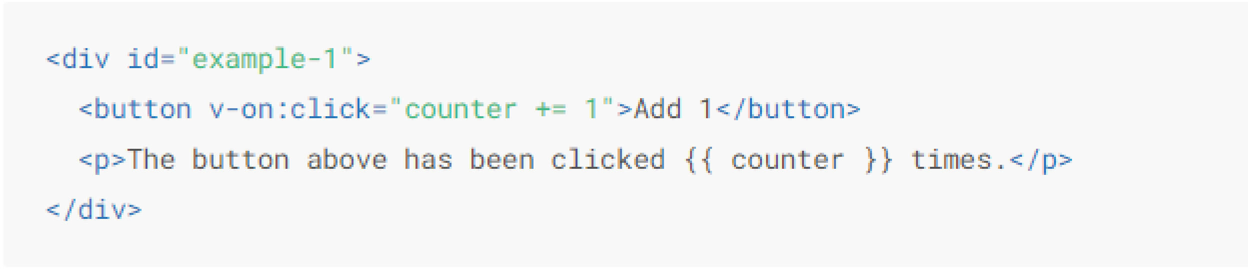
v-on events

- v-on:**click**=”someMethod”
- v-on:**change**=”someMethod”
- v-on:**submit**=”someMethod”
- v-on:**keyup**=”someMethod”
- v-on:**blur**=”someMethod”
- …
- Basically anything we had before, just in Vue.

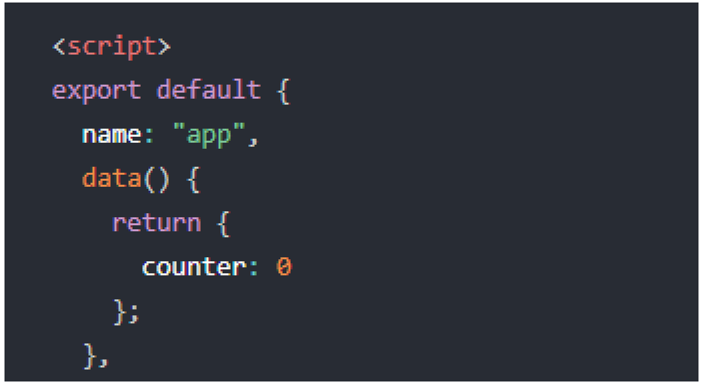
v-on keyboard events

- v-on:keyup.**enter**=”someMethod”
- v-on:keyup.**space**=”someMethod”
- v-on:keyup.**page-down**=”someMethod”
- v-on:keyup.**up**=”someMethod”
- v-on:keyup.**down**=”someMethod”
- v-on:keyup.**left**=”someMethod”
- v-on:keyup.**right**=”someMethod”

v-on directive: inline handler

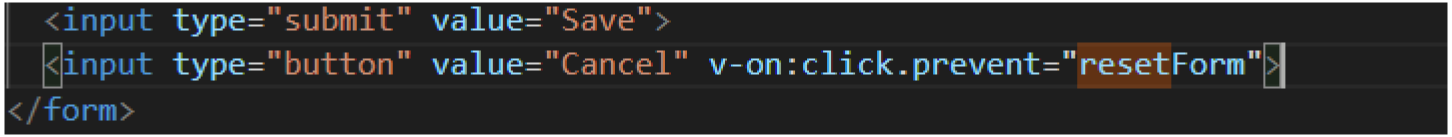


```
<div id="example-1">
  <button v-on:click="counter += 1">Add 1</button>
  <p>The button above has been clicked {{ counter }} times.</p>
</div>
```

