Introduction to Vue.js

Raimund Schnürer

Introduction

- Pronunciation: view
- Most trending JavaScript framework in 2016*
- Main developer: Evan You (yyx990803)
- Project start: 2013

- License: MIT (free for commercial and non-commercial use)
- Documentation: https://vuejs.org/v2/guide/
- API Reference: https://vuejs.org/v2/api/

- Installation: npm install vue or download from https://unpkg.com/vue

* https://risingstars2016.js.org/

Hello World

- Mustache syntax {{ }} evaluates the variable in the data object

```
<body>
  <script src="vue.js"></script>

  <div id="app">
    <p>{{ message }}</p>
  </div>

  <script>
    new Vue({
      el: '#app',
      data: {
        message: 'Hello World!'
      }
    })
  </script>
</body>
```



```
<div id="app">
  <p>Hello World!</p>
</div>
```

Filters

- Apply a function to a variable which is evaluated

```
<div id="app">
  <p>{{ message | upperCase }}</p>
</div>
```



```
<div id="app">
  <p>HELLO WORLD!</p>
</div>
```

```
data: {
  message: 'Hello World!',
},
filters: {
  upperCase: function(text) {
    return text.toUpperCase();
  }
}
```

Computed properties

- Derive properties from other properties

```
<div id="app">
  <p>{{ completeMessage }}</p>
</div>
```



```
<div id="app">
  <p>Hello World!</p>
</div>
```

```
data: {
  message: 'Hello '
},
computed: {
  completeMessage: function() {
    return this.message + 'World!';
  }
}
```

v-if and v-show

- Show or hide elements based on a condition

```
<div id="app">
  <p v-if="isEnabled">Hello</p>
  <p v-show="isVisible">World</p>
</div>
```

```
<div id="app">
  <!---->
  <p style="display: none;">World</p>
</div>
```

```
data: {
  isEnabled: false,
  isVisible: false
}
```

v-if and v-show

- Show or hide elements based on a condition

```
<div id="app">
  <p v-if="isEnabled">Hello</p>
  <p v-show="isVisible">World</p>
</div>
```

```
<div id="app">
  <p>Hello</p>
  <p>World</p>
</div>
```

```
data: {
  isEnabled: true,
  isVisible: true
}
```

- v-else or v-else-if is possible after v-if

v-for

- Duplicate html elements based on an input array

```
<div id="app">
  <p v-for="word in words">{{word}}</p>
</div>
```



```
<div id="app">
  <p>Hello</p>
  <p>World</p>
</div>
```

```
data: {
  words: ["Hello", "World"]
}
```

v-html

- Insert HTML into the DOM

```
<div id="app" v-html="htmlMessage">
</div>
```

```
<div id="app">
  <p>Hello World</p>
</div>
```

```
data: {
  htmlMessage: "<p>Hello World</p>"
}
```

v-on & methods

- Listen to events and call a method

```
<div id="app">
  <button v-on:click="changeMessage">
    Change greeting
  </button>
  <p>{{message}}</p>
</div>
```

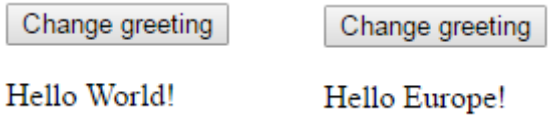


```
data: {
  message: "Hello World!"
},
methods: {
  changeMessage: function() {
    this.message = "Hello Europe!";
  }
}
```

v-on & methods

- Call other methods within a method

```
<div id="app">
  <button v-on:click="changeMessage">
    Change greeting
  </button>
  <p>{{message}}</p>
</div>
```

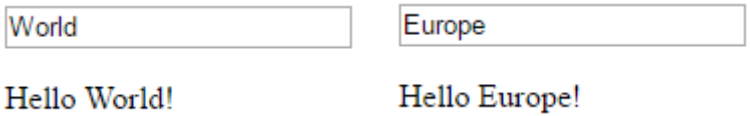

```
data: {
  message: "Hello World!"
},
methods: {
  greet: function(placeName) {
    return "Hello " + placeName + "!";
  },
  changeMessage: function() {
    this.message = this.greet("Europe");
  }
}
```

http://osgl.ethz.ch/training/introduction-to-vue-js/v-on-and-methods2.html

v-model

- Reflects the current state of form elements

```
<div id="app">
  <input v-model="greeting"></input>
  <p>Hello {{greeting}}!</p>
</div>
```



