Il framework Vue.js

Lorenzo Marconi

SAPIENZA
UNIVERSITÀ DI ROMA

Anno accademico 2020/2021

Cosa è Vue.js

- **Vue.js** (comunemente chiamato **Vue** e pronunciato come la parola "view") è un framework JavaScript open-source utilizzato per lo sviluppo di applicazioni Web *reattive*
- Essendo un framework front-end, esso influenza la struttura di base e lo sviluppo del codice di tutta l'applicazione
- Ha come obiettivo quello di semplificare la programmazione di una applicazione Web
- Pubblicato ufficialmente nel Febbraio del 2014

Caratteristiche principali

- Vue è un framework progressivo. Può essere integrato in modo incrementale in un progetto, a seconda delle necessità
- Si ragiona in termini di dati, variabili ed oggetti, astraendosi rispetto all'implementazione delle singole pagine
- Offre il cosiddetto *rendering dichiarativo*: permette di inserire gli elementi dell'interfaccia definiti come markup direttamente all'interno delle pagine HTML
- Si basa sulla composizione dei componenti: consente di suddividere il codice html in moduli riutilizzabili, ognuno con le sue proprietà

SAPIENZA UNIVERSITÀ DI ROMA

Includere Vue in una pagina Web

Si può farne il download seguendo questo link:
https://vuejs.org/v2/guide/installation.html, dove ci sono due versioni:

- La Development Version (versione non compressa), con modalità di debugging
- La Product Version (versione compressa), con tutti i possibili warning rimossi

Assumendo di avere nella cartella locale uno dei file dal sito di sopra:

- Collegamento alla versione non compressa:
  <**script** type = "text/javascript" src = "vue.js">
  </**script**>
- Collegamento alla versione compressa:
  <**script** type = "text/javascript" src = "vue.min.js">
  </**script**>

SAPIENZA UNIVERSITÀ DI ROMA

Obiettivo dell'esercitazione

- L'obiettivo principale di questa esercitazione è quello di prendere confidenza con Vue.js
- In particolare, discuteremo solamente alcuni degli aspetti più salienti di questo framework
- Per maggiori dettagli rimandiamo alla documentazione ufficiale: https://vuejs.org/v2/guide/

SAPIENZA UNIVERSITÀ DI ROMA

Primo esempio

Un file con estensione .html

```
<html>
    <head>
        <title> Prodotto in vendita </title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <script type="text/javascript" src="Vue/vue.min.js">
        </script>
    </head>
    <body>
        <div id="app">
            <h1>Prodotto in vendita: {{product}}</h1>
            Descrizione: {{description}}
        </div>
        <script type="text/javascript" src="mascherina.js">
        </script>
    </body>
</html>
```

Un file con estensione .js

```
var app = new Vue({
    el: '#app',
    data: {
        product: 'Mascherina',
        description: 'Mascherina chirurgica'
    }
});
```

Il cuore di Vue: Il binding

Analizziamo l'esempio fornito nella precedente slide:

- Nel file con estensione .js viene creato un nuovo oggetto tramite **new** Vue({ OPTIONS });
  dove OPTIONS contiene *parametri* opzionali per immagazzinare dati e performare azioni
- Il parametro **el:** effettua il **binding** tra l'oggetto Vue ed un elemento del DOM del documento html. Nel nostro esempio abbiamo **el:** '#app', pertanto l'oggetto di nome app verrà collegato all'elemento html con **id**=app.
- Le doppie parentesi graffe {{ }} nel codice html vengono usate come placeholders per effettivi valori contenuti all'interno del parametro **data:**. Nel nostro esempio abbiamo {{ product }} e {{ description }} che verranno quindi istanziati con 'Mascherina' e 'Mascherina chirurgica' nel risultante documento html.
  Chiaramente in **data:** possono essere presenti varie *proprietà* contenenti numeri, array e altri oggetti annidati

