Sviluppo Web con Vue.js

Chi sono

Marco Breveglieri

Software & Web Developer *@ABLS Team*
Blogger (www.compilaquindiva.com)
Host *@Delphi Podcast* (www.delphipodcast.com)

and sushi eater!

Introduzione

Che cos'è Vue.js?

Vue.js è un ***progressive framework*** per la costruzione di interfacce utente

Framework o libreria

Comparazione con altri celebri framework JavaScript

- Meglio chiamarla libreria, non framework
- Ha le prestazioni di React
- Usa convenzioni già viste in AngularJS (e altre librerie MVVM)

Caratteristiche

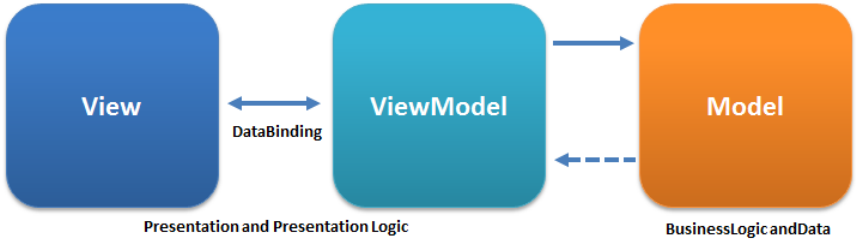
Incremental adoption

- Può essere gradualmente integrato in un'applicazione Web esistente
- Curva di apprendimento morbida

Obiettivi

Focalizzato sulla «View»

La libreria si concentra sulla gestione delle "viste" per generare e aggiornare la UI.



