LEARNING vuejs2

#vuejs2

Table of Contents

About

Chapter 1: Getting started with vuejs2

Remarks

Vue.js 2.X is a fast, lightweight framework for building user interfaces in Javascript. It is similar in many ways to Angular and React, and like those libraries can be used either to provide just the view layer (the V in MVC) for a larger application, or (in combination with other tools) to create fully-featured single-page web applications.

Vue takes a more minimal approach than Angular or React, however; it relies more on traditional web technologies (e.g. it allows JSX but encourages templating in HTML) and the use of outside libraries to complement its core functionalities. This gives it a faster learning curve than many other frameworks, and allows developers to continue using their preferred tools to accomplish tasks within Vue.

Web developers familiar with other front-end frameworks will find many familiar features in Vue, from a component-based paradigm and the virtual DOM to conditional rendering with directives such as `v-if`, `v-show`, and `v-hide`. These features are combined with innovations such as single-page templates and computed component properties.

Vue 2.X retains 90% of the API of 1.X, with the most significant changes involving renaming of component lifecycle hooks and the removal of support for fragment component instances. A migration helper is available for developers who want to upgrade from earlier versions of Vue.

Examples

"Hello, World!" Program

It's similar to Vue.js version 1, version 2 can be directly embedded in a single html file.
To include Vue.js2 in html file, make sure you have the script file included in the HTML. For example, use the following HTML. It should display the Hello message correctly.

```
<div id="app">{{message}}</div>
<script src="https://cdnjs.cloudflare.com/ajax/libs/vue/2.2.0/vue.js"></script>
<script>
    new Vue({
      el: '#app',
      data: {
        message: 'Hello Vue.js2'
      }
    })
</script>
```

See a jsfiddle demo for this example.

Hello World: Getting started with vue-cli

1. Install vue-cli:

   ```
   npm install -g vue-cli
   ```

2. start a project like:

   ```
    vue init <template> <project-name>
   ```

   where `<template>`:

   1. webpack - A full-featured Webpack + vue-loader setup with hot reload, linting, testing & css extraction.

   2. webpack-simple - A simple Webpack + vue-loader setup for quick prototyping.

   3. browserify - A full-featured Browserify + vueify setup with hot-reload, linting & unit testing.

   4. browserify-simple - A simple Browserify + vueify setup for quick prototyping.

   5. simple - The simplest possible Vue setup in a single HTML file

   For this example I'll use `webpack`

3. The `vue-cli` will let you go through a series of yes/no questions after which you will have a project ready with scaffolding.

4. `cd` into the project directory, it is the `<project-name>` in `vue init <template> <project-name>` and run `npm install`.

5. After the installation, run `npm run dev`.

Your hello world application is ready!

Read Getting started with vuejs2 online: https://riptutorial.com/vuejs2/topic/9276/getting-started-with-vuejs2

Chapter 2: Autocomplete

Examples

Create an autocomplete with Vuejs 2

HTML

```
<div :class="classes">
    <input v-model="compValue"
           type="text"
           class="form-control"
           @keydown.enter = 'enter'
           @keydown.down = 'down'
           @keydown.up = 'up'
           @input = "change"
    >
    <ul :class="dropDownClasses">
        <li v-for="(suggestion, index) in matches"
            v-html="suggestion"
            v-on:mouseover="mouseOverLi(index)"
            :class="{'active': isActive(index)}"
            @click="suggestionClick(index)"
        >
        </li>
    </ul>
</div>
```