SAPIENZA UNIVERSITÀ DI ROMA

Secondo esempio

Un file con estensione .html

```
<html>
    <head>
        <title> Prodotto in vendita </title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <script type="text/javascript" src="Vue/vue.min.js">
        </script>
    </head>
    <body>
        <div id="app">
            <img v-bind:src="image" />
        </div>
        <script type="text/javascript" src="mascherina.js">
        </script>
    </body>
</html>
```

Un file con estensione .js

```
var app = new Vue({
    el: '#app',
    data: {
        image: './assets/nera.png'
    }
});
```

Binding di attributi html

Analizziamo l'esempio fornito nella precedente slide:

- Nel file html troviamo:
  v-bind:src="image"
  La *direttiva* **v-bind** effettua il binding tra i dati contenuti in un oggetto Vue e l'attributo html (in questo caso src), in cui l'attuale placeholder (in questo caso "image") verrà sostituito con un effettivo valore
- Come prima, il parametro **el:** effettua il **binding** tra l'oggetto Vue ed un elemento del DOM del documento html. Nel nostro esempio abbiamo **el:** '#app', pertanto l'oggetto di nome app verrà collegato all'elemento html con **id**=app.
- Grazie alla direttiva **v-bind**, il placeholder "image" viene istanziato con il valore della proprietà image all'interno di **data**

Conditional rendering

Un file con estensione .html

```
<html>
    <head>
        <title> Prodotto in vendita </title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <script type="text/javascript" src="Vue/vue.min.js">
        </script>
    </head>
    <body>
        <div id="app">
            <p v-if="onSale && disp > 10">Disponibile</p>
            <p v-else-if="onSale && disp > 0">Ultime scorte!</p>
            <p v-else>Non disponibile</p>
        </div>
        <script type="text/javascript" src="mascherina.js">
        </script>
    </body>
</html>
```

Un file con estensione .js

```
var app = new Vue({
    el: '#app',
    data: {
        disp: 11,
        onSale: true
    }
});
```

Conditional rendering (segue)

Come si può facilmente intuire, a seconda dei valori *disp* e *onSale* del parametro **data** vengono mostrati cose differenti all'interno del risultante documento html

List rendering

Un file con estensione .html

```
<html>
    <head>
        <title> Prodotto in vendita </title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <script type="text/javascript" src="Vue/vue.min.js">
        </script>
    </head>
    <body>
        <div id="app">
            <ul>
                <li v-for="x in details">{{x.text}}</li>
            </ul>
            <div v-for="x in variants" :key="x.id">
                <p>{{x.color}}</p>
            </div>
        </div>
        <script type="text/javascript" src="mascherina.js">
        </script>
    </body>
</html>
```

List rendering (segue)

Un file con estensione .js

```
var app = new Vue({
    el: '#app',
    data: {
        details: [
            {text: '95% Cotone - 5% Elastan'},
            {text: 'Lavabile e riutilizzabile'},
            {text: 'Utilizzo ad uso civile, non medicale'}
        ],
        variants: [
            {id: 2241, color: 'lightblue'},
            {id: 2242, color: 'black'},
            {id: 2243, color: 'white'}
        ]
    }
});
```

List rendering (segue)

La direttiva **v-for** permette di scandire un valore di tipo array di una proprietà all'interno del parametro **data**. Quindi, tramite

```
<ul>
    <li v-for="x in details">{{x.text}}</li>
</ul>
```

viene creata una lista html non ordinata, un elemento per ciascun elemento dell'array details nel file .js.

Invece, tramite

```
<div v-for="x in variants" :key="x.id">
    <p>{{x.color}}</p>
</div>
```

viene creato un div html per ciascun elemento dell'array variants nel file .js. La direttiva **:key** potrà eventualmente essere utilizzata da Vue per riferirsi in modo non ambiguo agli elementi dell'array.