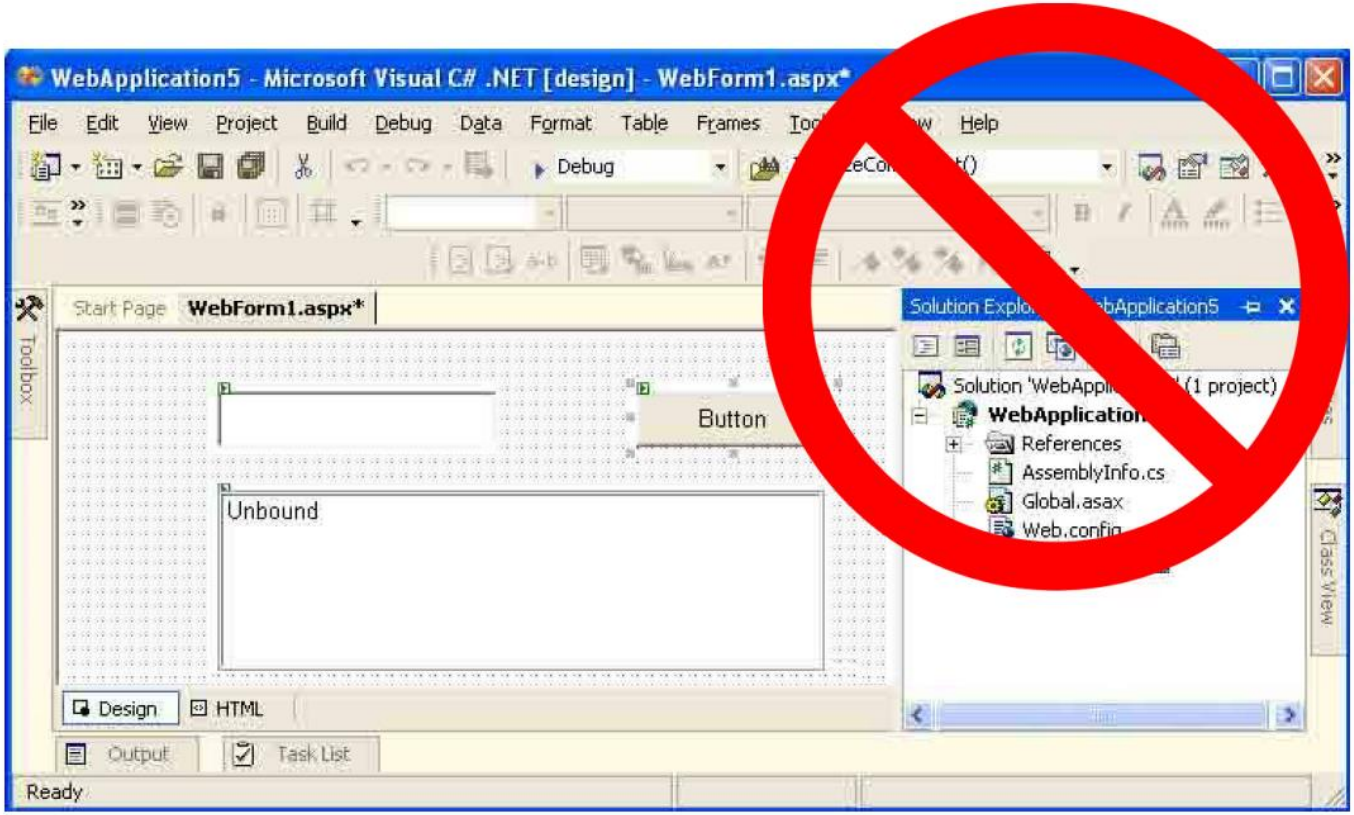
Possibilità

Perfetto anche per SPA sofisticate

- Supporta i moderni tool di sviluppo JavaScript
  - Node, WebPack, Browserify, ES2016, …
  - Approccio component-based
  - Disponibile *vue-cli*
- Numerosi componenti ed estensioni di terze parti

Premessa

Applicazioni Web oggi



Web Server "Fat" App: i pregi

- Mascherano la piattaforma HTTP
- Facili per sviluppatori abituati allo sviluppo desktop
- Favoriscono lo sviluppo rapido (RAD)
- Sono ideali per la "prototipazione"

Web Server "Fat" App: i difetti

- Mascherano la piattaforma HTTP
- Introducono informazioni ingombranti (es. ViewState)
- Consumo eccessivo di banda, memoria e CPU del server
- Scalabilità generale del sistema ridotta
- Separazione dei task designer/developer impossibile
- Unit/Integration test difficili da implementare

Linguaggi e standard

- Si basano sui linguaggi e sui protocolli standard
  - **HTML**(5)
  - **CSS**(3)
  - **JavaScript**

Tecnologie must-know

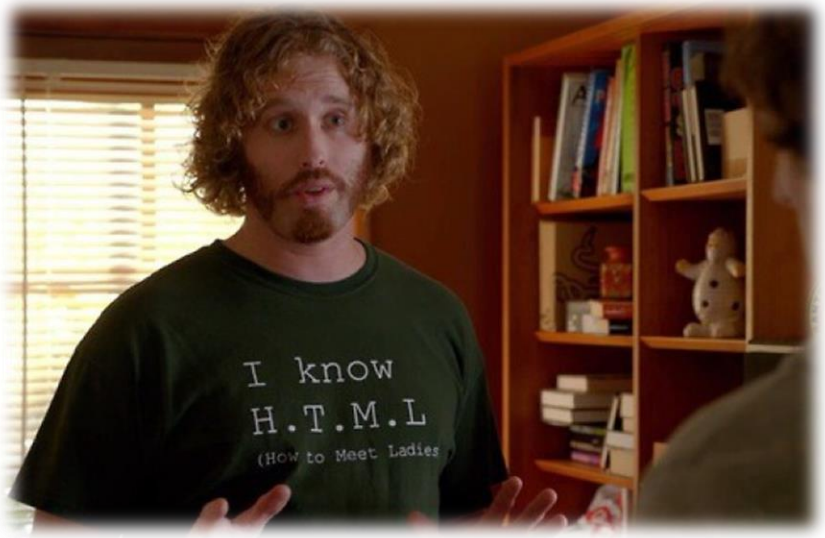
Single Page Application

- Applicazione Web (o parte di essa) costituita da una sola pagina
- Mima le caratteristiche di un'applicazione desktop
- Fornisce una esperienza utente più fluida
- Non ricarica l'intera pagina del browser per aggiornarsi
- Riflette cambiamenti dinamicamente su input dell'utente
- Trasferisce dati e risorse in modo ottimizzato
- Sfrutta le caratteristiche della piattaforma (il browser e le sue API)

Perché Vue.js?