Script

```
<script>
    export default {
        data: function() {
            return {
                compValue: this.value,
                current: 0,
                open: false,
            }
        },

        props: {
            value: {
                type: String,
                default: '',
            },

            suggestions: {
                type: Array,
                default: []
            },

            disabled: {
                type: Boolean,
                default: false,
            },
            readonly: {
                type: Boolean,
                default: false,
            },
        },

        computed: {
            classes: function() {
                return {
                    'br-auto-complete': true,
                    'open': this.openSuggestion
                };
            },

            dropDownClasses: function() {
                return {
                    'dropdown-menu': true,
                };
            },

            matches: function() {
                return this.suggestions.filter((str) => {
                    return str.toUpperCase().indexOf(this.compValue.toUpperCase()) >= 0;
                });
            },

            openSuggestion() {
                return this.compValue !== "" &&
                    this.matches.length != 0 &&
                    this.open === true;
            },
        },

        methods: {
            //When enter pressed on the input
            enter: function() {
                this.compValue = this.matches[this.current];
                this.open = false;
            },

            //When up pressed while suggestions are open
            up: function() {
                if(this.current > 0)
                    this.current--;
            },

            //When up pressed while suggestions are open
            down: function() {
                if(this.current < this.matches.length - 1)
                    this.current++;
            },

            //For highlighting element
            isActive: function(index) {
                return index === this.current;
            },

            mouseOverLi: function(index){
                this.current = index;
            },
            //When the user changes input
            change: function() {
                if (this.open == false) {
                    this.open = true;
                    this.current = 0;
                }
                this.$emit('input', this.compValue);
            },

            //When one of the suggestion is clicked
            suggestionClick: function(index) {
                this.compValue = this.matches[index];
                this.open = false;
                this.$emit('input', this.compValue);
            },
        },
    }
</script>
```

Read Autocomplete online: https://riptutorial.com/vuejs2/topic/9587/autocomplete

Chapter 3: Autocomplete

Examples

Autcomplete input

This is a small example of an autocomplete input.

CSS

```
.autocomplete {
  position: relative;
}
.autocomplete-list {
  position: absolute;
  z-index: 2;
  top: 25px;

  min-width: 150px;
  margin: 0;
  padding: 0;

  list-style: none;

  border: 1px solid #eee;
  border-radius: 4px;
  background-color: #fff;
}
.autocomplete-list li {
  margin: 0;
  padding: 8px 15px;

  border-bottom: 1px solid #eee;
}
.autocomplete-list li:last-child {
  border-bottom: 0;
}
.autocomplete-list li:hover,
.autocomplete-list li.active {
  background-color: #f5f5f5;
}
```

Javascript

```
const Autocomplete = Vue.component('autocomplete', {
  template: '#autocomplete-tpl',
  props: ['items'],
  data: function() {
    return {
      inputValue: '',
      searchMatch: [],
      selectedIndex: -1
    }
  },
  computed: {
    listToSearch() {
      if(typeof this.items !== 'undefined' && this.items.length > 0) {
        return this.items;
      } else {
        return [
          'input',
          'h1',
          'h2',
          'span',
          'div',
          'textarea',
          'margin',
          'padding',
          'background',
          'background-color',
          'background-size',
          'background-repeat'
        ]
      }
    }
  },
  watch: {
    inputValue(val) {
      this.searchMatch = [];
      if(this.inputValue !== '') {
        this.searchMatch = this.listToSearch.filter((el) => el.indexOf(val) >= 0);
      }
      if (this.searchMatch.length === 1 && this.inputValue === this.searchMatch[0]) {
        this.searchMatch = [];
      }

    }
  },
  methods: {
    moveDown() {
      if(this.selectedIndex < this.searchMatch.length-1) {
        this.selectedIndex++;
      }
    },
    moveUp() {
      if(this.selectedIndex !== -1) {
        this.selectedIndex--;
      }
    },
    selectItem(index) {
      this.selectedIndex = index;
    },
    chooseItem() {
      if(this.selectedIndex !== -1) {
        this.inputValue = this.searchMatch[this.selectedIndex];
        this.selectedIndex = -1;
      }
    }
  }
});


new Vue({
  el: '#app'
});
```

HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Autocomplete</title>
    <script type="text/x-template" id="autocomplete-tpl">
  <div class="autocomplete">
    <input @keydown.13="chooseItem" @keydown.40="moveDown" @keydown.38="moveUp" v-
model="inputValue" type="text">


      <ul class="autocomplete-list" v-if="searchMatch.length > 0">
        <li :class="{active: selectedIndex === index}"v-for="(result, index) in searchMatch"
@click="selectItem(index), chooseItem()">
          {{result}}
      </li>
      </ul>
      </div>
    </script>
  </head>
  <body>


    <div id="app">
      Own items:<br />
      <autocomplete :items="['all', 'auto', 'complete', 'items']"></autocomplete>
      <br /><br />
      Standard items:<br />
      <autocomplete></autocomplete>
    </div>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/vue/2.3.3/vue.js"></script>

  </body>
</html>
```

Read Autocomplete online: https://riptutorial.com/vuejs2/topic/10148/autocomplete

Chapter 4: Cascading Dropdown

Introduction

How to build dropdown's that are dependent on each other.

Examples

Car Models of Make

HTML (classes used are based on Semantic-UI)

```
<div id="app" class="ui grid">
  <div class="row">
    <div class="column">
      <div class="ui label">Vechicle Make</div>
      <select class="ui dropdown" v-model="make" id="vehicle-makes">
        <option v-for="option in makes_options" v-bind:value="option.id">
          {{ option.text }}
        </option>
      </select>
    </div>
  </div>
  <div class="row">
    <div class="column">
      <div class="ui label">Vechicle Model</div>
      <select class="ui dropdown" id="vehicle-models" v-model="model">
        <option
          v-for="option in model_options[make]"
          :value="option.id"
          :key="option.id"
        >
          {{ option.text }}
        </option>
      </select>
    </div>
  </div>
</div>
```

Javascript

```
<script>
var model_options = {
  1: [{ text: "Accord", id: 1 }, { text: "Civic", id: 2 }],
  2: [{ text: "Corolla", id: 3 }, { text: "Hi Ace", id: 4 }],
  3: [{ text: "Altima", id: 5 }, { text: "Zuke", id: 6 }],
  4: [{ text: "Alto", id: 7 }, { text: "Swift", id: 8 }]
};

var makes_options = [
  { text: "Honda", id: 1 },
  { text: "Toyota", id: 2 },
  { text: "Nissan", id: 3 },
  { text: "Suzuki", id: 4 }
];

var vm_makes = new Vue({
  el: "#app",
  data: {
    make: null,
    model: null,
    makes_options: makes_options,
    model_options: model_options,
  },
  watch: {
    make: function(event) {
      $('#vehicle-models').dropdown('clear');
    }
  }
});

// Eveything works fine if I remove this...
$('.ui.dropdown').dropdown();
</script>
```

Read Cascading Dropdown online: https://riptutorial.com/vuejs2/topic/10405/cascading-dropdown

Chapter 5: Filter in vuejs

Examples

Filter Functionality in Vuejs

```
<!doctype html>

<html>

<head>
    <title>Page Title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="initial-scale=1.0">
    <script src="vue.js"></script>
    <style>
        * {font-family: sans-serif;}
        #app {margin: 0 auto; width:500px;}
        button {
            background: #ccc;
            color: #fff;
            -webkit-appearance:none;
            background: #000;
            border: 0;
            padding: 10px;
            text-align: center;
            width: 49%;
        }
        .searchText{height: 25px;width: 386px;}
        .addItem {height: 30px;width: 226px; padding: 0 10px }
        button.active {
            background: #94d464;
            color: #fff;
        }
        .itemBox {margin: 0 0 10px 0;padding: 0;border: 1px dashed #000;}
        .itemBox li:first-child{background: #000; color:#fff; text-align: center;}
        .itemBox li:first-child span{background: #fff; color:#000;}
        .itemBox li{background: #f9f9f9; padding:10px; list-style: none;border-bottom: 1px
dashed #000;}
        .itemBox li span { float: right;display: block;background: #94d464;height:
35px;margin: -8px -9px 0 0;width: 79px;text-align: center;line-height: 35px;}
    </style>
</head>