Catturare gli eventi

Un file con estensione .html

```
<html>
    <head>
        <title> Prodotto in vendita </title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <script type="text/javascript" src="Vue/vue.min.js">
        </script>
    </head>
    <body>
        <div id="app">
            <button v-on:click="cart = cart + 1">
                Aggiungi al carrello
            </button>
            <p>Carrello ({{cart}})</p>
        </div>
        <script type="text/javascript" src="mascherina.js">
        </script>
    </body>
</html>
```

Un file con estensione .js

```
var app = new Vue({
    el: '#app',
    data: {
        cart: 0
    }
});
```

Catturare gli eventi - Utilizzo del parametro method

```
<html>
    <head>
        <title> Prodotto in vendita </title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <script type="text/javascript" src="Vue/vue.min.js">
        </script>
    </head>
    <body>
        <div id="app">
            <button v-on:click="addToCart()">
                Aggiungi al carrello
            </button>
            <p>Carrello ({{cart}})</p>
        </div>
        <script type="text/javascript" src="mascherina2.js">
        </script>
    </body>
</html>
```

```
var app = new Vue({
    el: '#app',
    data: {
        cart: 0
    },
    methods: {
        addToCart: function() {
            this.cart += 1;
        }
    }
});
```

Catturare gli eventi - Utilizzo del parametro method (segue)

- Tramite la direttiva **v-on** mettiamo in ascolto Vue sull'elemento del DOM su cui viene applicato (nel nostro caso sul click di un button). In particolare, oltre che una semplice espressione, **v-on** accetta anche chiamate a funzioni

- Queste funzioni vengono definite all'interno del parametro **methods** di un oggetto Vue. Le funzioni possono accedere alle varie proprietà definite in **data** dello stesso oggetto tramite *this*, come avviene nella maggior parte dei linguaggi di programmazione

Un esempio completo - Foglio di stile .css

```
body {
    font-family: tahoma;
    color:#282828;
    margin: 0px;
}

.nav-bar {
    background: linear-gradient(-90deg, #84CF6A, #16C0B0);
    height: 60px;
    margin-bottom: 15px;
}

.product {
    display: flex;
    flex-flow: wrap;
    padding: 1rem;
}

img {
    border: 1px solid #d8d8d8;
    width: 70%;
    margin: 40px;
    box-shadow: 0px .5px 1px #d8d8d8;
}

.product-image {
    width: 80%;
}
```

Un esempio completo - Foglio di stile .css (segue)

```
.product-image,
.product-info {
    margin-top: 10px;
    width: 50%;
}

.color-box {
    width: 70px;
    height: 50px;
    line-height: 50px;
    margin-top: 5px;
    text-align: center;
    white-space: nowrap;
}

.cart {
    margin-right: 25px;
    float: right;
    border: 1px solid #d8d8d8;
    padding: 5px 20px;
}

button {
    margin-top: 30px;
    border: none;
    background-color: #1E95EA;
    color: white;
    height: 40px;
    width: 100px;
    font-size: 14px;
}
```

Un esempio completo - Foglio di stile .css (segue)

```
.disabledButton {
    background-color: #d8d8d8;
}

.review-form {
    width: 400px;
    padding: 20px;
    margin: 40px;
    border: 1px solid #d8d8d8;
}

input {
    width: 100%;
    height: 25px;
    margin-bottom: 20px;
}

textarea {
    width: 100%;
    height: 60px;
}

.tab {
    margin-left: 20px;
    cursor: pointer;
}

.activeTab {
    color: #16C0B0;
    text-decoration: underline;
}
```

Un esempio completo - File con estensione .html

```
<html>
    <head>
        <title> Prodotto in vendita </title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <link rel="stylesheet" type="text/css" href="style.css" />
        <script type="text/javascript" src="Vue/vue.min.js">
        </script>
    </head>
    <body>
        <div class="nav-bar"></div>
        <div id="app">
            <div class="cart">
                <p>Carrello ({{cart}})</p>
            </div>
            <div class="product">
                <div class="product-image">
                    <img v-bind:src="image" />
                </div>
                <div class="product-info">
                    <h1>Prodotto in vendita: {{product}}</h1>
                    Descrizione: {{description}}
                    <p v-if="disp > 10 && onSale">Disponibile</p>
                    <p v-else-if="disp > 0 && onSale">Ultime scorte!</p>
                    <p v-else>Non disponibile</p>
                    <ul>
                        <li v-for="x in details">{{x.text}}</li>
                    </ul>
```