Accessibile

Ti basta conoscere
HTML, CSS e JavaScript.

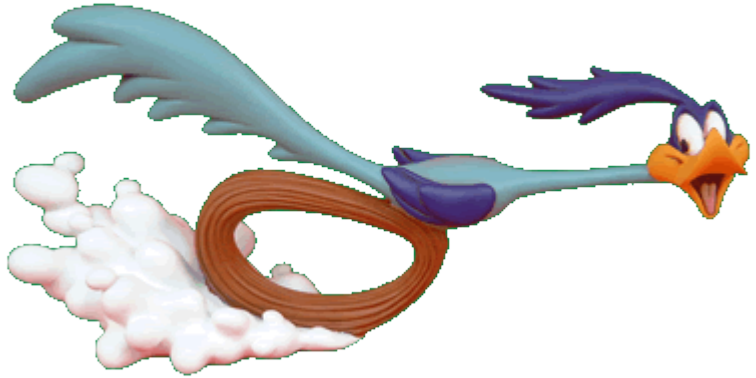


Versatile

Può essere usato in progetti sia piccoli che di grandi dimensioni.

Performante

Estremamente veloce grazie al proprio Virtual DOM.



Compatto

Il runtime è molto piccolo: solo ~19Kb.

Primi passi

Importazione

Importare lo script di Vue nella pagina:

```
<script src="https://unpkg.com/vue"></script>
```

Interpolazione

Creare il markup dell'applicazione:

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

Modello

Costruire un'**istanza dell'oggetto Vue**:

```
<script>
    var app = new Vue({
        el: "#app",
        data: {
            message: "Hello Vue!"
        }
    });
</script>
```

Modalità d'uso

Declarative Rendering

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

<script>
    var app = new Vue({
        el: "#app",
        data: {
            message: "Hello Vue!"
        }
    });
</script>
```

Attribute Binding

```
<div id="app">
  <p v-bind:title="message">
      Sposta qui il mouse per vedere il messaggio.
  </p>
</div>

<script>
    var app = new Vue({
        el: "#app",
        data: {
            message: "Hello Vue!"
        }
    });
</script>
```

Conditionals

```
<div id="app">
    <p v-if="visible">Ora mi vedi!</p>
</div>

<script>
    var app = new Vue({
        el: "#app",
        data: {
            visible: true
        }
    });
</script>
```

Loop

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

<script>
    var app = new Vue({
        el: "#app",
        data: {
            todoList: [
                { text: "Imparare Vue" },
                { text: "Integrarlo con Delphi" },
                { text: "Costruire applicazioni spaziali!" }
            ]
        }
    });
</script>
```

Eventi

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

<script>
    var app = new Vue({
        el: "#app",
        data: {
            message: "REDRUM"
        },
        methods: {
            reverseMessage: function () {
                this.message = this.message.split("").reverse().join("");
            }
        }
    });
</script>
```

Binding

Two-Way Binding

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

<script>
    var app = new Vue({
        el: "#app",
        data: {
            message: "REDRUM"
        }
    });
</script>
```

Filtri

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

<script>
    var app = new Vue({
        el: "#app",
        data: {
            message: "redrum"
        },
        filters: {
            reverse: function (value) {
                return value.split("").reverse().join("");
            },
            toUpper: function (value) {
                return value.toUpperCase();
            },
        }
    });
</script>
```

Computed Properties

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

<script>
    var app = new Vue({
        el: "#app",
        data: {
            message: "REDRUM"
        },
        computed: {
            reversedMessage: function () {
                return this.message.split("").reverse().join("");
            }
        }
    });
</script>
```

Componenti!

