# Il framework Vue.js

Lorenzo Marconi


SAPIENZA
UNIVERSITÀ DI ROMA

Anno accademico 2021/2022

# 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
- Versioni: 1.0 (2015), 2.0 (2016), 3.0 (2020)
  In questa esercitazione useremo Vue 3

# 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

# 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 rilevanti di questo framework
- Per maggiori dettagli rimandiamo alla documentazione ufficiale (vuejs.org/guide)

# 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 src="https://unpkg.com/vue@3"></script>
    </head>
    <body>
        <div id="app">
            <h1>Prodotto in vendita: {{product}}</h1>
            Descrizione: {{description}}
        </div>
        <script type="application/javascript" src="app.js"></script>
    </body>
</html>
```

## Un file con estensione `.js`

```
let app = Vue.createApp({
    data() {
        return {
            product: 'Cappello',
            description: 'Cappello di lana'
        }
    }
});

app.mount('#app');
```

# Come "montare" applicazioni Vue

Analizziamo l'esempio fornito nella precedente slide:

- Lato HTML, importiamo Vue.js con la seguente chiamata:
  `<script src="https://unpkg.com/vue@3"></script>`
- Nel file con estensione `.js` viene creato un nuovo oggetto tramite
  `Vue.createApp();`
  funzione usata per creare il *root component* dell'applicazione
- La funzione **`mount()`** effettua il **mounting** dell'oggetto Vue su uno specifico elemento del DOM. Nel nostro esempio abbiamo **`mount`**`('#app')`, pertanto l'oggetto verrà collegato all'elemento HTML con identificativo `'app'`.
- Le doppie parentesi graffe {{ }} nel codice HTML vengono usate come placeholders per effettivi valori restituiti da **`data()`**. Nel nostro esempio abbiamo {{ `product` }} e {{ `description` }}, che verranno istanziati rispettivamente con `'Cappello'` e `'Cappello di lana'` nel risultante documento HTML.

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 src="https://unpkg.com/vue@3"></script>
    </head>
    <body>
        <div id="app">
            <img v-bind:src="image" />
        </div>
        <script type="application/javascript" src="app.js"></script>
    </body>
</html>
```

## Un file con estensione `.js`

```
let app = Vue.createApp({
    data() {
        return { image: './assets/nero.jpg' }
    }
});
app.mount('#app');
```

# Binding di attributi HTML

Analizziamo l'esempio fornito nella precedente slide:

- Nel file HTML troviamo:

  `v-bind:src="image"`

  La *direttiva* **`v-bind`** (scorciatoia: `:`) 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
- Grazie alla direttiva **`v-bind`**, il placeholder `image` viene istanziato con il valore della proprietà image all'interno di **`data`**
- Come prima, la funzione **`mount()`** ci permette di "montare" l'oggetto Vue su uno specifico elemento del DOM

# 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 src="https://unpkg.com/vue@3"></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="application/javascript" src="app.js"></script>
    </body>
</html>
```

## Un file con estensione `.js`

```
let app = Vue.createApp({
    data() {
        return {
            disp: 11,
            onSale: true
        }
    }
});
app.mount('#app');
```

# 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

La direttiva **`v-show`** ha un funzionamento simile a **`v-if`** ma nasconde l'elemento tramite CSS anziché aggiungerlo/rimuoverlo dal DOM. Questo può risultare vantaggioso in termini di performance, tuttavia $(i)$ consente più faticosamente di ramificare il flusso di controllo e $(ii)$ va usato tenendo a mente che l'elemento nascosto sarà comunque esistente

# 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 src="https://unpkg.com/vue@3"></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="application/javascript" src="app.js"></script>
    </body>
</html>
```

SAPIENZA UNIVERSITÀ DI ROMA

# List rendering (segue)

Un file con estensione `.js`

```
let app = Vue.createApp({
    data() {
        return {
            details: [
                {text: 'Lana a maglia'},
                {text: 'Pompon tipo raccoon'},
                {text: 'Felpato foderato'}
            ],
            variants: [
                {id: 2241, color: 'lightgrey'},
                {id: 2242, color: 'black'},
                {id: 2243, color: 'beige'}
            ]
        }
    }
});
app.mount('#app');
```

# 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 src="https://unpkg.com/vue@3"></script>
    </head>
    <body>
        <div id="app">
            <button v-on:click="cart = cart + 1">
                Aggiungi al carrello
            </button>
            <p>Carrello ({{cart}})</p>
        </div>
        <script type="application/javascript" src="app.js"></script>
    </body>
</html>
```

## Un file con estensione `.js`

```
let app = Vue.createApp({
    data() {
        return { cart: 0 }
    }
});
app.mount('#app');
```

# 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 src="https://unpkg.com/vue@3"></script>
    </head>
    <body>
        <div id="app">
            <button v-on:click="addToCart()">
                Aggiungi al carrello
            </button>
            <p>Carrello ({{cart}})</p>
        </div>
        <script type="application/javascript" src="app2.js"></script>
    </body>
</html>
```

```
let app = Vue.createApp({
    data() {
        return { cart: 0 }
    },
    methods: {
        addToCart: function() {
            this.cart++;
        }
    }
});
app.mount('#app');
```

# Catturare gli eventi - Utilizzo del parametro method (segue)

- Tramite la direttiva **`v-on`** (scorciatoia: **`@`**) 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