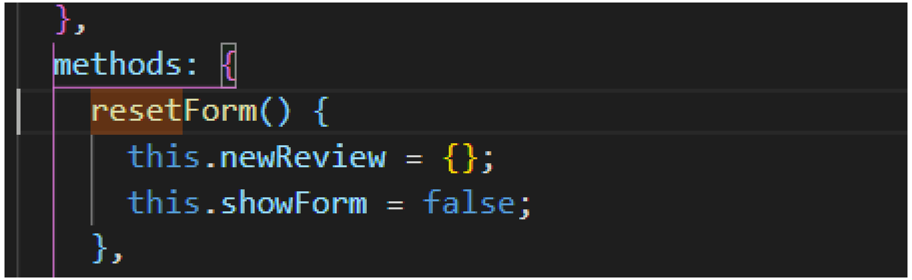


```
<script>
export default {
  name: "app",
  data() {
    return {
      counter: 0
    };
  },
```

v-on directive: method event handler



```
    <input type="submit" value="Save">
    <input type="button" value="Cancel" v-on:click.prevent="resetForm">
</form>
```



```
    },
    methods: {
      resetForm() {
        this.newReview = {};
        this.showForm = false;
      },
```

v-on modifiers

- **v-on:click.stop** - Identical to **event.stopPropagation()**
- **v-on:click.prevent** - Identical to **event.preventDefault()**

v-on modifiers

- **v-on:click.stop** - Identical to **event.stopPropagation()**
- **v-on:click.prevent** - Identical to **event.preventDefault()**

These also exist, but you'll likely never use them:

- **v-on:click.self** - Ignores bubbled up events from children
- **v-on:click.capture** - Uses capturing instead of bubbling
- **v-on:click.once** - Only care about the first occurrence

https://vuejs.org/v2/guide/events.html

Event modifiers

- Just like in Vanilla JS, we may want to prevent default action or stop propagation:

Here we saying: when the user submits the form, call the method **addNewReivew**

```
<form v-if="showForm === true" v-on:submit.prevent="addNewReview">
```

Event modifiers: prevent

- The v-on directive can be modified with a prevent keyword, which prevents the default behavior of a HTML element from executing:

```
<form v-if="showForm === true" v-on:submit.prevent="addNewReview">
```

Note that we are overriding the default behavior of the form submission, and instead choosing to handle the scenario ourselves with our own method.

Event Modifiers: stop

- The v-on directive can be modified with a stop keyword, disabling event bubbling up the DOM.

```
<a v-on:click.stop=“modifyNewReview">
```

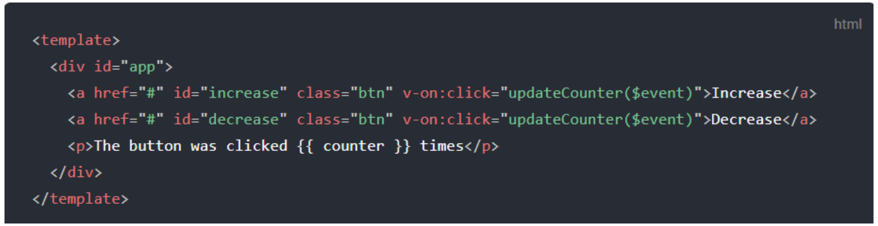
Note that we are stopping the propagation from bubbling up through the DOM.

$event variable

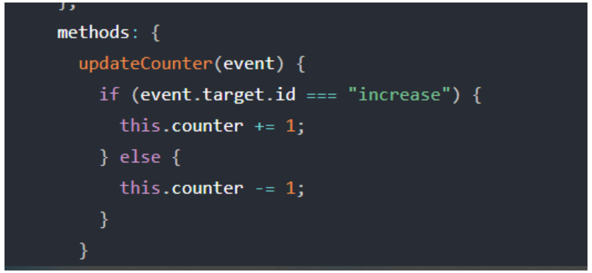
- We may need to pass the original DOM event to a method

```
<button v-on:click="warn('Form cannot be submitted yet.', $event)">
```

```
// …
methods: {
   warn(message, event) {
      if (event) {
         event.preventDefault()
      }
      alert(message)
   }
};
```



```html
<template>
  <div id="app">
    <a href="#" id="increase" class="btn" v-on:click="updateCounter($event)">Increase</a>
    <a href="#" id="decrease" class="btn" v-on:click="updateCounter($event)">Decrease</a>
    <p>The button was clicked {{ counter }} times</p>
  </div>
</template>
```



```
  },
  methods: {
    updateCounter(event) {
      if (event.target.id === "increase") {
        this.counter += 1;
      } else {
        this.counter -= 1;
      }
    }
```