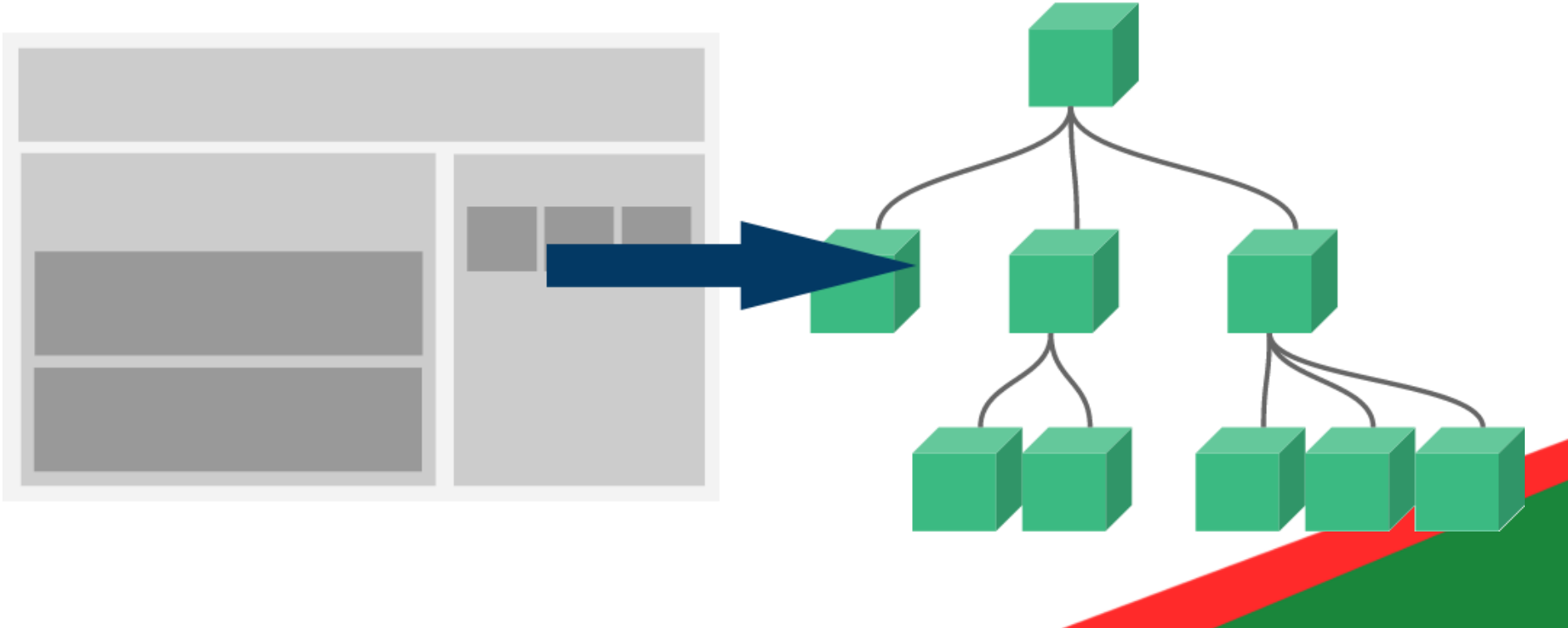
Componenti

Ogni interfaccia utente può essere scomposta in **componenti**, ossia parti

- più piccole
- incapsulate
- indipendenti
- riutilizzabili

Componenti

UI Decomposition



Creazione

Creare un componente è molto semplice…

```
Vue.component('todo-item', {
    template: '<li>Cosa da fare</li>'
})
```

Utilizzo

…e anche utilizzarlo. ☺

```
<ul>
      <todo-item></todo-item>
      <todo-item></todo-item>
      <todo-item></todo-item>
</ul>
```

Proprietà

Aggiungiamo proprietà per renderlo "configurabile"...

```
Vue.component('todo-item', {
    props: ['task'],
    template: '<li>{{ task.text }}</li>'
})
```