Un esempio completo - File con estensione .html (segue)

```
                    <div v-for="x in variants" :key="x.id" class="color-box"
                    v-bind:style="{backgroundColor: x.htmlColor}">
                        <p v-on:click="updateImage(x.image)">{{x.color}}</p>
                    </div>
                    <button v-on:click="addToCart()"
                    v-bind:disabled="!onSale || disp==0"
                    v-bind:class="{disabledButton: !onSale || disp==0}">
                        Aggiungi al carrello
                    </button>
                </div>
            </div>
        </div>
        <script type="text/javascript" src="mascherina.js">
        </script>
    </body>
</html>
```

Un esempio completo - File con estensione .js

```
var app = new Vue({
    el: '#app',
    data: {
        product: 'Mascherina',
        description: 'Mascherina chirurgica',
        image: './assets/celeste.png',
        disp: 9,
        onSale: true,
        details: [
            { text: '95% Cotone - 5% Elastan'},
            { text: 'Lavabile e riutilizzabile'},
            { text: 'Utilizzo ad uso civile, non medicale'}
        ],
        variants: [
            {id: 2241, color: 'celeste',
            image: './assets/celeste.png', htmlColor: 'lightblue'},
            {id: 2242, color: 'nera',
            image: './assets/nera.png', htmlColor: 'black'},
            {id: 2243, color: 'bianca',
            image: './assets/bianca.png', htmlColor: 'white'}
        ],
        cart: 0
    },
    methods: {
        addToCart: function() {
            this.disp = this.disp-1;
            this.cart = this.cart+1;
        },
        updateImage: function(im) {
            this.image = im;
        }
    }
});
```

SAPIENZA UNIVERSITÀ DI ROMA

Il parametro computed

```
<html>
    <head>
        <title> Prodotto in vendita </title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <link rel="stylesheet" type="text/css" href="style.css" />
        <script type="text/javascript" src="Vue/vue.min.js">
        </script>
    </head>
    <body>
        <div id="app">
            <h1>{{fullname}}</h1>
        </div>
        <script type="text/javascript" src="mascherina.js">
        </script>
    </body>
</html>
```

```
var app = new Vue({
    el: '#app',
    data: {
        name: 'Mario',
        surname: 'Rossi'
    },
    computed: {
        fullname: function() {
            return this.name + ' ' + this.surname;
        }
    }
});
```

Il parametro computed (segue)

- Le proprietà contenute all'interno del parametro **computed** assumono valori dinamici che possono dipendere da valori di altre proprietà del nostro oggetto Vue
- Tuttavia, essi possono essere usati nel documento html come se fossero proprietà statiche all'interno di **data**
- Vue è in grado di capire da quali proprietà "dipende" una certa proprietà in **computed** e quindi è in grado di aggiornarne il suo valore dinamicamente se dovesse cambiare una sua "dipendenza".

Ritorniamo al nostro esempio completo. Tenendo questo a mente, andiamo a modificare il nostro codice in modo tale che ogni colore della mascherina abbia una sua disponibilità, e quindi non globalmente come mostrato in precedenza