SAPIENZA UNIVERSITÀ DI ROMA

# 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;
}
```

SAPIENZA UNIVERSITÀ DI ROMA

# 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 src="https://unpkg.com/vue@3"></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="application/javascript" src="app.js"></script>
    </body>
</html>
```

# Un esempio completo - File con estensione .js

```
let app = Vue.createApp({
    data() { return {
        product: 'Cappello',
        description: 'Cappello di lana',
        image: './assets/grigio.jpg',
        disp: 9,
        onSale: true,
        details: [
            { text: 'Lana a maglia'},
            { text: 'Pompon tipo raccoon'},
            { text: 'Felpato foderato'}
        ],
        variants: [
            {id: 2241, color: 'grigio',
            image: './assets/grigio.jpg', htmlColor: 'lightgrey'},
            {id: 2242, color: 'nero',
            image: './assets/nero.jpg', htmlColor: 'black'},
            {id: 2243, color: 'beige',
            image: './assets/beige.jpg', htmlColor: 'beige'}
        ],
        cart: 0
    } },
    methods: {
        addToCart: function() {
            this.disp--;
            this.cart++;
        },
        updateImage: function(im) {
            this.image = im;
        }
    }
});
app.mount('#app');
```

# 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 src="https://unpkg.com/vue@3"></script>
    </head>
    <body>
        <div id="app">
            <h1>{{fullname}}</h1>
        </div>
        <script type="application/javascript" src="app.js"></script>
    </body>
</html>
```

```
let app = Vue.createApp({
    data() {
        return {
            firstname: 'Mario',
            lastname: 'Rossi'
        }
    },
    computed: {
        fullname: function() {
            return this.firstname + ' ' + this.lastname;
        }
    }
});
app.mount('#app');
```

# 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 restituite da **`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, per ogni colore, il prodotto 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 src="https://unpkg.com/vue@3"></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="application/javascript" src="app.js"></script>
    </body>
</html>
```

# Un esempio completo - File con estensione .js

```
let app = Vue.createApp({
    data() {
        return {
            product: 'Cappello',
            description: 'Cappello di lana',
            selectedVariant: 0,
            details: [
                { text: 'Lana a maglia'},
                { text: 'Pompon tipo raccoon'},
                { text: 'Felpato foderato'}
            ],
            variants: [
                {id: 2241, color: 'grigio', disp:9, onSale:true,
                image: './assets/grigio.jpg', htmlColor: 'lightgrey'},
                {id: 2242, color: 'nero', disp:11, onSale:false,
                image: './assets/nero.jpg', htmlColor: 'black'},
                {id: 2243, color: 'beige', disp:12, onSale:true,
                image: './assets/beige.jpg', htmlColor: 'beige'}
            ],
            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;
        }
    }
});
app.mount('#app');
```

# Components

- Oltre al root component, possiamo definire in generale un qualsiasi numero di **`components`**, in modo da organizzare il codice in moduli riutilizzabili e ben strutturati
- Per registrare un component si può utilizzare:

  `app.component('nomeComponent', { OPTIONS });`

  dove in `OPTIONS` è possibile specificare le proprietà già discusse **`data`**, **`methods`** e **`computed`**, in aggiunta ad altre quali:
  - **`props:`**, che contiene proprietà che potranno essere usate come attributi HTML
  - **`template:`**, codice HTML che specifica il modello su cui sarà basato il component

# Components (esempio)

```
let app = Vue.createApp({
    data() { return {
        groceryList: [
            { id: 0, text: 'Vegetables' },
            { id: 1, text: 'Cheese' },
            { id: 2, text: 'Whatever else humans are supposed to eat' }
        ],
        num: 10
    } }
});

app.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>`
});

app.mount('#app');
```

# Components (esempio)

Un file con estensione `.html`

```
<html>
    <head>
        <script src="https://unpkg.com/vue@3"></script>
    </head>
    <body>
        <div id="app">
            <visualize v-bind:list="groceryList" v-bind:value="num"></visualize>
        </div>
        <script type="application/javascript" src="app.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 src="https://unpkg.com/vue@3"></script>
    </head>
    <body>
        <div class="nav-bar"></div>
        <div id="app">
            <div class="cart">
                <p>Carrello ({{ cart }})</p>
            </div>

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

        </div>
        <script type="application/javascript" src="app.js"></script>
    </body>
</html>
```

# Un esempio completo - File con estensione .js

```
let app = Vue.createApp({
    data() {
        return { cart: 0 }
    },
    methods: {
        updateCart: function() {
            this.cart++;
        }
    }
});

app.component('product', {
    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>
```

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

```
            <button v-on:click="addToCart()"
            v-bind:disabled="!onSale || disp==0"
            v-bind:class="{disabledButton: !onSale || disp==0}">
                Aggiungi al carrello
            </button>
        </div>
    </div>
`,

data() {
    return {
        product: 'Cappello',
        description: 'Cappello di lana',
        selectedVariant: 0,
        details: [
            { text: 'Lana a maglia'},
            { text: 'Pompon tipo raccoon'},
            { text: 'Felpato foderato'}
        ],
        variants: [
            {id: 2241, color: 'grigio', disp:9, onSale:true,
            image: './assets/grigio.jpg', htmlColor: 'lightgrey'},
            {id: 2242, color: 'nero', disp:11, onSale:false,
            image: './assets/nero.jpg', htmlColor: 'black'},
            {id: 2243, color: 'beige', disp:12, onSale:true,
            image: './assets/beige.jpg', htmlColor: 'beige'}
        ]
    };
},
```

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

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

app.mount('#app');
```