<body>
    <div id="app">
        <h2 v-bind:title="h1">{{h1}}</h2>
        <div id="tabs">
            <button v-on:click="tabfn(true)" v-bind:class="{active : tab}">Show</button>
            <button v-on:click="tabfn(false)" v-bind:class="{active : !tab}">Hide</button>
<br><br>
            <div id="food-item-list" v-if="tab">
                <label for=""> Search Food : <input type="text" class="searchText" v-
model="searchText" @keyup="searchFood"></label>
                <h3>Food Item</h3>
                <ul class="itemBox">
                    <!--<food-item v-for="food in list" :foodlist="food" :searchtxt
```

```
="searchText"></food-item>-->
                    <li>Food List </li>
                    <li v-for="food in bindList">{{food.item}} <span>□
{{food.price}}</span></li>
                </ul>
                <input type="text" v-model="addItem" class="addItem" placeholder="Enter food
item name">
                <button v-on:click="addItemFn">Add your item</button>
            </div>
            <div v-else>
                <p>No item have in this section</p>
            </div>
        </div>

    </div>
</body>
<script>

    var data = {
        h1: "Vue js five.html (Filters functionality)",
        tab: true,
        list: [{"item": "Kathiyavadi",price:"200"}, {"item": "Gujrati",price:"180"}, {"item":
"Panjabi",price:"150"}, {"item": "South-Indian",price:"120"}, {"item":
"Bangali",price:"100"}],
        addItem: "",
        searchText: "",
        htmlData: "<p>Your order will surve in just 2 min</p>",
        price:"$",
        total : 1000,
        bindList:[]
    }
    var app = new Vue({
        el: "#app",
        data: data,
        methods: {
            tabfn: function(bolian) {
                this.tab = bolian
            },
            addItemFn: function() {
                if (!this.addItem.length) return
                this.list.push({
                    "item": this.addItem
                })
                this.addItem = ""
                this.searchFood()
            },
            searchFood : function(value){
                //this.bindList = this.list;
                this.bindList = []
                for(var i=0; i < this.list.length; i++){
                    if(this.list[i].item.indexOf(this.searchText) != -1){
                        this.bindList.push(this.list[i])
                    }
                    console.log(this.bindList)
                }
            }
        },
        computed:{
            reverseH1_:function(){
                return this.h1.split('').reverse().join('');
            },
            getTotal:{
                get: function(){
                    return this.price + this.total
                },
                set : function(newvalue){
                    this.price = "#"
                }
            }
        },
        filters:{

        },
    })

    app.searchFood();


</script>

</html>
```

Color changes

```
<!doctype html>

<html>

<head>
    <title>Page Title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="initial-scale=1.0">
    <script src="vue.js"></script>
    <style>
        * {font-family: sans-serif;}
        #app {margin: 0 auto; width:500px;}
        button {
            background: #ccc;
            color: #fff;
            -webkit-appearance:none;
            background: #000;
            border: 0;
            padding: 10px;
            text-align: center;
            width: 49%;
        }
        .searchText{height: 25px;width: 386px;}
        .addItem {height: 30px;width: 226px; padding: 0 10px }
        button.active {
            background: #94d464;
            color: #fff;
        }
        .itemBox {margin: 0 0 10px 0;padding: 0;border: 1px dashed #000;}
        .itemBox li:first-child{background: #000; color:#fff; text-align: center;}
        .itemBox li:first-child span{background: #fff; color:#000;}
        .itemBox li{background: #f9f9f9; padding:10px; list-style: none;border-bottom: 1px
dashed #000;}
        .itemBox li span { float: right;display: block;background: #94d464;height:
35px;margin: -8px -9px 0 0;width: 79px;text-align: center;line-height: 35px;}
    </style>
```

