

Tehnici Web

CURSUL 11

Semestrul I, 2018-2019
Carmen Chirita

<https://sites.google.com/site/fmitehniciweb/>

VUE.JS

<https://vuejs.org/>

„Vue is a progressive framework for building user interfaces”

- Approachable
- Versatile
- Performant

Cum se introduce Vue.js?

- Se descarca fișierul de pe site-ul oficial și se insereaza în fișierul html folosind tagul `<script>`
- Se include direct din CDN

```
<script src="https://cdn.jsdelivr.net/npm/vue"></script>
```

Vue.js - Funzionalità

Reactive Interfaces

Declarative Rendering

Data Binding

Directives

Template Logic

Components & Nesting

Event Handling

Computed Properties

CSS Transitions & Animation

Custom Filters

Vue.js - Declarative rendering (redare declarativa)

Permite sa afisez în mod declarativ date în DOM folosind o sintaxa de template-uri

HTML

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

JavaScript

```
var obVue = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!'
  }
})
```

Afisare în browser

Hello Vue!

Vue.js - Declarative rendering (redare declarativa)

Pasii:

- crearea unei instante Vue: `var obVue=new Vue({});`

Parametrii:obiect de optiuni

`el: '#app'` //leagă noul obiect Vue cu elementul HTML avand id = "app"

```
data: {  
  message: 'Hello Vue!'    //continutul datelor redade  
}
```

- interpolarea textului folosind sintaxa `{{ }}`: `{{ message }}`

Elementul HTML se va schimba atunci când obiectul Vue se va schimba:

```
obVue.message="Alt mesaj";
```

Vue.js - Directive

Sunt attribute HTML speciale create cu prefixul **v-** care produc schimbări în DOM atunci când valoarea expresiei lor se modifică

Exemple:

```
<a v-bind:href="url"></a>
```

```
<p v-if="greeting">Hello!</p>
```

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

```
<span v-text="msg"></span>
```

```
<span v-once>This will never change: {{ msg }}</span>
```

Vue.js - Directive

Exemplu: directiva **v-bind**

HTML

```
<div id="app-2">  
  <span v-bind:title="message">  
    Hover your mouse over me for a few seconds  
    to see my dynamically bound title!  
  </span>  
</div>
```

JavaScript

```
var app2 = new Vue({  
  el: '#app-2',  
  data: {  
    message: 'You loaded this page on ' + new Date().toLocaleString()  
  }  
})
```

{{ }} nu pot fi folosite în interiorul atributelor HTML

Vue.js - Conditionals and Loops

Directiva **v-if** (controleaza afisarea unui element)

HTML

```
<div id="app-3">  
  <span v-if="seen">Now you see me</span>  
</div>
```

JavaScript

```
var app3 = new Vue({  
  el: '#app-3',  
  data: {  
    seen: true  
  }  
})
```

`app3.seen = false` //elementul va fi invizibil/eliminat din DOM

Vue.js - Conditionals and Loops

Directiva **v-for** (afișează o listă de elemente utilizând datele dintr-un Array)
v-for = "item în items"

HTML

```
<div id="app-4">
  <ol>
    <li v-for="todo in todos">
      {{ todo.text }}
    </li>
  </ol>
</div>
```

```
1. Learn JavaScript
2. Learn Vue
3. Build something awesome
```

JavaScript

```
var app4 = new Vue({
  el: '#app-4',
  data: {
    todos: [
      { text: 'Learn JavaScript' },
      { text: 'Learn Vue' },
      { text: 'Build something
awesome' }
    ]
  }
})
```

app4.todos.push({ text: 'New item' }) //se adauga un nou element listei

Vue.js - Two-Way Binding

Directiva v-model

- leagă valoarea elementelor HTML de datele aplicației

HTML

```
<div id="app-5">  
  <p>{{ message }}</p>  
  <input v-model="message">  
</div>
```

JavaScript

```
var app5 = new Vue({  
  el: '#app-5',  
  data: {  
    message: 'Hello Vue!'  
  }  
})
```

Hello Vue!

Hello Vue!

-modificarea obiectului Vue are ca efect
modificarea valorii elementelor HTML

-modificarea valorii elementului HTML (input)
va produce modificarea valorii prop. ob. Vue

Handling User Input

Directiva **v-on**

- ataseaza handler de evenimente care invoca metode în instantele Vue

HTML

```
<div id="app-6">  
  <p>{{ message }}</p>  
  <button v-on:click="reverseMessage">  
Reverse Message</button>  
</div>
```

!sj.euV olleH

Reverse Message

JavaScript

```
var app6 = new Vue({  
  el: '#app-6',  
  data: {  
    message: 'Hello Vue.js!'  
  },  
  methods: {  
    reverseMessage: function () {  
      this.message =  
this.message.split('').reverse().join('')  
    }  
  }  
})
```

Conditional rendering (redarea conditionata)

Directiva v-if cu <template>

<template> este folosit pentru a grupa elemente în scopul redării conditionate

HTML

```
<div id="app-7">
<template v-if="ok">
  <h1>Title</h1>
  <p>Paragraph 1</p>
  <p>Paragraph 2</p>
</template>
</div>
```

JavaScript

```
var app7 = new Vue({
  el: '#app-7',
  data: {
    ok: true
  }
})
```

Conditional rendering

Directiva `v-else`

- indica un "bloc else" pentru `v-if`;
- trebuie sa urmeze imediat după un element `v-if`

HTML

```
<div id="app-8">  
  <div v-if="ok">  
    Now you see me  
  </div>  
  <div v-else>  
    Now you don't  
  </div>  
</div>
```

Now you see me

ok: true

JavaScript

```
var app8 = new Vue({  
  el: '#app-8',  
  data: {  
    ok: true  
  }  
})
```