Un esempio completo - File con estensione .html

```
<html>
    <head>
        <title> Prodotto in vendita </title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <link rel="stylesheet" type="text/css" href="style.css" />
        <script type="text/javascript" src="Vue/vue.min.js">
        </script>
    </head>
    <body>
        <div class="nav-bar"></div>
        <div id="app">
            <div class="cart">
                <p>Carrello ({{cart}})</p>
            </div>
            <div class="product">
                <div class="product-image">
                    <img v-bind:src="image" />
                </div>
                <div class="product-info">
                    <h1>Prodotto in vendita: {{product}}</h1>
                    Descrizione: {{description}}
                    <p v-if="disp > 10 && onSale">Disponibile</p>
                    <p v-else-if="disp > 0 && onSale">Ultime scorte!</p>
                    <p v-else>Non disponibile</p>
                    <ul>
                        <li v-for="x in details">{{x.text}}</li>
                    </ul>
```

Un esempio completo - File con estensione .html (segue)

```
                    <div v-for="(x, index) in variants" :key="x.id" class="color-box"
                    v-bind:style="{backgroundColor: x.htmlColor}">
                        <p v-on:click="updateProduct(index)">{{x.color}}</p>
                    </div>
                    <button v-on:click="addToCart()"
                    v-bind:disabled="!onSale || disp==0"
                    v-bind:class="{disabledButton: !onSale || disp==0}">
                        Aggiungi al Carrello
                    </button>
                </div>
            </div>
        </div>
        <script type="text/javascript" src="mascherina.js">
        </script>
    </body>
</html>
```

Un esempio completo - File con estensione .js

```
var app = new Vue({
    el: '#app',
    data: {
        product: 'Mascherina',
        description: 'Mascherina chirurgica',
        selectedVariant: 0,
        details: [
            { text: '95% Cotone - 5% Elastan'},
            { text: 'Lavabile e riutilizzabile'},
            { text: 'Utilizzo ad uso civile, non medicale'}
        ],
        variants: [
            {id: 2241, color: 'celeste', disp:9, onSale:true,
            image: './assets/celeste.png', htmlColor: 'lightblue'},
            {id: 2242, color: 'nera', disp:11, onSale:false,
            image: './assets/nera.png', htmlColor: 'black'},
            {id: 2243, color: 'bianca', disp:12, onSale:true,
            image: './assets/bianca.png', htmlColor: 'white'}
        ],
        cart: 0
    },
```

Un esempio completo - File con estensione .js (segue)

```
    methods: {
        addToCart: function() {
            this.variants[this.selectedVariant].disp -= 1;
            this.cart += 1;
        },
        updateProduct: function(i) {
            this.selectedVariant = i;
        }
    },
    computed: {
        onSale: function(){
            return this.variants[this.selectedVariant].onSale;
        },
        disp: function(){
            return this.variants[this.selectedVariant].disp;
        },
        image: function(){
            return this.variants[this.selectedVariant].image;
        }
    }
});
```

Components: data, template e props

- Tramite i **components** messi a disposizione da Vue è possibile organizzare il codice in moduli riutilizzabili e ben strutturati
- Per registrare un component si può utilizzare:
  Vue.component('nomeComponent', { OPTIONS });
  dove OPTIONS contiene parametri quali **data**, **methods** e **computed** (che assumono lo stesso significato di un qualsiasi oggetto Vue), in aggiunta ad altri parametri come ad esempio:
  - **props:**, che contiene proprietà che consentono di configurare un componente
  - **template:**, codice HTML che specifica come deve essere visualizzato il componente

  Nota bene la differenza: In un oggetto Vue, il parametro **el:** effettua il binding ad un elemento del DOM. In un componente Vue, il parametro **template:** viene usato per specificare il suo HTML

SAPIENZA UNIVERSITÀ DI ROMA

Components: data, template e props (segue)

Un file con estensione .js

```
Vue.component('visualize', {
    props: {
        list: {
            type: Array,
            required:true
        },
        value: {
            type: Number,
            required:true
        }
    },
    template: `
        <div>
            <div v-for="x in list">
                <h1>{{ x.text }}</h1>
            </div>
            <h2>{{ value }}</h2>
        </div>`
});

var app = new Vue({
    el: '#app',
    data: {
        groceryList: [
            { id: 0, text: 'Vegetables' },
            { id: 1, text: 'Cheese' },
            { id: 2, text: 'Whatever else humans are supposed to eat' }
        ],
        a: 10
    }
});
```