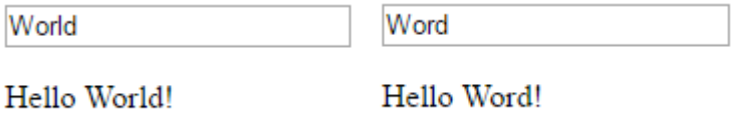

```
data: {
  greeting: 'World'
}
```

- v-model is also used for text-areas, check boxes, radio buttons, and dropdowns

watch

- Observe changes of variables in the data object

```
<div id="app">
  <input v-model="greeting"></input>
  <p>Hello {{greeting}}!</p>
</div>
```




```
data: {
  greeting: 'World'
},
watch: {
  greeting: function(newGreeting, oldGreeting) {
    console.log("Old greeting: " + oldGreeting);
    console.log("New greeting: " + newGreeting);
  }
}
```

```
Old greeting: World
New greeting: Word
```

:<HTML attribute>

- Set the attribute value of an HTML element

```
<div id="app">
  <a :href="link">Hello World</p>
</div>
```



```
<div id="app">
  <a href="http://www.world.com/">Hello World</p>
</div>
```

```
data: {
  link: 'http://www.world.com/'
}
```

:class

- Specify a condition when to add a class

```
<style>
   .large {
     font-weight: bold;
     font-size: 20;
   }
</style>
```

```
<div id="app">
  <p :class="{ large: isActive }">Hello World!</p>
</div>
```



```
<div id="app">
  <p class="large">Hello World!</p>
</div>
```

```
data: {
  isActive: true
}
```

:style

- … or specify style properties based on variables

```
<div id="app">
  <p :style="{ fontWeight : weight,
               fontSize : size }">
    Hello World!
  </p>
</div>
```



```
<div id="app">
  <p style="font-weight: bold;
            font-size: 20;">
    Hello World!
  </p>
</div>
```

```
data: {
  weight: "bold",
  size: 20
}
```

Components

```
<div id="app">
  <my-message></my-message>
</div>

new Vue({
  el: '#app',
  components: {
    'my-message': {
      template: '<p>{{ message }}</p>',
      data: function() {
        return {
          message: "Hello World!"
        };
      }
    }
  }
})
```

```
<div id="app">
  <p>Hello World!</p>
</div>
```

- Data in components needs to be a function returning an object
- Convention: name of components are lowercase with hyphen

Components - Reusability

- Components can be reused in an application and also by other applications

```
var myMessage = {
  template: '<p>{{ message }}</p>',
  data: function() {
    return {
      message: "Hello World!"
    };
  }
}


new Vue({
  el: '#app',
  components: {
    'my-message': myMessage
  }
})
```

Components - Reusability

- Components can be reused in an application and also by other applications

MyMessage.js

```
module.exports = {
  template: '<p>{{ message }}</p>',
  data: function() {
    return {
      message: "Hello World!"
    };
  }
}
```

```
new Vue({
  el: '#app',
  components: {
    'my-message': require('MyMessage.js')
  }
})
```

Components - Properties

- Parent components can pass data to child components

```
<div id="app">
  <my-message prefix="Hello" :suffix="greeting"></my-message>
</div>
```



```
<div id="app">
  <p>Hello World!</p>
</div>
```

```
new Vue({
  el: '#app',
  components: {
    'my-message': {
      template: '<p>{{ prefix + " " + suffix + "!" }}</p>',
      props: ["prefix", "suffix"]
    }
  },
  data: {
    greeting: "World"
  }
})
```

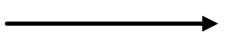
http://osgl.ethz.ch/training/introduction-to-vue-js/component-properties.html

Components - Events

- Child components can emit events for parent components

```
<div id="app">
  <my-button v-on:clicked="buttonClicked"></my-button>
</div>


new Vue({
  el: '#app',
  components: {
    'my-button': {
      template: '<button v-on:click="onClick">Click me</button>',
      methods: {
        onClick: function() {
          this.$emit("clicked", "Hello World!");
        }
      }
    }
  },
  methods: {
    buttonClicked: function(message) {
      console.log(message);
    }
  }
})
```



http://osgl.ethz.ch/training/introduction-to-vue-js/component-events.html

Components - Events

- For non parent-child communication (e.g. sibling components, far away components), you can use either an event bus or shared states

EventBus.js

```
module.exports = new Vue();
```

```
var bus = require("EventBus.js");
bus.$emit("clicked", "Hello World");
```

```
var bus = require("EventBus.js");
bus.$on("clicked", function(message) {
  console.log(message);
});
```

Components - Shared state

- For non parent-child communication (e.g. sibling components, far away components), you can use either an event bus or shared states

SuffixState.js

```
module.exports = {
  suffix: 'World'
}
```

```
data: function() {
  return {
    prefix: "Hello",
    shared: require("SuffixState.js")
  }
}
```

```
data: function() {
  return {
    prefix: "Good bye",
    shared: require("SuffixState.js")
  }
}
```