```
</head>

<body>
    <div id="app">
        <h2 v-bind:title="h1">{{h1}}</h2>
        <input type="range" v-model="range" min="10" step="1" max="100" @input="manage">
        <div :style="style"></div>
    </div>
</body>
<script>
    var data = {
        h1:"Color manage",
        range:10,
        style:{"height":"100px","width":"130px","background":"rgb(0, 0, 0)"}
    }
    var app = new Vue({
        el: "#app",
        data: data,
        methods: {
            manage:function(value){
                console.log(this.range)
                this.style["background"] = "rgb(0, "+this.range+", 0)"
            }
        },
        computed:{

        },
        filters:{

        },
    })


</script>

</html>
```

Tab functionality in VueJs

```
<!doctype html>

<html>

<head>
    <title>Page Title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="initial-scale=1.0">
    <script src="vue.js"></script>
    <style>
        * {
            font-family: sans-serif;
        }

        #app {
            margin: 0 auto;
            width: 500px;
        }

        ul {
            list-style: none;
            margin: 0;
            padding: 0;
        }

        ul li {
            display: inline-block;
            padding: 10px 20px;
            border: 1px solid #000;
            margin-left: 10px;
            border: 1px solid #000;
        }

        ul li.active {
            display: inline-block;
            padding: 10px 20px;
            background: #94d464;
        }
        .tab-title li {cursor: pointer; border-bottom: none;}

    </style>
</head>

<body>
    <div id="app">
        <h2 v-bind:title="h1">{{h1}}</h2>
        <div id="tab">
            <ul class="tab-title">
                <li v-for="(title, index) in tab" @click="activeTab(index)" :class="{active:
(index == activeTabIndex)}">{{title.title}}</li>
            </ul>
            <ul>
                <li v-for="(title,index) in tab" v-if="index ==
activeTabIndex">{{title.content}}</li>
            </ul>
        </div>
    </div>
</body>
<script>
    var data = {
        h1: "Tab system",
        activeTabIndex: 0,
        tab: [{
            title: "one",
            content: "Lorem ipsum dolor sit amet, consectetur adipisicing elit. Inventore aut
provident cum rerum! Vero nemo error nesciunt sunt illo ea iste porro pariatur necessitatibus!
Quidem unde voluptatem animi cum, adipisci.Lorem ipsum dolor sit amet, consectetur adipisicing
elit. Inventore aut provident cum rerum! Vero nemo error nesciunt sunt illo ea iste porro
pariatur necessitatibus! Quidem unde voluptatem animi cum, adipisci."
        }, {
            title: "two",
            content: "Lorem ipsum dolor sit amet, consectetur adipisicing elit. Inventore aut
provident cum rerum! Vero nemo error nesciunt sunt illo ea iste porro pariatur necessitatibus!
Quidem unde voluptatem animi cum, adipisci."
        }, {
            title: "three",
            content: "Lorem ipsum dolor sit amet, consectetur adipisicing elit. Inventore aut
provident cum rerum! Vero nemo error nesciunt sunt illo ea iste porro pariatur necessitatibus!
Quidem unde voluptatem animi cum, adipisci.Lorem ipsum dolor sit amet, consectetur adipisicing
elit. Inventore aut provident cum rerum! Vero nemo error nesciunt sunt illo ea iste porro
pariatur necessitatibus! Quidem unde voluptatem animi cum, adipisci.Lorem ipsum dolor sit
amet, consectetur adipisicing elit. Inventore aut provident cum rerum! Vero nemo error
nesciunt sunt illo ea iste porro pariatur necessitatibus! Quidem unde voluptatem animi cum,
adipisci."
        }, {
            title: "four",
            content: "consectetur adipisicing elit. Inventore aut provident cum rerum! Vero
nemo error nesciunt sunt illo ea iste porro pariatur necessitatibus! Quidem unde voluptatem
animi cum, adipisci."
        }]
    }
    var app = new Vue({
        el: "#app",
        data: data,
        methods: {
            activeTab: function(indx) {
                this.activeTabIndex = indx
            }
        },
        computed: {