v-if and v-else

```
<!-- will render 'The condition is true' into the DOM -->
<div id="example">
    <h1 v-if="a">The condition is true</h1>
</div>
```

```
var vm = new Vue({
    el: '#example',
    data: {
        a: true,
        b: false
    }
});
```

```
<!-- in this case, nothing will be rendered except for the containing 'div' -->
<div id="example">
    <template v-if="b">
        <h1>Heading</h1>
        <p>Paragraph 1</p>
        <p>Paragraph 2</p>
    </template>
</div>
```

```
<div v-if="'a' === 'b'"> This will never be rendered. </div>
<template v-else>
    <ul>
      <li> You can also use templates with v-else. </li>
      <li> All of the content within the template </li>
      <li> will be rendered. </li>
    </ul>
</template>
```

v-if

The v-if directive will render a DOM element only if certain conditions are met. Consider the following:

```
<template>
   <div class="main">
       <p>Only Bob can see this:</p>
       <p class="description" v-if="name === 'Bob'">Hello {{name}} this
        message will self destruct in 10 seconds.</p>
   </div>
</template>


<script>
export default {
 name: 'product-review',
 data() {
   return {
     name: 'Bob',
     description: 'secret agent'
   }
 }
}
</script>
```

Only Bob can see this:

Hello Bob this message will self destruct in 10 seconds.

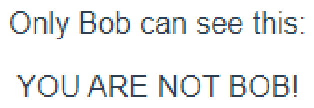
Note that the second paragraph has a v-if directive.

The element will only display if the name attribute is Bob.

v-else

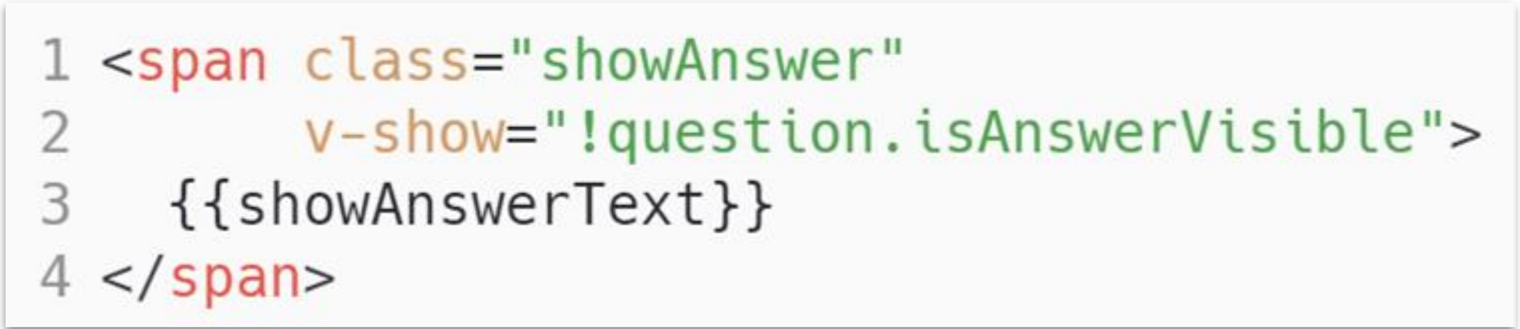
The v-else directive ONLY renders if the v-if is false. Consider the following:

```
<template>
   <div class="main">
       <p>Only Bob can see this:</p>
       <p class="description" v-if="name === 'Bob'">Hello {{name}} this
        message will self destruct in 10 seconds.</p>
       <p class="description" v-else>YOU ARE NOT BOB! <p>
   </div>
</template>

<script>
export default {
 name: 'product-review',
 data() {
   return {
     name: ‘Tim',
     description: 'secret agent'
   }
 }
}
</script>
```