Mixins

- Reuse data, methods, filters, etc. in components

UpperCase.js

```
module.exports = {
  filters: {
    upperCase: function(text) {
      return text.toUpperCase();
    }
  }
}
```

```
<div id="app">
  <p>{{ message | upperCase }}</p>
</div>
```

```
<div id="app">
  <p>HELLO WORLD!</p>
</div>
```

```
new Vue({
  el: '#app',
  data: {
    message: 'Hello World!'
  },
  mixins: [
    require("UpperCase.js")
  ]
})
```

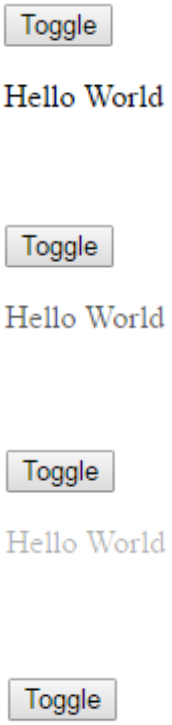
http://osgl.ethz.ch/training/introduction-to-vue-js/mixins.html

Transitions

```
<style>
  .fade-enter-active, .fade-leave-active {
    transition: opacity .5s
  }
  .fade-enter, .fade-leave-to {
    opacity: 0;
  }
</style>

<div id="app">
  <button v-on:click="toggle">Toggle</button>
  <transition name="fade">
    <p v-if="isEnabled">Hello World</p>
  </transition>
</div>

data: {
  isEnabled: true
},
methods: {
  toggle: function() {
    this.isEnabled = !this.isEnabled;
  }
}
```



http://osgl.ethz.ch/training/introduction-to-vue-js/transitions.html

Lifecycle of components

- Hook to functions before components are initialised, updated, and destroyed

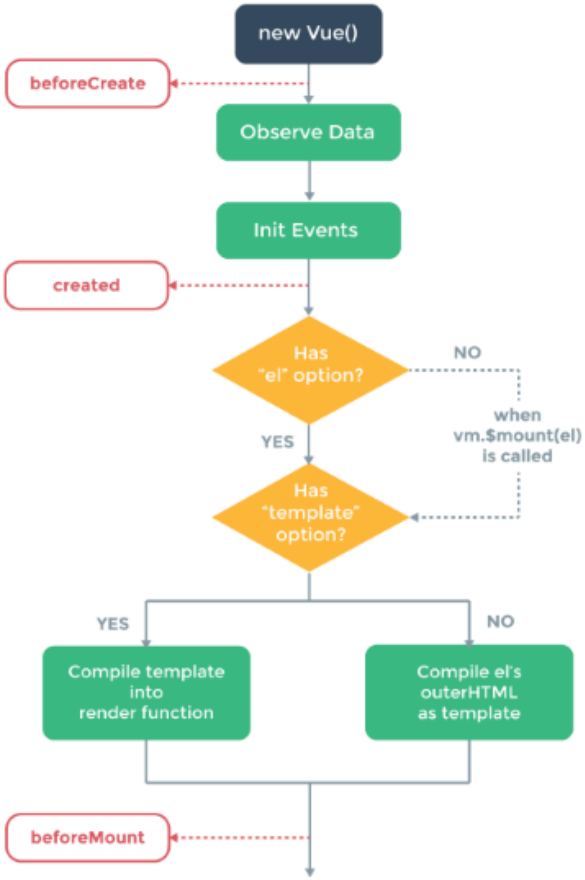
```
<div id="app">
  <p>{{ message }}</p>
</div>
```

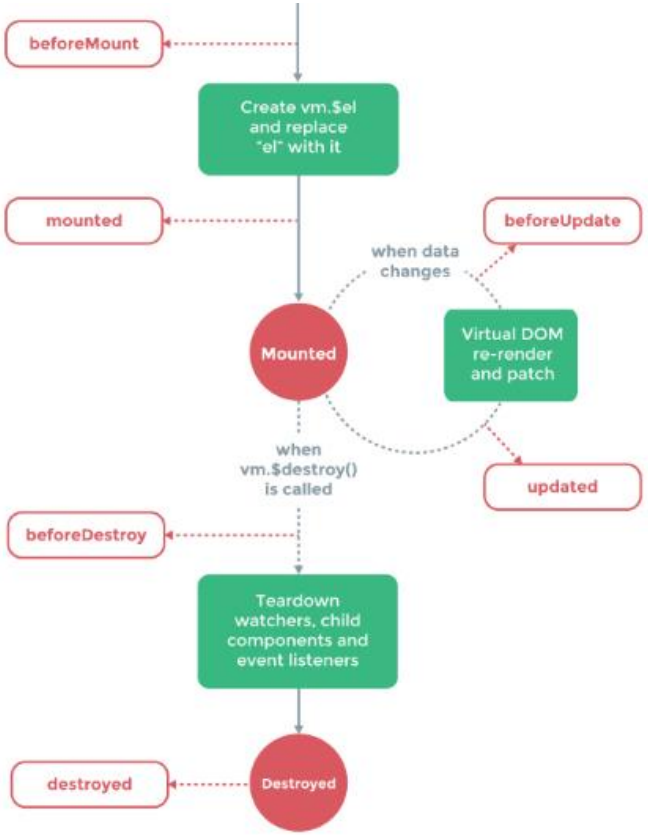


```
<div id="app">
  Hello Europe!
</div>
```

```
data: {
  message: 'Hello World!'
},
created: function() {
  this.message = 'Hello Europe!'
}
```