Now you don't

ok: false

Conditional rendering

Directiva `v-else-if`

- indica un "bloc else if" pentru `v-if`
- trebuie sa urmeze imediat după un element `v-if` sau `v-else-if`

HTML

```
<div id="app-9">
  <div v-if="type === 'A' ">
    A
  </div>
  <div v-else-if="type === 'B' ">
    B
  </div>
  <div v-else-if="type === 'C' ">
    C
  </div>
  <div v-else>
    Not A/B/C
  </div>
</div>
```

JavaScript

```
var app9 = new Vue({
  el: '#app-9',
  data: {
    type: 'A'
  }
})
```

Conditional rendering

Directiva **v-show** (controleaza afisarea unui element)

HTML

```
<h1 id="app" v-show="ok">Hello!</h1>
```

JavaScript

```
var app = new Vue({  
  el: '#app',  
  data: {  
    ok: true  
  }  
})
```

- spre deosebire de v-if, elementul cu v-show va rămâne în DOM
- v-show nu acceptă elementul <template> și nici nu funcționează cu v-else

Conditional rendering

Controlul elementelor reutilizabile folosind atributul `key`

- Vue.js poate reutiliza anumite elemente în loc sa le redea de la început;
- De exemplu, când se permite utilizatorilor sa comute intre mai multe tipuri de login

```
<template v-if="loginType === 'username'">
  <label>Username</label>
  <input placeholder="Enter your
username">
</template>
<template v-else>
  <label>Email</label>
  <input placeholder="Enter your email
address">
</template>
```

Username

Username

Email

Conditional rendering

Controlul elementelor reutilizabile folosind atributul `key`

- Daca nu se dorește reutilizarea elementelor, se va adauga un atribut `key` cu valoare unica

```
<template v-if="loginType === 'username'">
  <label>Username</label>
  <input placeholder="Enter your username"
  key="username-input">
</template>
<template v-else>
  <label>Email</label>
  <input placeholder="Enter your email
address" key="email-input">
</template>
```

List rendering (redarea listelor)

v-for cu al doilea argument pentru index

v-for="(item, index) in items"

HTML

```
<ul id="example-1">
  <li v-for="item in items">
    {{ item.message }}
  </li>
</ul>
```

JavaScript

```
var example1 = new Vue({
  el: '#example-1',
  data: {
    items: [
      { message: 'Foo' },
      { message: 'Bar' }
    ]
  }
})
```

HTML

```
<ul id="example-2">
  <li v-for="(item, index) in items">
    {{ parentMessage }} - {{ index }} -
    {{ item.message }}
  </li>
</ul>
```

JavaScript

```
var example2 = new Vue({
  el: '#example-2',
  data: {
    parentMessage: 'Parent',
    items: [
      { message: 'Foo' },
      { message: 'Bar' }
    ]
  }
})
```

- Parent - 0 - Foo
- Parent - 1 - Bar

List rendering

HTML

```
<ul id="v-for-object"
class="demo">
  <li v-for="value in object">
    {{ value }}
  </li>
</ul>
```

JavaScript

```
new Vue({
  el: '#v-for-object',
  data: {
    object: {
      firstName: 'John',
      lastName: 'Doe',
      age: 30
    }
  }
})
```

- John
- Doe
- 30

v-for cu un obiect

v-for cu doua argumente

```
<div v-for="(value, key) in object">
  {{ key }}: {{ value }}
</div>
```

```
firstName: John
lastName: Doe
age: 30
```

v-for cu trei argumente

```
<div v-for="(value, key, index) in object">
  {{ index }}. {{ key }}: {{ value }}
</div>
```

```
0. firstName: John
1. lastName: Doe
2. age: 30
```

List rendering

v-for într-un interval

HTML

```
<div id="dd">  
  <span v-for="n in 10">{{ n }}  
</span>  
</div>
```

JavaScript

```
new Vue({  
  el: '#dd'  
  .....  
})
```

Se va afisa

1 2 3 4 5 6 7 8 9 10

List rendering

v-for cu <template>

Se folosește pentru a reda un bloc de elemente multiple

HTML

```
<ul id="ii">
  <template v-for="item in
shoppingItems">
    <li>{{item.name }}</li>
    <li>{{item.price}}</li>
  </template>
</ul>
```

JavaScript

```
new Vue({
  el: '#ii',
  data: {
    shoppingItems: [
      {name: 'apple', price: '10'},
      {name: 'orange', price: '12'}
    ]
  }
})
```

- apple
- 10
- orange
- 12

List rendering - Detectarea schimbarii array-ului

Metode:

push()
pop()
shift()
unshift()
splice()
sort()
reverse()

- Baz
- Bar
- Foo

HTML

```
<ul id="example-1">  
  <li v-for="item in items">  
    {{ item.message }}  
  </li>  
</ul>
```

JavaScript

```
var example1 = new Vue({  
  el: '#example-1',  
  data: {  
    items: [  
      { message: 'Foo' },  
      { message: 'Bar' }  
    ]  
  }  
})
```

```
example1.items.push({ message: 'Baz' })  
example1.items.reverse();
```

Aceste metode modifica array-ul original

List rendering - Detectarea schimbarii array-ului

Metode:

filter()

concat()

slice()

//nu modifica
array-ul original

HTML

```
<ul id="example-1">
  <li v-for="item in items">
    {{ item.message }}
  </li>
</ul>
```

JavaScript

```
var example1 = new Vue({
  el: '#example-1',
  data: {
    items: [
      { message: 'Foo' },
      { message: 'Bar' }
    ]
  }
})
```

```
example1.items = example1.items.filter(function (item) {
  return item.message.match(/Foo/) })
```

```
example1.items.concat([{ message: 'Baz' }]);
```