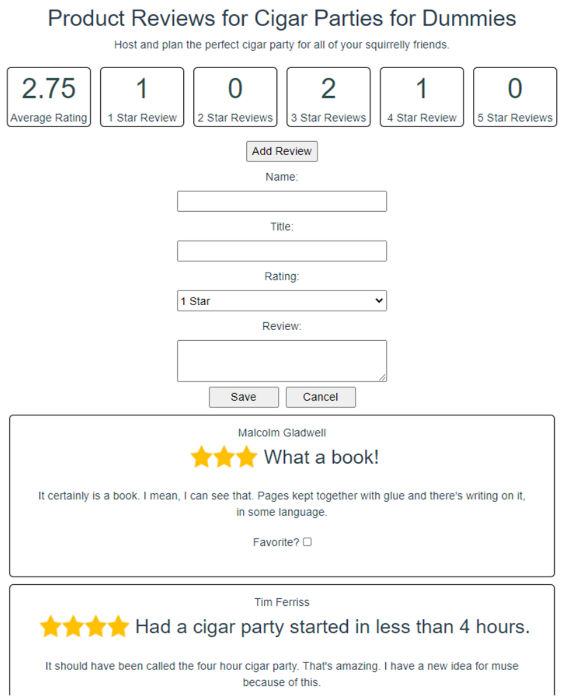
```
Only Bob can see this:

YOU ARE NOT BOB!
```

v-show



```
<span class="showAnswer"
      v-show="!question.isAnswerVisible">
  {{showAnswerText}}
</span>
```

Toggling form visibility



Product Reviews for Cigar Parties for Dummies

Host and plan the perfect cigar party for all of your squirrelly friends.

2.75	1	0	2	1	0
Average Rating	1 Star Review	2 Star Reviews	3 Star Reviews	4 Star Review	5 Star Reviews

Add Review

Name:

Title:

Rating:

1 Star

Review:

Save Cancel

Malcolm Gladwell

What a book!

It certainly is a book. I mean, I can see that. Pages kept together with glue and there's writing on it, in some language.

Favorite?

Tim Ferriss

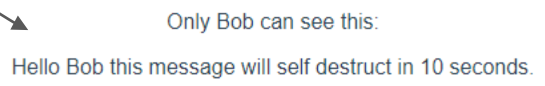
Had a cigar party started in less than 4 hours.

It should have been called the four hour cigar party. That's amazing. I have a new idea for muse because of this.

v-show

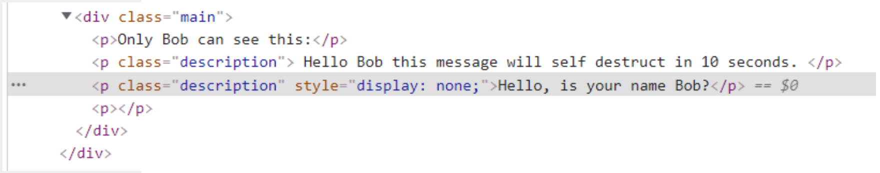
The v-show will hide elements but still have them on the page. Consider the following:

```
<template>
   <div class="main">
       <p>Only Bob can see this:</p>
       <p class="description" v-show=“!name">Hello, is your name Bob?</p>
   </div>
</template>


<script>
export default {
 name: 'product-review',
 data() {
   return {
     name: ‘Bob',
     description: 'secret agent'
   }
 }
}
</script>
```



Only Bob can see this:

Hello Bob this message will self destruct in 10 seconds.

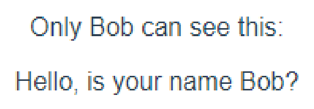


```
▼<div class="main">
   <p>Only Bob can see this:</p>
   <p class="description"> Hello Bob this message will self destruct in 10 seconds. </p>
•••   <p class="description" style="display: none;">Hello, is your name Bob?</p> == $0
   <p></p>
  </div>
</div>
```