Lifecycle of components





https://vuejs.org/v2/guide/instance.html#Lifecycle-Diagram

Vueify

```
<template>
  <p>{{ message }}</p>
</template>

<style>
  p {
    font-weight: bold;
  }
</style>

<script>
  module.exports = {
    data: function() {
      return {
        message: 'Hello World!'
      }
    }
  }
</script>
```

HelloWorld.vue

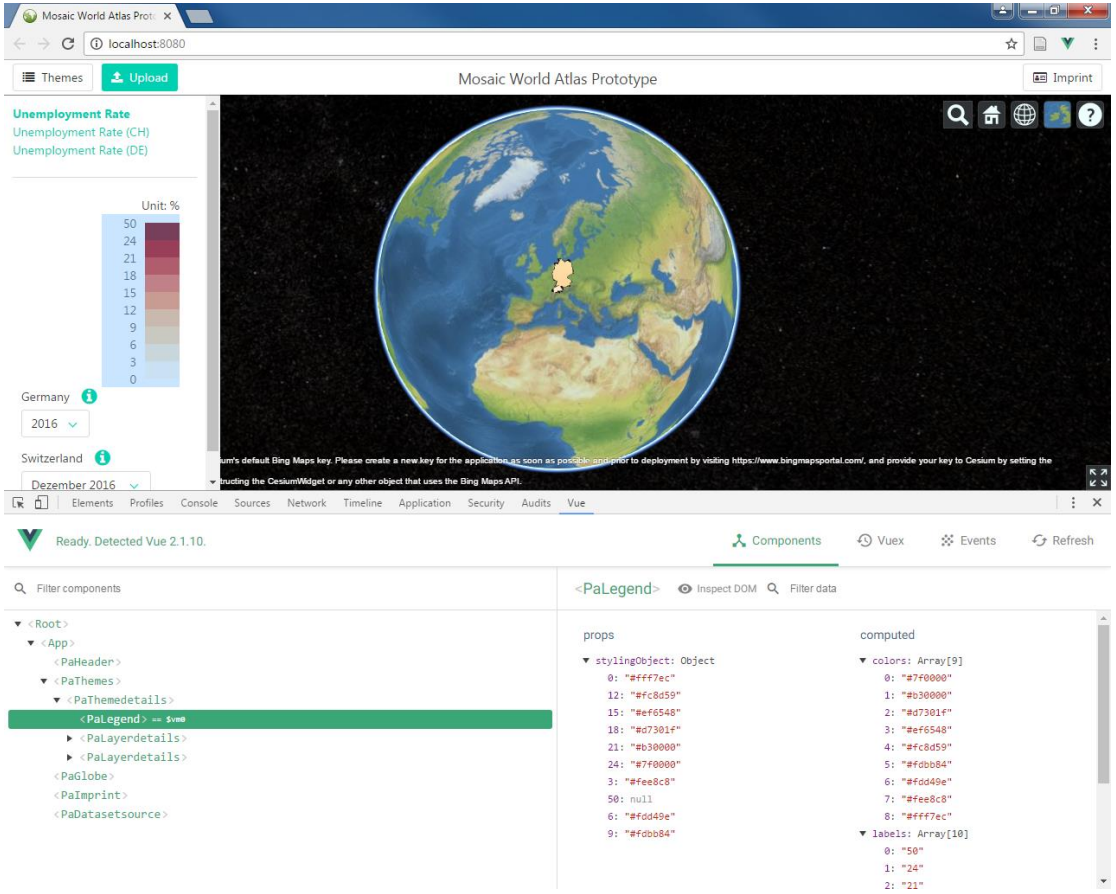
The template needs to have a single root element.

The style is set globally, but it can also be scoped

- .vue files are loaded with vue-loader in webpack

Vue dev tools

- Chrome app which helps to inspect the current state of components in development mode



Vue-cli and vue-templates

- Inititialises a folders and configuration files for getting started with a new Vue.js project via the command line

- npm install vue-cli
- node <path to vue-cli> init webpack <project_name> → installs the template called webpack

➢ Now you are ready to start your first Vue.js project!