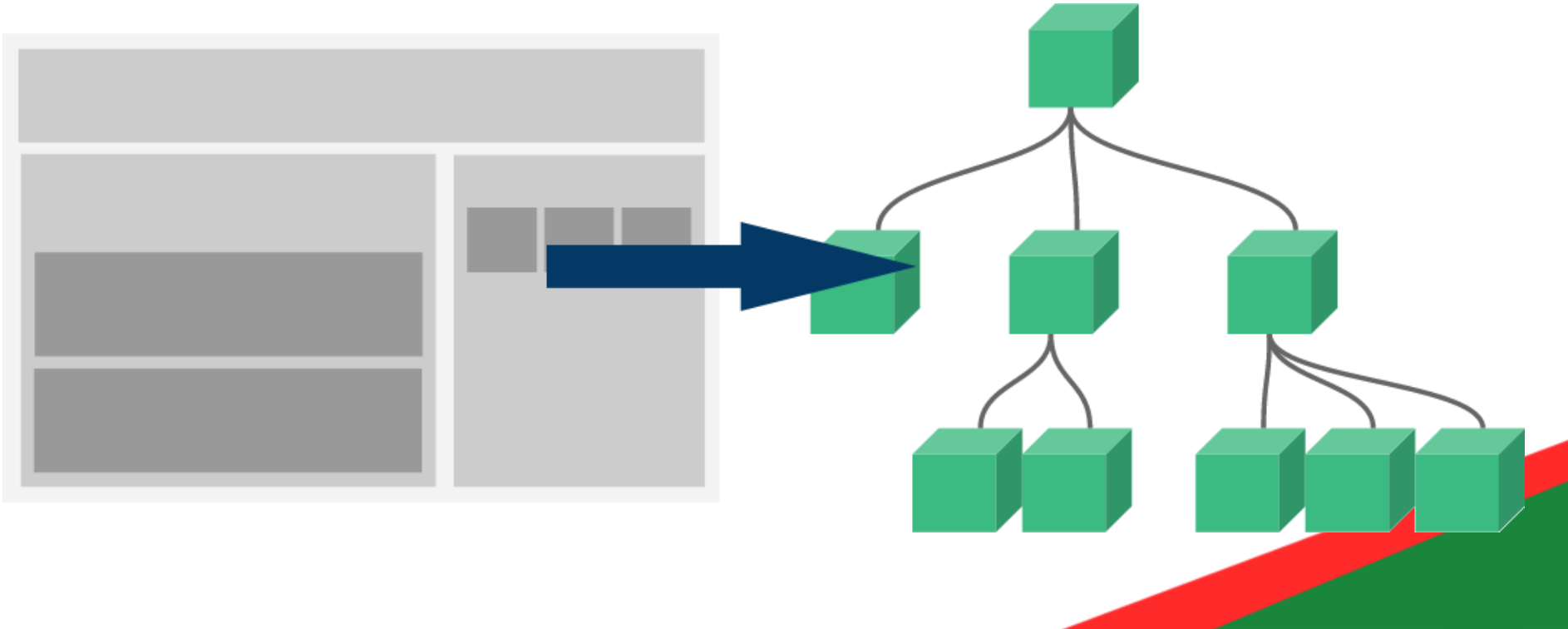
Valori

...e successivamente **valorizziamo la proprietà**.

```
<ul>
    <todo-item v-bind:task="{ text: 'Task 1' }"></todo-item>
    <todo-item v-bind:task="{ text: 'Task 2' }"></todo-item>
    <todo-item v-bind:task="{ text: 'Task 3' }"></todo-item>
</ul>
```

Componenti

Ricordate lo schema iniziale?



Struttura di un'applicazione

Esempio:

```
<app-title></app-title>
<app-nav>
    <app-menuitem></app-menuitem>
    <app-menuitem></app-menuitem>
    <app-menuitem></app-menuitem>
</app-nav>
<app-view>
    <app-sidebar>
        <app-contextmenu></app-contextmenu>
    </app-sidebar>
    <app-content>
        <app-content-title></app-content-title>
    </app-content>
</app-view>
<app-footer></app-footer>
```

Dietro le quinte

Istanza di Vue

Le istanze Vue sono dotate di **estensioni**.

```
var data = { a: 1 };

var vm = new Vue({
    el: '#example',
    data: data
});

vm.$data === data; // -> true

vm.$el === document.getElementById('example') ; // -> true

// $watch is an instance method
vm.$watch('a', function (newVal, oldVal) {
    // this callback will be called when `vm.a` changes
});
```

Ciclo di vita

Gestione degli eventi del ciclo di vita

- `created`
- `mounted`
- `updated`
- `destroyed`

KEEP
CALM
AND
ASK
QUESTIONS

Thanks!