v-show

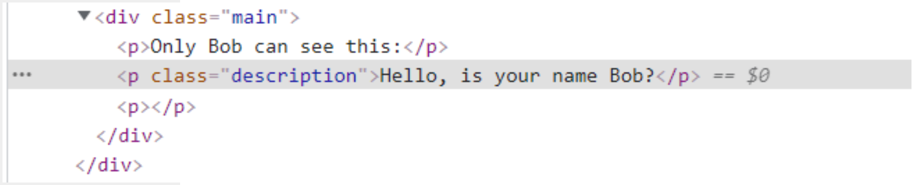
The v-show will hide elements but still have them on the page. Consider the following:

```
<template>
   <div class="main">
       <p>Only Bob can see this:</p>
       <p class="description" v-show=“!name">Hello, is your name Bob?</p>
   </div>
</template>


<script>
export default {
 name: 'product-review',
 data() {
   return {
     name: '',
     description: 'secret agent'
   }
 }
}
</script>
```



Only Bob can see this:

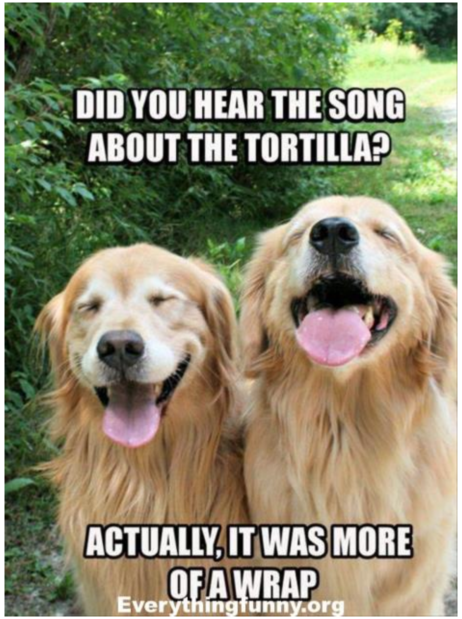
Hello, is your name Bob?



```
▼<div class="main">
   <p>Only Bob can see this:</p>
•••   <p class="description">Hello, is your name Bob?</p> == $0
   <p></p>
  </div>
</div>
```

Let’s Implement Some Event Handlers – but first RECAP!

DID YOU HEAR ABOUT THE SCARECROW WHO WON THE AWARD?
HE WAS OUT STANDING IN HIS FIELD.


DID YOU HEAR THE SONG ABOUT THE TORTILLA?
ACTUALLY, IT WAS MORE OF A WRAP
Everythingfunny.org

Objectives

- Use the methods property to define methods

```
<script>
export default {
    name: "product-review",
    data() {
              ...
    },
    computed: {
              ...
    },
    methods: {
              //your methods go here
    }
}
</script>
```

Sprint LTE 9:37 PM 48%

Dad

Today 11:25 AM

Have you heard of Murphy's Law?

Yep

Have you heard of Cole's Law?

I haven't

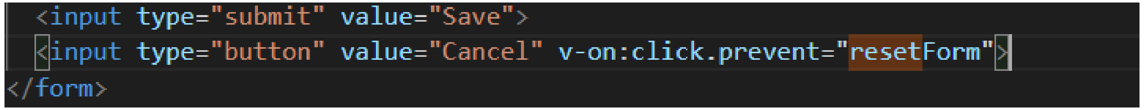
It is thinly sliced cabbage

How dare you

Twitter: @AdamBroud

Objectives

- Use the methods property to define methods
- Implement event handling using the v-on directive



```
    <input type="submit" value="Save">
    <input type="button" value="Cancel" v-on:click.prevent="resetForm">
</form>
```

Objectives

- Use the methods property to define methods
- Implement event handling using the v-on directive
- Utilize component methods inside of event handlers

```
<button v-on:click="warn('Form cannot be submitted yet.',
$event)">
```

```
// …
methods: {
   warn(message, event) {
      if (event) {
         event.preventDefault()
      }
      alert(message)
   }
};
```

Objectives

- Use the methods property to define methods
- Implement event handling using the v-on directive
- Utilize component methods inside of event handlers
- Use event modifiers, like propagation and default

People in Athens hate getting up early
because Dawn is tough on Greece.
fb.com vidadaisypupsofpalm

```
<a v-on:click.stop=“modifyNewReview">
```

```
<form v-if="showForm === true" v-on:submit.prevent="addNewReview">
```