Components: data, template e props (segue)

Un file con estensione .html

```
<html>
    <head>
        <script type="text/javascript" src="Vue/vue.min.js"></script>
    </head>
    <body>
        <div id="app">
            <visualize v-bind:list="groceryList" v-bind:value="a"></visualize>
        </div>
        <script type="text/javascript" src="indexScript.js"></script>
    </body>
</html>
```

Ritorniamo al nostro esempio completo. Tenendo questo a mente, andiamo a modificare il nostro codice in modo tale da utilizzare un component Vue

Un esempio completo - File con estensione .html

```
<html>
    <head>
        <title> Prodotto in vendita </title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <link rel="stylesheet" type="text/css" href="style.css" />
        <script type="text/javascript" src="Vue/vue.min.js">
        </script>
    </head>
    <body>
        <div class="nav-bar"></div>
        <div id="app">
            <div class="cart">
                <p>Carrello ({{cart}})</p>
            </div>

            <mascherina v-on:add-to-cart="updateCart"></mascherina>

        </div>
        <script type="text/javascript" src="mascherina.js">
        </script>
    </body>
</html>
```

Un esempio completo - File con estensione .js

```
Vue.component('mascherina', {
    template: `
        <div class="product">
            <div class="product-image">
                <img v-bind:src="image"/>
            </div>
            <div class="product-info">
                <h1>Prodotto in vendita: {{product}}</h1>
                Descrizione: {{description}}
                <p v-if="disp > 10 && onSale">Disponibile</p>
                <p v-else-if="disp > 0 && onSale">Ultime scorte!</p>
                <p v-else>Non disponibile</p>
                <ul>
                    <li v-for="x in details">{{x.text}}</li>
                </ul>
                <div v-for="(x, index) in variants" :key="x.id" class="color-box"
                v-bind:style="{backgroundColor: x.htmlColor}">
                    <p v-on:click="updateProduct(index)">{{x.color}}</p>
                </div>
                <button v-on:click="addToCart()"
                v-bind:disabled="!onSale || disp==0"
                v-bind:class="{disabledButton: !onSale || disp==0}">
                    Aggiungi al carrello
                </button>
            </div>
        </div>
    `,
```

Un esempio completo - File con estensione .js (segue)

```
data: function(){
    return {
        product: 'Mascherina',
        description: 'Mascherina chirurgica',
        selectedVariant: 0,
        details: [
            { text: '95% Cotone - 5% Elastan'},
            { text: 'Lavabile e riutilizzabile'},
            { text: 'Utilizzo ad uso civile, non medicale'}
        ],
        variants: [
            {id: 2241, color: 'celeste', disp:9, onSale:true,
            image: './assets/celeste.png', htmlColor: 'lightblue'},
            {id: 2242, color: 'nera', disp:11, onSale:false,
            image: './assets/nera.png', htmlColor: 'black'},
            {id: 2243, color: 'bianca', disp:12, onSale:true,
            image: './assets/bianca.png', htmlColor: 'white'}
        ]
    };
},

methods: {
    addToCart: function() {
        this.variants[this.selectedVariant].disp -= 1;
        this.$emit('add-to-cart');
    },
    updateProduct: function(i) {
        this.selectedVariant = i;
    }
},
```

Un esempio completo - File con estensione .js (segue)

```
    computed: {
        onSale: function(){
            return this.variants[this.selectedVariant].onSale;
        },
        disp: function(){
            return this.variants[this.selectedVariant].disp;
        },
        image: function(){
            return this.variants[this.selectedVariant].image;
        }
    }
});

var app = new Vue({
    el: '#app',
    data: {
        cart: 0
    },
    methods: {
        updateCart: function() {
            this.cart += 1;
        }
    }
});
```