        },
        filters: {

        },
    })

</script>

</html>
```

Read Filter in vuejs online: https://riptutorial.com/vuejs2/topic/9718/filter-in-vuejs

Chapter 6: Props

Introduction

This small example shows you how to pass props to a component.

More information: https://vuejs.org/v2/guide/components.html#Passing-Data-with-Props

Examples

VueJS 2 props example

JavaScript:

```
Vue.component('props-component', {
    template: '#props-component-template',
    // array of all props
    props: ['myprop', 'calcprop']
});


new Vue({
    el: '#app'
});
```

HTML:

```
<div id="app">
    <template id="props-component-template">
        <div>Props: {{myprop}} - {{calcprop}}</div>
    </template>

    <props-component
        myprop="prop_value"
        :calcprop="Math.random()">
    </props-component>
</div>
```

Read Props online: https://riptutorial.com/vuejs2/topic/10123/props

Chapter 7: Routing

Introduction

VueJS Routing with vue-router.

Documentation: https://router.vuejs.org/en/

Examples

Hash Router

JavaScript:

```
const MainPage = {
  template: '#mainpage-template'
}
const Page1 = {
  template: '#page1-template'
}
const Page2 = {
  template: '#page2-template'
}
const Page3 = {
  template: '#page3-template'
}

const router = new VueRouter({
  mode: 'hash',
  routes: [{
      path: '/',
      component: MainPage
    },
    {
      path: '/page1',
      component: Page1
    },
    {
      path: '/page2',
      component: Page2
    },
    {
      path: '/page3',
      component: Page3
    }
  ]
})

new Vue({
  router: router,
  el: '#app'
});
```

HTML:

```
<template id="mainpage-template">
  <div>
    mainpage
  </div>
</template>
<template id="page1-template">
  <div>
    Page 1
  </div>
</template>
<template id="page2-template">
  <div>
    Page 2
  </div>
</template>
<template id="page3-template">
  <div>
    Page 3
  </div>
</template>


<div id="app">
  <h1>Router</h1>
  <ul>
    <li><router-link to="/">mainpage</router-link></li>
    <li><router-link to="/page1">/page1</router-link></li>
    <li><router-link to="/page2">/page2</router-link></li>
    <router-link tag="li" to="/page3">/page3</router-link>
  </ul>

  <router-view></router-view>
</div>
```

different ways of defining routes

route/index.js

```
  import About from '@/components/About'

  const router = new Router({

    routes: [
     {
       path: '/',
       name: 'home',
       component: {template: "<div>Home</div>"}
     },

     {
       path: '/about',
       component: About
     },

     require('./work').default,

     // - NOT FOUND
     {path: '/404', component: {template: "<div>404 not found.</div>"}},
     {path: "/*", redirect: "/404"}
    ]
  });

  export default router
```

route/work.js

```
import Work from '@/components/Work/Work';
import Workitem from '@/components/Work/Workitem';

export default {
  path: '/work',
  name: 'work',
  component: Work,
  children: [
    {path: '/all', component: {template: '<div>Some text</div>'}},
    {path: ':id', name: 'work.view', component: Workitem},
    {path: ':id/edit', name: 'work.edit', component: Workitemedit},
  ],
  meta: {requiresAuth: true}
}
```

Read Routing online: https://riptutorial.com/vuejs2/topic/10124/routing

Credits

S. No	Chapters	Contributors
1	Getting started with vuejs2	Amresh Venugopal, Community, jaredsk, YChi Lu
2	Autocomplete	Markus Lenz
3	Cascading Dropdown	Storm
4	Filter in vuejs	Parth Jasani
5	Props	Phil
6	Routing	Phil, remmargorp